ECE 788 – Pattern Recognition, Machine Learning and Information Theory

Scribe Notes

The following are scribe notes taken for my course ECE 788 – Pattern Recognition, Machine Learning and Information Theory during the Fall 2016 semester. They have not been seriously edited, but they may still be of some use. (You have been warned!)

With many thanks to the scribes (Annan Dong, Salman Habib, Wei Jiang, Maximilian Klass, Shruti Kulkarni, Mustafa Ozen, Anakha Vasanthakumaribabu, Maike Venhaus and Anqi Xiong) and to all who attended or audited this course,

Osvaldo Simeone

Newark, Dec. 2016

1. Introduction

1 Introduction

Some definitions

- *Pattern recognition (engineering)* - automatic discovery of regularities in data for decision/prediction/data mining.
- *Machine learning (computer science)* - development of efficient algorithms to learn models from data.
- *Information theory* - quantification of informational relationships among data ⇒ performance criteria for learning
- Learning is useful when it is too difficult or too costly to design an algorithm that solves a specific problem based on domain knowledge.
- Remark: Information theory also provides a conceptual framework to derive learning algorithms, namely Minimum Description Length (MDL), to be discussed.

Learning tasks

1. Supervised learning: Inference of function $t = y(x)$ from a set of training examples $\{x_n, t_n\}_{n=1}^N$ with the goal of minimizing the generalization error (error over new data points), see Fig. 1. There are two types of supervised learning:
   - Classification (t discrete)
   - Regression (t continuous).

2 Unsupervised learning: Inference of a function $y(x)$ or generating mechanism from training examples $\{x_n\}_{n=1}^N$ with the goal of discovering hidden structure in the data, see Fig. 2. We can distinguish the following tasks:
   - Clustering - group similar examples
   - Density estimation - determine data distribution
   - Dimensionality reduction - describe data in a smaller space

3 Reinforcement Learning: Inference of optimal on-line action based on rewards/punishment obtained as a result of previous actions.

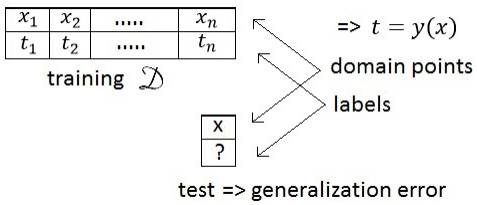


Figure 1: Illustration of supervised learning. The dependent variable, or label, t is to be predicted based on the domain variable x.

Additional category of learning tasks

- Passive vs. active learning: a passive learner is given the training example; an active learner can choose these (e.g. choosing songs and asking experts to clarify them).
- Offline vs. online learning: Supervised or unsupervised learning can also be formulated as online learning in contrast to batch learning (see Fig. 3).

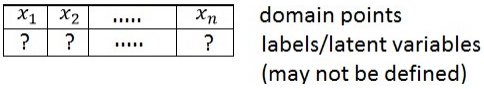


Figure 2: Illustration of unsupervised learning for clustering or dimensionality reduction.

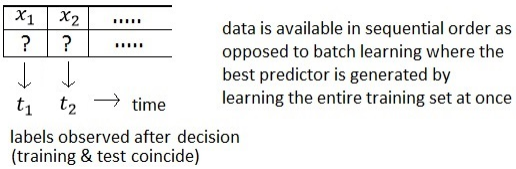


Figure 3: Illustration of online supervised learning.

2 Example: Regression (Supervised Learning)

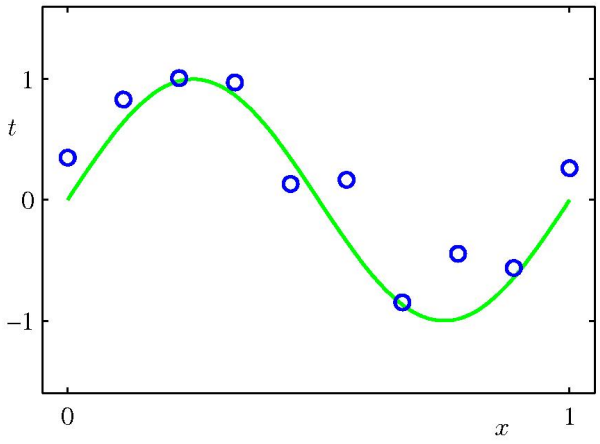


Figure 4: Example of a training set (blue points). The green line corresponds to the mean of the "true", and unknown, distribution $p_o(x,t)$ used to generate the data set (from [1]).

- Training set $\mathcal{D}$: $\mathbf{x} = (x_1, ..., x_N)^T$ domain points or instances and $\mathbf{t} = (t_1, ..., t_N)$ labels (Fig. 4)
- Each training data point (x_n, t_n) is statistically equivalent and drawn from the "true" (and unknown) distribution $p_0(x,t)$:

$$(x_n, t_n) \underset{i.i.d.}{\sim} p_0(x,t), \ i = 1, ..., N. \tag{1}$$

- We would like to learn a machine that predicts t from x for a *new* pair $(x,t) \sim p_0(x,t)$, which is independent of the training set.

- Learning = Conversion of experience into expertise or knowledge ≠Memorizing.
- Learning is clearly impossible without making further assumptions (*No free lunch theorem*).
- *Assumption*: Data comes from some parameterized class of models (hypothesis class).
- *Goal of learning*: Find best model in the hypothesis class in terms of generalization error, i.e., error on a new independent pair $(x,t) \sim p_0(x,t)$.
- This is the nature of the scientific method: Theories should be able to provide predictions under generalized conditions as compared to the available data.
- For this example, we choose the following *probabilistic discriminative model* (Fig. 5):

$$t = y(x,w) + z, \text{ where } z \sim \mathcal{N}(0, \beta^{-1}) \tag{2}$$

$$\text{and } y(x,\mathbf{w}) = \sum_{j=0}^{M} w_j x^j \tag{3}$$

- *Model parameters*: $(\mathbf{w}, \beta)$ for any fixed model order M.

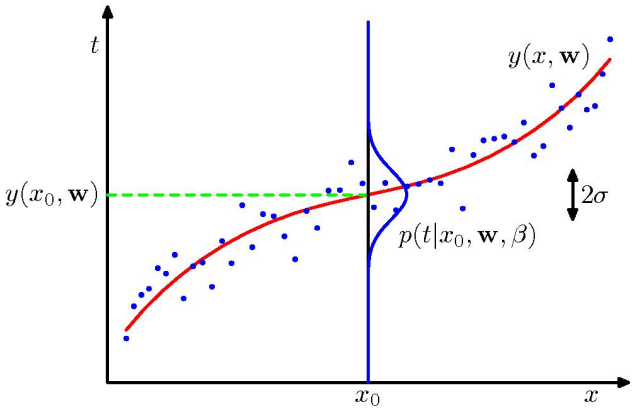


Figure 5: Illustration of the probabilistic discriminative class of models $p(t|x,\mathbf{w},\beta)$ selected for the regression example (from [1]).

- Conditional pdf of each value of the label t given the domain point x according to the model:

$$p(t|x,\mathbf{w},\beta) = \mathcal{N}(t|y(x,\mathbf{w}),\beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} \exp\left(\frac{-(t-y(x,\mathbf{w}))^2}{2\beta^{-1}}\right). \tag{4}$$

- Conditional pdf of the labels $\mathbf{t}$ given the domain points $\mathbf{x}$ in the training data set $\mathcal{D} = \{\mathbf{x},\mathbf{t}\}$:

$$p(\mathbf{t}|\mathbf{x},\mathbf{w},\beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|x_n,\mathbf{w},\beta^{-1}) \text{ likelihood function of } (\mathbf{w},\beta) \tag{5}$$

or

$$\ln p(\mathbf{t}|\mathbf{x},\mathbf{w},\beta) = \sum_{n=1}^{N} \ln \mathcal{N}(t_n|x_n,\mathbf{w},\beta^{-1}) \text{ log-likelihood function of } (\mathbf{w},\beta)$$

$$= -\frac{\beta}{2}\sum_{n=1}^{N}(y(x_n,\mathbf{w}) - t_n)^2 + \frac{N}{2}\ln\frac{\beta}{2\pi}. \tag{6}$$

- Goal of learning: Find the "best" model $(\mathbf{w},\beta)$ so that it "generalizes" over unobserved examples $(x,t) \sim p_0(x,t)$.

Remarks

- The "true" model does not have to be included in the family of assumed models.
- Choosing the wrong family of models may cause failure to learn ("inductive bias" problem).
- Example: B.F. Skinner's experiment on pigeons:

  - serve food at regular intervals, irrespective of pigeon's behavior;

  - pigeons learned to keep repeating the action they were doing the first time food was served (wrong model!).
- The choice of the model depends on many factors including ethical considerations. For instance, including as independent variables in x the ZIP code of an individual seeking credit at a bank may discriminate against immigrants or minorities [9].

3 Maximum Likelihood (ML) Approach

- Maximize the probability of the training set (trust the data!):

$$\underbrace{\underset{\mathbf{w},\beta}{\text{minimize}} -\frac{1}{N}\sum_{n=1}^{N}\ln p(t_n|x_n,\mathbf{w},\beta)}_{\Rightarrow\text{ML estimate of }\mathbf{w}\text{ and }\beta:\ \mathbf{w}_{ML},\beta_{ML}}. \tag{7}$$

- Predictive distribution for new observation (x,t): $p(t|x,\mathbf{w}_{ML},\beta_{ML})$
- Using the assumed probabilistic discriminative model, the ML problem is:

$$\underset{\mathbf{w},\beta}{\text{minimize}}\ \beta\underbrace{\frac{1}{N}\sum_{n=1}^{N}(y(x_n,\mathbf{w})-t_n)^2}_{L(\mathbf{w})\text{: mean squared loss}}-\ln\frac{\beta}{2\pi} \tag{8}$$

- The squared loss $\ell(y,t)=(y-t)^2$ measures the squares of the green segments seen in Fig. 6.

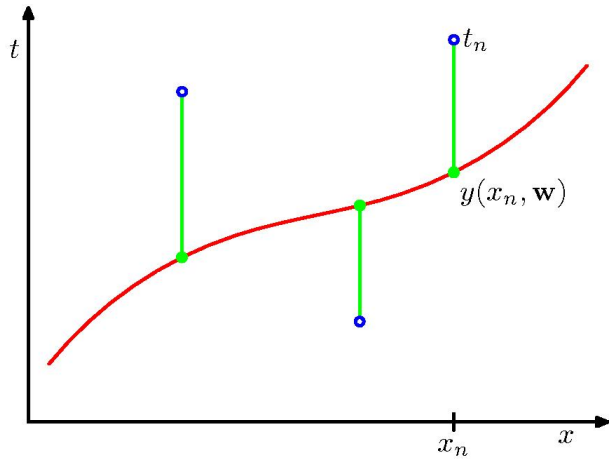


Figure 6: Illustration of the error terms $|t_n-y_n|$ (from [1]).

- ML estimates:

$$\begin{cases}\mathbf{w}_{ML}\leftarrow\min\limits_{\mathbf{w}}L(\mathbf{w})\\ \frac{1}{\beta_{ML}}=\frac{1}{N}L(\mathbf{w}_{ML})\end{cases} \tag{9}$$

- $\mathbf{w}_{ML}$ is the solution of a least squares problem and can be found in closed form as discussed in Problem 1 (see, e.g., [3]).

- Predictive distribution for new observation (x, t), see Fig. 7:

$$p(t|x, \mathbf{w}_{ML}, \beta_{ML}) = \mathcal{N}(t|y(x, \mathbf{w}_{ML}), \beta_{ML}^{-1}). \tag{10}$$

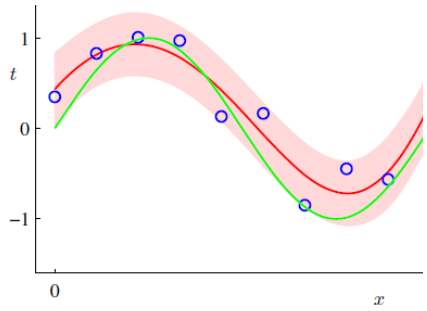


Figure 7: Illustration of a predictive distribution for a new observation (this is actually obtained via the Bayesian approach to be discussed below) (from [1]).

- ML tries only to fit the data by minimizing $L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(y(x_n, \mathbf{w}) - t_n)^2$, which is counterproductive for learning if N is not very large (as compared to the model order M). Therefore, ML may lead to overfitting, see Fig. 8.

- Overfitting = "Remembering but not learning".

- Learning aims at minimizing the error over unseen examples and hence at minimizing the generalization error (also known as true error or true/generalization/average risk/loss)

$$L_0(\mathbf{w}) = E_{(x,t)\sim p_0(x,t)}[(y(x, \mathbf{w}) - t)^2], \tag{11}$$

which is averaged over the unknown "true" distribution $p_0(x, t)$, while ML learning only minimizes the empirical error over the training set. Note that in some references the generalization error is defined as the difference between the training error $L(\mathbf{w})$ and the generalization error $L_0(\mathbf{w})$.

- Overfitting occurs when the generalization error is significantly larger than the training error $L(\mathbf{w})$.

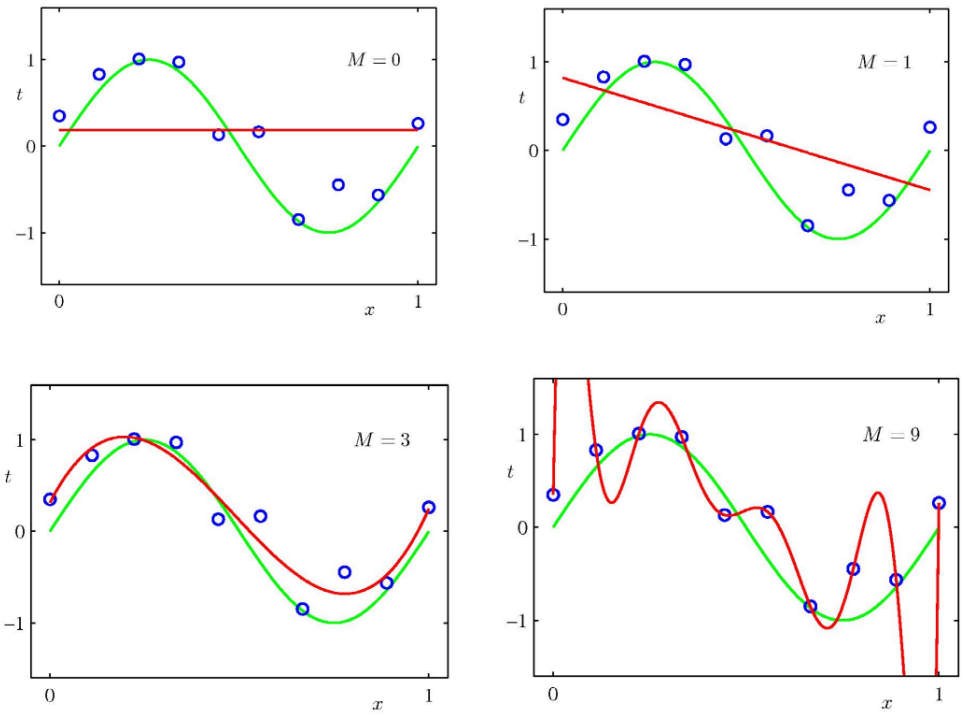


Figure 8: A large M can cause overfitting with the ML approach (from [1]).

- To solve this problem, we can use model selection: If, for instance, M=3 is selected in Fig. 8, overfitting is avoided. Model selection requires validation (or, more generally, cross validation), which is illustrated in Fig. 9.

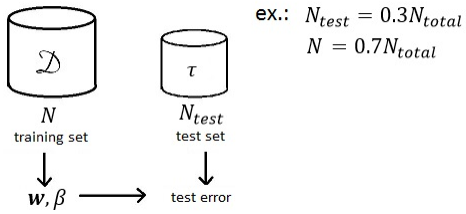


Figure 9: (Hold-out validation: A hold-out, or test, data set of N_{test} samples is used to estimate the generalization error (11).

- In validation, the hold-out, or test, data set allows us to estimate the generalization error by means of the test error:

$$L_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{n=1}^{Ntest} (t_n - y(x_n, \mathbf{w}))^2 \tag{12}$$

where (x_n, t_n) are the samples in the test set.

- The model order M should be selected so as to minimize the test error, which is a proxy for the generalization error, see Fig. 10.

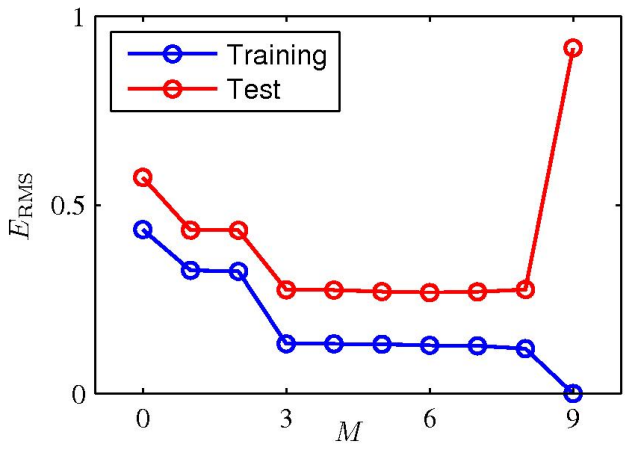


Figure 10: A larger M always reduces the training error given that the model has more degrees of freedom to fit the data points (memorizing). However, a larger M may lead to a poor generalization error, as measured by the test error (from [1]).

- The overfitting problem depends on the size of the data set N, see Fig. 11.
- More generally, one can use cross-validation:
  - (From Wikipedia) In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k-1$ subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling (see below) is that all observations are used for both training and validation, and each observation is used for validation exactly once. 10-fold cross-validation is commonly used. When $k = N$ (the number of observations), the k-fold cross-validation is known as leave-one-out cross-validation.

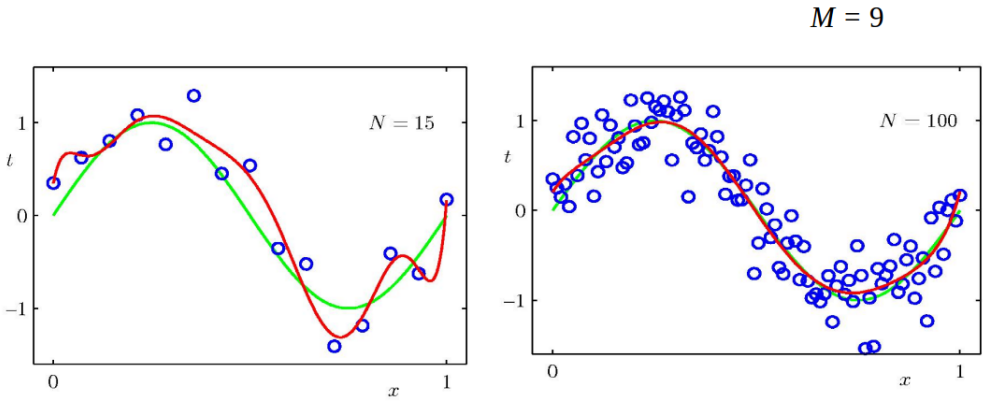


Figure 11: When N is large enough as compared to M, the training error becomes close to the generalization error and the overfitting problem is less pronounced (from [1]).

Maximum A Posteriori (MAP) Approach

- As seen, the ML estimate suffers from overfitting. Experiments reveal that overfitting is typically manifested by large values of $\mathbf{w}$ (see Fig. 12).

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Figure 12: Large values of $\mathbf{w}$ indicate that overfitting problem is taking place. This a priori information is exploited by the MAP approach (from [1]).

- Inductive bias: use the a priori information that $\mathbf{w}$ tends to be not too large in the absence of overfitting.
- A priori distribution:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|0, \underbrace{\alpha^{-1}}_{\text{hyperparameter}} \mathbf{I}) = \prod_{m=0}^{M} \mathcal{N}(w_m|0, \alpha^{-1}). \tag{13}$$

- The a priori distribution encodes our degree of uncertainty or belief on $\mathbf{w}$.
- Rather than maximizing the probability of the data $p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} p(t_n|x_n, \mathbf{w}, \beta)$ as in ML, we maximize the joint probability of the data and of $\mathbf{w}$, namely

$$p(\mathbf{t}, \mathbf{w}|\mathbf{x}, \beta) = p(\mathbf{w}) \prod_{n=1}^{N} p(t_n|x_n, \mathbf{w}, \beta) \tag{14}$$

- Note that a prior probability can also be assumed for β, but in this example we leave β as a parameter with no prior distribution.
- This yields the Maximum a Posteriori (MAP) estimates:

$$\underset{\mathbf{w},\beta}{\text{minimize}} \; -\frac{1}{N}\sum_{n=1}^{N} \ln p(t_n|x_n, \mathbf{w}, \beta) - \frac{1}{N} \ln p(\mathbf{w})$$

$$\Rightarrow \underset{\mathbf{w},\beta}{\text{minimize}} \; \beta \underbrace{\frac{1}{N}\sum_{n=1}^{N}(y(x_n, \mathbf{w}) - t_n)^2}_{L(\mathbf{w})} - \ln\frac{\beta}{2\pi} + \frac{\alpha}{N}\|\mathbf{w}\|^2 \tag{15}$$

$$\Rightarrow \begin{cases} \mathbf{w}_{MAP} = \arg\min_{\mathbf{w}} L(\mathbf{w}) + \underbrace{\frac{\lambda}{N}\|\mathbf{w}^2\|}_{\text{quadratic regularization}} \\ \beta_{MAP} = \beta_{ML} \end{cases} \tag{16}$$

- The solution of the regularized least squares problem (or ridge regression) can be found in closed form (see Problem 3).
- The modified error yields smaller values of $\mathbf{w}$ (see Fig. 13), and is hence referred to in statistics as shrinkage.
- Predictive distribution for new observations (x, t):

$$p(t|x, \mathbf{w}_{MAP}, \beta_{MAP}) = \mathcal{N}(t|y(x, \mathbf{w}_{MAP}), \beta_{MAP}^{-1}) \tag{17}$$

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Figure 13: Impact of the regularization parameter λ on the MAP estimates for the model parameter $\mathbf{w}$.

- But how to choose the hyperparameter λ (or α) ?

- Increasing λ reduces overfitting but may create a large bias. Increasing λ is hence akin to reducing M in the ML approach, see Fig. 14.

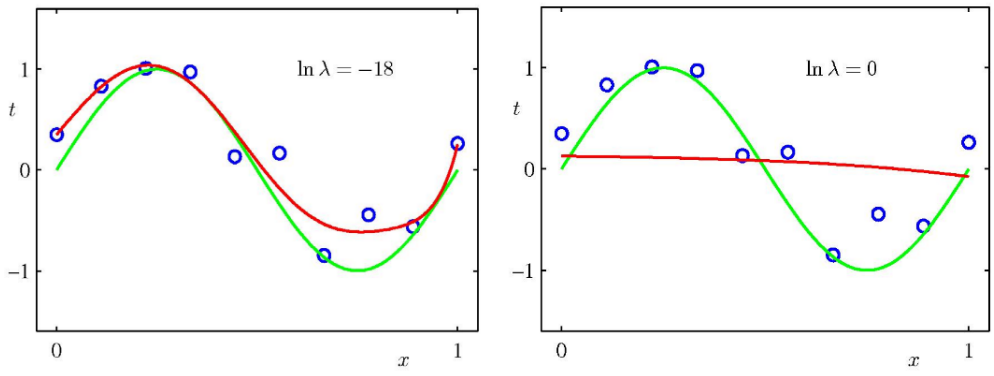


Figure 14: Adverse affects of too much regularization: increasing λ has a similar effect as reducing M in the ML approach (from [1]).

- Choosing λ (or α) hence generally requires validation, see Fig. 15, in a manner similar to ML.

- However, if one has good reasons (e.g., from domain knowledge) to choose a given λ, model selection can be directly performed by selecting the model order M that maximizes $\sum_{n=1}^{N} \ln p(t_n|x_n, \mathbf{w}_{MAP}, \beta_{MAP}) + \ln p(\mathbf{w}_{MAP})$. In fact, the prior term $\ln p(\mathbf{w}_{MAP})$ generally penalizes large orders M, since with large M, due to overfitting, $\ln p(\mathbf{w}_{MAP})$ will tend to be large. The criterion $\sum_{n=1}^{N} \ln p(t_n|x_n, \mathbf{w}_{MAP}, \beta_{MAP}) + \ln p(\mathbf{w}_{MAP})$ can be seen as an example of the principle of Minimum Description Length (MDL), as further discussed below [8].

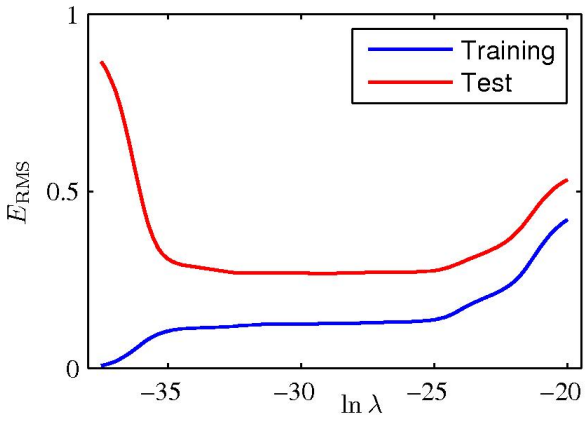


Figure 15: A lower λ always reduces the training error given that the objective function of MAP tends to the ML criterion $L(\mathbf{w})$ of minimizing the training error. However, a small λ may lead to a poor generalization error, as measured by the test error (from [1]).

Remark Validation for both ML and MAP can be in principle avoided via Structural Risk Minimization (SRM), which minimizes the analytical bound on the generalization error (see Chapter 4). The key idea is to add a penalty term to the empirical risk that accounts for the model size and for the number of observations. The Bayesian approach, to be discussed below, can also theoretically operate without validation, although, as MP, it depends on the choice of a specific prior.

4 Bayesian Approach

- The frequentist approach adopted so far follows the steps:

  1. Obtain point estimates of the model parameters $(\mathbf{w}, \beta)$ via ML or MAP;
  2. Compute the predictive distribution $p(t|x, \mathbf{w}, \beta)$ as an approximation of the "true" distribution.

- The Bayesian approach instead treats parameters, data set labels $\mathbf{t}$ and new label t as as jointly distributed random variables and use the rules of probability to determine the predictive probability $p(t|\mathcal{D})$

- Note that we don't need to postulate the existence of a "true" distribution.

- Assume β known for simplicity in the following.

- Joint distribution assumed in Bayesian approach:

$$p(\mathbf{t}, \mathbf{w}, t|\mathbf{x}, x) = \underbrace{p(\mathbf{w})}_{\text{a priori distribution}} \underbrace{p(\mathbf{t}|\mathbf{w}, \mathbf{x})}_{\text{likelihood}} \underbrace{p(t|\mathbf{w}, x)}_{\text{distribution of new data}} \tag{18}$$

- It is often useful to drop the dependence on the domain points x and x since they are fixed parameters to write only the joint distribution of the random variables in the model as

$$p(\mathbf{t}, \mathbf{w}, t) = \underbrace{p(\mathbf{w})}_{\text{a priori distribution}} \underbrace{p(\mathbf{t}|\mathbf{w})}_{\text{likelihood}} \underbrace{p(t|\mathbf{w})}_{\text{distribution of new data}} \tag{19}$$

- A graphical representation in terms of a Bayesian network (to be discussed) can be found in Fig. 16.

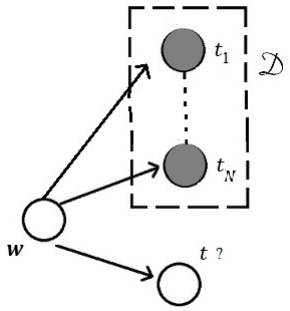


Figure 16: Bayesian network describing the joint distribution (19) used in the Bayesian approach.

- Apply rules of probability to calculate the predictive distribution $p(t|\mathbf{t})$ from the joint distribution (19)

$$\begin{aligned} p(t|\mathbf{t}) &= \frac{p(\mathbf{t},t)}{p(\mathbf{t})} \\ &= \frac{\int p(\mathbf{w})p(\mathbf{t}|\mathbf{w})p(t|\mathbf{w})d\mathbf{w}}{p(\mathbf{t})} \\ &= \int \frac{p(\mathbf{w})p(\mathbf{t}|\mathbf{w})}{p(\mathbf{t})} p(t|\mathbf{w})d\mathbf{w} \\ &= \int p(\mathbf{w}|\mathbf{t})p(t|\mathbf{w})d\mathbf{w}. \end{aligned} \tag{20}$$

- Putting back the dependence on the domain variables, we get (recall that $\mathcal{D} = \{(\mathbf{x}, \mathbf{t})\}$)

$$p(t|x,\mathcal{D}) = \int \underbrace{p(\mathbf{w}|\mathcal{D})}_{\text{posterior distribution of } \mathbf{w}} \underbrace{p(t|x,\mathbf{w})}_{\mathcal{N}(t|y(x,\mathbf{w}),\beta^{-1})} d\mathbf{w} \tag{21}$$

- In summary, the Bayesian approach operates as follows:

  1. calculate the posterior distribution of the parameters, namely $p(\mathbf{w}|\mathcal{D})$;
  2. calculate the predictive distribution using (21).

- See Fig. 17 for an illustration of how the posterior probability depends on the size of the data set.

- For the problem at hand, as shown on p. 31 in [1], the predictive distribution obtained under the Bayesian approach is

$$p(t|x,\mathcal{D}) = \mathcal{N}(t|y(x,\mathbf{w}_{MAP}), s^2(x)), \tag{22}$$

  where the expression for the variance $s^2(x)$ can be found in [1, p. 31]. Interestingly, the variance of the predictive distribution depends on x: values of x closer to the existing points in the training sets generally exhibit a smaller variance (see Fig. 7).

- The Bayesian approach in principle allows model selection to be performed without validation (see Appendix B). In fact, when averaging over the posterior distribution, one gets "vague" predictions, rather than fit to the training data, due to the fact that large portions of the parameter space have significant posterior probability.

- Optimization over the hyperparameters can be carried out using Bayesian optimization (although this requires also setting hyperparameters!) [7].

- In a sense, the Bayesian approach unifies the problems of inference model, selection and predictions into the Bayesian inference problem of computing posterior probabilities.

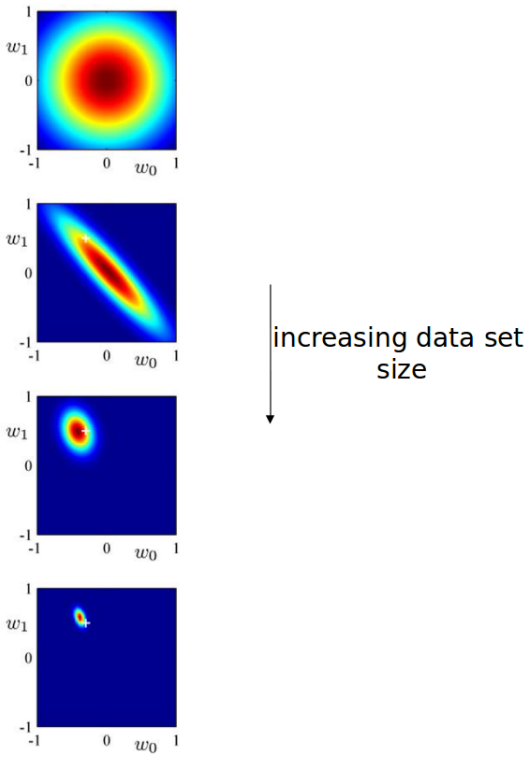


Figure 17: Illustration of the variation of the posterior distribution of $\mathbf{w}$ as a function of the size N of the data set: for $N = 0$, the posterior equal the prior distribution, while for large N the posterior tends to be centered around the ML estimate (from [1]).

5 Minimum Description Length (MDL)

- A conceptually different approach from both the frequentist viewpoint, which is based on the idea that there exists a "true" model, and the Bayesian model, which allows the data to be explained by more than one models as weighted by the posterior distribution, is MDL.
- MDL is based on the following idea: Choose the model that allows one to compress the data the most. The MDL framework hence does not assume neither the existence of a "true" model nor prior distributions on the models. It instead relies on the information-theoretic concept of universal compression.
- As proved via information-theoretic tools, given a probabilistic model $p(x)$, it is possible to design a (decodable) compression scheme for which the value x is described by (approximately) $-\log_2 p(x)$ bits. Therefore, we can interpret, e.g., the quantity $-\sum_{n=1}^{N} \log_2 p(t_n|x_n, \mathbf{w}_{MAP}, \beta) - \log_2 p(\mathbf{w}_{MAP})$ seen above for MAP as the number of bits needed to compress the model $\mathbf{w}_{MAP}$ and the data $\mathbf{t}$ (assuming that β is known) using the model learned by MAP. This shows that MAP can be interpreted as a particular instance of MDL (based on so-called two-part codes).
- Details can be found in [6]. Limited additional discussion will be provided in the rest of the notes about MDL.

6 From Inference to Optimal Decision

- From the discussed inference (or model learning) step, we obtain a predictive distribution $p(t|x)$ for either frequentist or Bayesian approaches.
- We can then take $p(t|x)$ as a "soft" prediction of t given x (see Fig. 7).
- However, we often need a single value of $t = y(x)$. In fact, in the regression problem studied above, we have taken as prediction the mean of the predictive distribution, i.e.,

$$t = y(x, \mathbf{w}) = E_{p(t|x)}[t|x], \tag{23}$$

where $\mathbf{w}$ is either the ML or MAP estimate. But why was this choice made?

- For instance, suppose that, for a certain problem, we get $p(t|x) = 0.5\mathcal{N}(t|3x, 1) + 0.5\mathcal{N}(t| - 3x, 1)$. What should be the predictive function $t = y(x)$.
- The answer is that it depends on the adopted loss function. In particular, once a loss function $\ell(t, y)$ has been selected, one should solve for the optimal decision

$$y(x) \leftarrow \min_y \underbrace{\int \ell(t, y)p(t|x)dt}_{\text{average loss (given } x)} \tag{24}$$

- Examples of choices for the loss function include the ℓ_q loss $\ell(t, y) = (t - y)^q$ with $q \geq 0$ (see Fig. 18). For instance, with $q = 2$, we get the quadratic loss considered above.
- The optimal decision in (24) is given as
  - $q = 2 \Rightarrow$ mean
  - $q = 1 \Rightarrow$median
  - $q = 0 \Rightarrow$mode (maximum)
- To prove that with ℓ_q the optimal decision is the mean of the predictive distribution, it is sufficient to set the derivative in (24) to zero:

$$\frac{dE[\ell|x]}{dy} = 2\int (t - y)p(t|x)dt = 0$$

$$\Rightarrow y(x) = \int tp(t|x)dt = E[t|x]. \tag{25}$$

- See Problem 4 for an example.

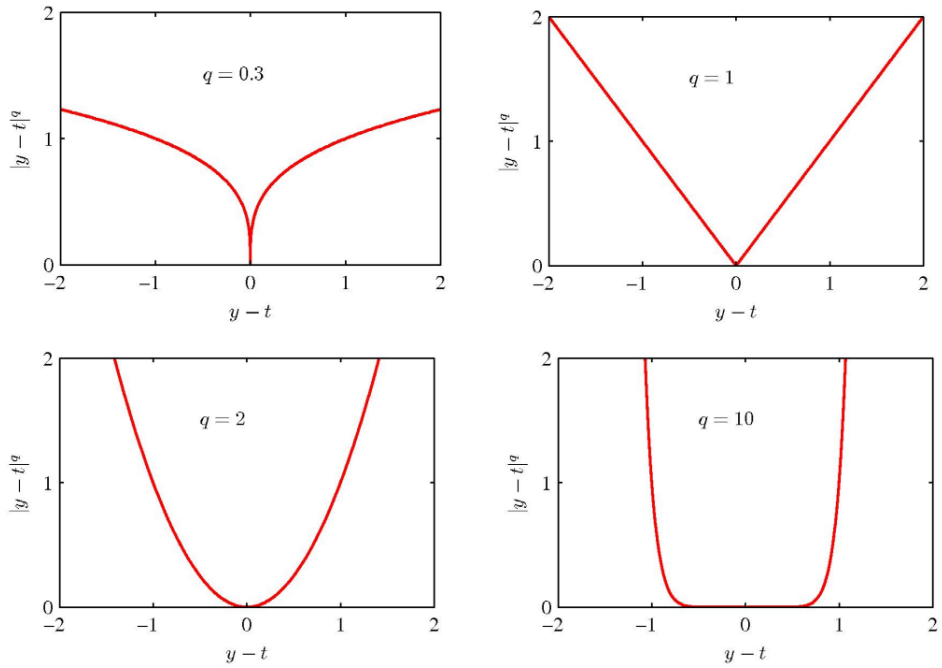


Figure 18: Illustration of ℓ_q loss functions (from [1]).

- Finally, we recall that the generalization error is defined as

$$\underbrace{E_{p_0(x,t)}}_{\text{unknown}}[\ell(t, y(x))]. \tag{26}$$

7 The Learning Cycle

- To summarize the discussion above, learning generally consists of the following steps:
  - Choice of a model class (see below for classification)
  - Learning (under a probabilistic model):
    - Inference: Learn the probabilistic model $p(t|x)$;
    - Optimal decision (prediction): Use (24)
    - Validation or cross-validation: Estimate the generalization error (26)
    - Go back to inference by adapting model order or hyperparameters if validation or cross-validation not satisfactory
  - Go back to choice of a model class if learning not satisfactory

8 Discriminative vs. Generative Models

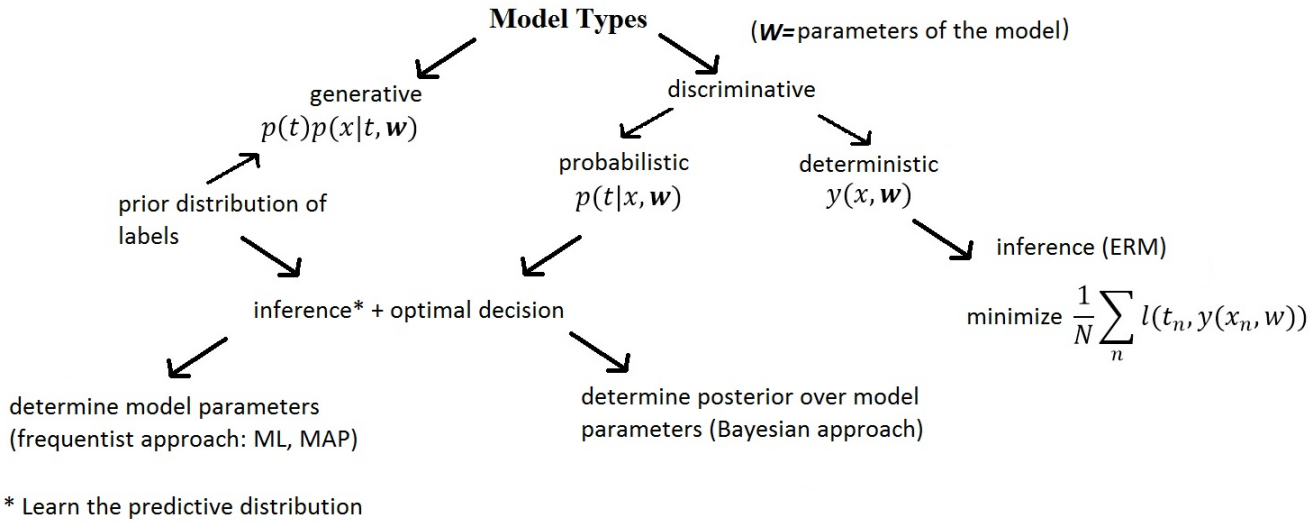


Figure 19: Summary of machine learning models.

- In the example above, we have used a probabilistic discriminative model. A taxonomy of machine learning models for supervised learning can be seen in Fig. 19.
- As opposed to discriminative models, generative models require making stronger assumptions, modeling also the data distribution $p(x)$: if model is incorrect, this may lead to bias problems. However, there are potential advantages:
  - Generative models allow to deal with missing data, i.e., with cases in which some entries of X may be missing in some data points.
  - Generative models enable some supervised learning, where some of the labels t are avoidable.

$$\mathcal{D}: \begin{cases} \{(x_n, t_n)\}_{n=1}^{N'} \\ \{x_n\}_{n=1}^{N''} \end{cases} \quad \leftarrow \text{useful to learn } p(x): \text{ yields information on } p(t|x) \tag{27}$$

  - See p. 268 of for additional discussion.
- A note on terminology: In some literature, particularly concerning probabilistic graphical models (see [4]), the inference step as defined here (that is, the identification of a probabilistic model with desired features) is defined as "learning", while the optimal prediction/decision step (that is, the decision, possibly Bayesian, one one set of variables given another) is known as "inference".

9 An Information Theoretic Interpretation of ML

- Here, we revisit ML and point out an interesting connection to an information-theoretic metric, namely the KL divergence, which is extensively used in machine learning.
- ML solves

$$\underset{\mathbf{w},\beta}{\text{minimize}} \underbrace{-\frac{1}{N}\sum_{n=1}^{N} \ln p(t_n|x_n,\mathbf{w},\beta)}_{\text{log-loss}} \tag{28}$$

- When N is large, we have by the law of large numbers:

$$\begin{aligned} &\underset{\mathbf{w},\beta}{\text{minimize}}\ E_{p_0(t,x)}\left[\ln \frac{1}{p(t|x,\mathbf{w},\beta)}\right] \\ \iff &\underset{\mathbf{w},\beta}{\text{minimize}} \int p_0(x)\left(\int p_0(t|x)\ln\frac{1}{p(t|x,\mathbf{w},\beta)}dt\right)dx \\ \iff &\underset{\mathbf{w},\beta}{\text{minimize}} \int p_0(x)\underbrace{\left(\int p_0(t|x)\ln\frac{p_0(t|x)}{p(t|x,\mathbf{w},\beta)}dt\right)}_{KL(p_0(t|x)\|p(t|x,\mathbf{w},\beta))}dx. \end{aligned} \tag{29}$$

- The relative entropy or Kullback-Leibler (KL) divergence

$$KL(p\|q) = \int p(x)\ln\frac{p(x)}{q(x)}dx \tag{30}$$

  measures the "distance" between two distributions p and q.
- Ex.: With $p(x)=\mathcal{N}(x|\mu_1,\sigma_1^2)$, $q(x)=\mathcal{N}(x|\mu_2,\sigma_2^2)$, we have

$$KL(p\|q) = \frac{1}{2}\left(\frac{\sigma_1^2}{\sigma_2^2}+\frac{(\mu_2-\mu_1)^2}{\sigma_2^2}-1+\ln\frac{\sigma_2^2}{\sigma_1^2}\right) \tag{31}$$

  With $\sigma_1^2=\sigma_2^2$

$$KL(p\|q) = \frac{(\mu_2-\mu_1)^2}{\sigma^2} \tag{32}$$

- It measures the error in the test $X\sim p(x)$ vs. $X\sim q(x)$ (see Appendix A for a more precise statement).
- It measures the overhead in describing $X\sim p(x)$ if the compression algorithm is based on $q(x)$.
- Some properties:
  - $KL(p\|q)\neq KL(q\|p)$
  - $KL(p\|q)\geq 0$ with equality iff $p(x)=q(x)$
  - See other properties on pp. 55-58 of [1].
- From the interpretation given above of the KL divergence, ML tries to make the model distribution $p(t|x,\mathbf{w},\beta)\approx p_0(t|x)$ when N is large. In particular, when N is large, the ML consistently estimates the true distribution if the latter is part of the model class.
- For finite N, ML tries to minimize the empirical approximation (28).
- Remarks:
  - In machine learning, the notation $KL(p\|q)$ is used even when q is unnormalized (i.e., $\int q(x)dx\neq 1$ but $q(x)\geq 0$).
  - We can write

$$KL(p\|q) = \underbrace{-\int p(x)\ln q(x)dx}_{\text{cross-entropy } H(p,q)} \underbrace{-\left(-\int p(x)\ln p(x)dx\right)}_{\text{entropy } H(p)} \tag{33}$$

  - A related learning criterion that is based on information theoretic measures is that of the information bottleneck.

10 Appendix A: An Interpretation of the KL Divergence via Hypothesis Testing

- Consider the test between the hypotheses $\mathcal{H}_0 : X \sim p(x)$ and $\mathcal{H}_1 : X \sim q(x)$.
- Assume that $\Pr(\mathcal{H}_0) = \Pr(\mathcal{H}_1) = \frac{1}{2}$.
- A test is characterized by an "acceptance" region $\mathcal{A} \subseteq \mathcal{X}$ such that $\mathcal{X} \in \mathcal{A} \Rightarrow \mathcal{H}_0$; $\mathcal{X} \in \mathcal{A}^c \Rightarrow \mathcal{H}_1$.
- The probability of error is given as

$$P_{error} = \frac{1}{2}p(\mathcal{A}^c) + \frac{1}{2}q(\mathcal{A})$$

$$= \frac{1}{2} - \frac{1}{2}(p(\mathcal{A}) - q(\mathcal{A})).$$

- The minimum probability of error over the tests is given as

$$\Rightarrow \min_{\{\text{test}\}} P_{error} = \frac{1}{2}(1 - \underbrace{\max_{\mathcal{A}} (p(\mathcal{A}) - q(\mathcal{A}))}_{=\frac{1}{2}\sum_x |p(x)-q(x)| = \|p(x)-q(x)\|_{TV} \text{ variational distance}})$$

$$= \frac{1}{2}(1 - \underbrace{\|p(x) - q(x)\|_{TV}}_{\leq \frac{1}{\sqrt{2}}\sqrt{KL(p\|q)} \text{ Pinsker inequality}})$$

$$\geq \frac{1}{2}(1 - \frac{1}{\sqrt{2}}\sqrt{KL(p\|q)}). \tag{34}$$

11 Appendix B: Model Selection in the Bayesian Approach

- Under the Bayesian approach, model selection can be performed by choosing the model that maximizes the *marginal likelihood*

$$p(\mathbf{t}|\mathbf{x}) = \int p(\mathbf{w}) \prod_{n=1}^{N} p(t_n|x_n, \mathbf{w}) d\mathbf{w}, \tag{35}$$

  which is computed on the training set. Note that the above holds for any probabilistic model with parameter vector $\mathbf{w}$.

- Removing the dependence on $\mathbf{x}$ to simplify the notation, this can be evaluated as

$$p(\mathbf{t}) = \prod_{n=1}^{N} p(t_n|t_1, ..., t_{n-1}), \tag{36}$$

  where

$$p(t_n|t_1, ..., t_{n-1}) = \int p(\mathbf{w}|\mathcal{D}_{n-1}) p(t_n|x_n, \mathbf{w}) d\mathbf{w}, \tag{37}$$

  with $\mathcal{D}_{n-1} = \{(x_i, t_i)\}_{i=1}^{n-1}$. Note that (37) follows in a similar way as (21) and can hence be computed for the example at hand using (22).

- Looking at (37), we observe that the approach can be seen as a form of Bayesian cross-validation in which the data set $\mathcal{D}_{n-1}$ is used to train a predictor for the label t_n.
- It can be seen that the marginal likelihood is generally not an increasing function of M and that it consistently estimates the correct model order [4].

12 Problems

1. Consider the regression problem studied above with the discriminative model

$$t = \sum_{m=0}^{M} w_m x^m + z, \text{ with } z \sim \mathcal{N}(z|0, \beta^{-1}).$$

Given a training set $\mathcal{D} = \{(x_n, t_n)\}_{n=1}^N$, find an analytical expression for the ML estimate $\mathbf{w}_{ML}$ as a function of the $N \times 1$ column vector $\mathbf{t} = [t_1, ..., t_N]^T$ (the superscript T represents matrix transpose) and of the $N \times M$ matrix

$$\mathbf{X} = [\mathbf{1}\ \mathbf{x}\ \mathbf{x}^2 ... \mathbf{x}^M],$$

where $\mathbf{1}$ is an $N \times 1$ column vector of all ones, $\mathbf{x} = [x_1, ..., x_N]^T$ and $\mathbf{x}^m = \mathbf{x} = [x_1^m, ..., x_N^m]^T$. To do this, write the optimization problem that yields $\mathbf{w}_{ML}$ as the minimization of the expression $||\mathbf{t} - \mathbf{X}\mathbf{w}||^2$, where $||\mathbf{a}||^2 = \sum_i a_i^2$ is the (quadratic) norm of a vector $\mathbf{a} = [a_1, ..., a_N]^T$. (Hint: Show that $||\mathbf{t} - \mathbf{X}\mathbf{w}||^2 = \sum_{n=1}^N (t_n - \sum_{m=0}^M w_m x_n^m)^2$. Then, recognize that this is a least square problem and look up the solution in a textbook.)

2. Load in MATLAB the data that you can download from here, which includes training variables $\mathbf{x}$ and $\mathbf{t}$. You can do this by using the line `load hw1data` (the folder in which the data is stored should be in the MATLAB search path) or by using the button "Import data". Note that we have $N = 20$ training points.

a. Plot $\mathbf{t}$ versus $\mathbf{x}$. You can do this by using: `plot(t,x,'o')`.

b. For $M = 5$, obtain $\mathbf{w}_{ML}$ using the equation derived at point 1 (which should be: `w=pinv(X)*t` or equivalently `w=inv(X'*X+lambda*eye(M+1))*X'*t`). Plot the resulting function $y(x, \mathbf{w}) = \sum_{m=0}^M w_m x^m$ with $\mathbf{w} = \mathbf{w}_{ML}$ on top of the training set (Hint: Use `hold on`).

c. Repeat the point above for M between 1 and 6. For each value of M compute the training root mean squared error $\sqrt{L(\mathbf{w})} = 1/\sqrt{N}||\mathbf{t} - \mathbf{X}\mathbf{w}||$ and plot it as a function of M in a second figure. Why does the error decrease as a function of M?

d. As we have seen, if M is large, however, overfitting may occur. To estimate the amount of overfitting and select a proper value for M, we now evaluate the error on the hold-out or test set variables `xtest` and `ttest`, which you can find in your workspace. This can be done by computing the root mean squared loss obtained with the ML estimate on this set. Plot the test error on the same figure. Which value of M would you choose?

e. Estimate the precision β and plot at point b above $\sum_{m=0}^M w_m x^m \pm \beta^{-1}$.

f. Plot the training and test errors versus N with $M = 7$ by using the training variable vectors `x1` and `t1` in the workspace. Can you interpret the results?

3. Continuing the problem above, we would like to compare the performance of ML and MAP. To this end, load the same data set and follow the steps below.

a. Plot $\mathbf{t}$ versus $\mathbf{x}$. You can do this by using: `plot(t,x,'o')`.

b. For $M = 6$, consider ten possible values for λ, namely $\lambda = \exp(v)$ with $v = \ln \lambda$ taking one of ten equally spaced values between -30 and 10 (You can generate such vector as `linspace(-30,10,Nlambda)`). For each value of λ, compute the MAP estimate of the model parameters, which is given as

$$\mathbf{w}_{MAP} = (\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t},$$

with the definitions given in the previous assignment. Note that, when $\lambda = 0$, we obtain the ML solution (why?). Plot the resulting functions $y(x, \mathbf{w}) = \sum_{m=0}^M w_m x^m$ with $\mathbf{w} = \mathbf{w}_{MAP}$ on top of the training set.

c. For each value of λ compute the training root mean squared error $\sqrt{L(\mathbf{w})} = 1/\sqrt{N}||\mathbf{t} - \mathbf{X}\mathbf{w}||$ and plot it as a function of $\ln \lambda$ in a second figure. Why does the error decrease as λ decreases?

d. If λ is small, however, overfitting may occur. To estimate the amount of overfitting and select a proper value for λ, we now evaluate the error on the hold-out or test set variables `xtest` and `ttest`, which you can find in your workspace. This can be done by computing the root mean squared error obtained with the MAP estimate on this set. Plot the test error on the same figure. Which value of λ would you choose?

4. Assume that, as a result of inference (using ML, MAP or a Bayesian approach), you obtain the following predictive distribution for a given value of x

$$p(t) = 0.5\mathcal{N}(t|3, 1) + 0.5\mathcal{N}(t|-3, 1).$$

Consider the family of loss functions ℓ_q with $\ell(t - y) = |t - y|^q$.

a. What is the optimal prediction y under the quadradic, or ℓ_2, loss?
b. What is the optimal prediction y under the ℓ_1 loss?
c. What is the optimal prediction y under the ℓ_0 loss?

5. [5]You are asked to design a learning algorithm to predict whether patients are going to suffer a heart attack. Relevant patient features the algorithm has access to include blood pressure (BP), body-mass index (BMI), age (A), level of physical activity (P), and income (I). You have to choose between two classification algorithms – the first picks only the features BP and BMI, while the other considers all the features. Explain the pros and cons of each choice and how the number of available labeled training samples will affect your choice.

References

[1] C. Bishop, *Pattern recognition and Machine Learning*, Springer, 2006.

[2] K. P. Murphy, *Machine learning: a probabilistic perspective*, MIT press, 2012.

[3] I. Selesnick, "Least Squares with Examples in Signal Processing," Lecture Notes (link).

[4] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*, MIT press, 2009.

[5] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge University Press, 2014.

[6] P. D. Grünwald, *The minimum description length principle*, MIT press, 2007.

[7] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE,* vol. 104, no. 1, pp. 148-175, Jan. 2016.

[8] A. Honkela and H. Valpola, "Variational learning and bits-back coding: an information-theoretic view to Bayesian learning," *IEEE Transactions on Neural Networks*, vol. 15, no. 4, pp. 800-810, Jul. 2004.

[9] C. O'Neil, *Weapons of math destruction: How big data increases inequality and threatens democracy.* Crown Publishing Group, 2016.

2. Learning and Probabilistic Models

1 Introduction

- As discussed, probabilistic models (frequentist or Bayesian) play a key role in machine learning.
- In this section, we review some key results on typical probabilistic models and corresponding ML/MAP and posterior probability calculation.
- We specifically cover a general class of probabilistic models known as the linear exponential family and provide general results concerning learning, both frequentist and Bayesian, that enable the adoption of a large set of probabilistic models.
- We start by considering unsupervised learning scenarios in which the goal is the *inference* task of learning the distribution of the observations

$$x_n \underset{i.i.d.}{\sim} p(x),\ n = 1, \ldots, N \tag{1}$$

- We will see later how to define models suitable for supervised learning via Generalized Linear Models (GLMs).

2 The (Linear) Exponential Family

- The linear exponential family contains probabilistic models of the form

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\eta}) &= \frac{1}{Z(\boldsymbol{\eta})} \exp\left(\sum_k \eta_k u_k(\mathbf{x})\right) m(\mathbf{x}) \\ &= \frac{1}{Z(\boldsymbol{\eta})} \exp\left(\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})\right) m(\mathbf{x}), \end{aligned} \tag{2}$$

  where

  - $\mathbf{x}$ discrete or continuous
  - $\boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \vdots \end{bmatrix}$ natural parameters (defined on a convex set)
  - $\mathbf{u}(\mathbf{x}) = \begin{bmatrix} \mathrm{u}_1(\mathbf{x}) \\ \vdots \end{bmatrix}$ sufficient statistics
  - $Z(\boldsymbol{\eta}) = \int \exp\left(\boldsymbol{\eta}^{\mathrm{T}} \mathbf{u}(\mathbf{x})\right) m(\mathbf{x})\, \mathrm{d}\mathbf{x}$ partition function
  - $m(\mathbf{x})$ base measure (independent of $\boldsymbol{\eta}$)

- Ex: Gaussian distribution

$$x \sim \mathcal{N}\left(x|\mu, \sigma^2\right) = \frac{1}{\left(2\pi\sigma^2\right)^{1/2}} \exp\left(-\frac{x^2}{2\sigma^2} + \frac{\mu}{\sigma^2}x - \frac{\mu^2}{2\sigma^2}\right) \tag{3}$$

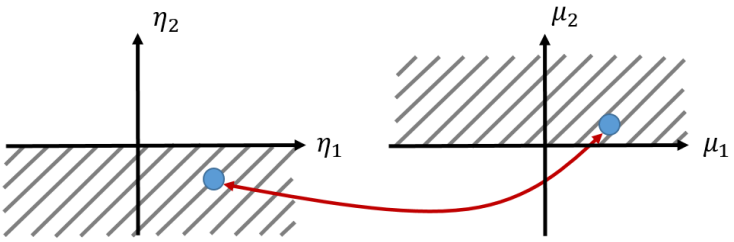


Figure 1: One-to-one mapping between natural parameters and mean parameters.

$$\boldsymbol{\eta} = \begin{bmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{bmatrix}$$
$$\mathbf{u}(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$
$$Z(\boldsymbol{\eta}) = \left(\frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left(-\frac{1}{2\sigma^2}\mu^2 \right) \right)^{-1}$$
$$m(x) = 1$$

- There is a one-to-one mapping between natural parameters $\boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{bmatrix} \in (\mathbb{R},\ \mathbb{R}^-)$ and mean parameters

$$\begin{bmatrix} \mathrm{E}[u_1(x)] = \mathrm{E}[x] = \mu = -\eta_1/2\eta_2 \\ \mathrm{E}[u_2(x)] = \mathrm{E}[x^2] = \sigma^2 + \mu^2 = -1/2\eta_2 + (\eta_1/2\eta_2)^2 \end{bmatrix},$$

  and hence we can learn either $\boldsymbol{\eta}$ or $\boldsymbol{\mu}$ (see Fig. 1).

- It is useful to write (2) as

$$\ln p\left(\mathbf{x}|\boldsymbol{\eta}\right) = \boldsymbol{\eta}^T \mathbf{u}\left(\mathbf{x}\right) - A\left(\boldsymbol{\eta}\right) + \ln m\left(\mathbf{x}\right) \tag{4}$$

  where $A\left(\boldsymbol{\eta}\right) = \ln Z\left(\boldsymbol{\eta}\right)$ is the log-partition function, which can be proved to be convex ($\cup$) in $\boldsymbol{\eta}$.

- It follows that

$$\ln p\left(\mathbf{x}|\boldsymbol{\eta}\right) = \text{linear}\left(\boldsymbol{\eta}\right) + \text{concave}\left(\boldsymbol{\eta}\right) + \text{constant}, \tag{5}$$

  and hence $\ln p\left(\mathbf{x}|\boldsymbol{\eta}\right)$ is a concave ($\cap$) function of $\boldsymbol{\eta}$.

- Concave functions are easy to optimize, and hence learning via ML is easy.

- Ex: From (3), for the Gaussian distribution, we have

$$\ln \mathcal{N}\left(x|\mu,\sigma^2\right) = -\frac{x^2}{2\sigma^2} + \frac{\mu}{\sigma^2}x - \frac{\mu^2}{2\sigma^2} - \frac{1}{2}\ln\left(2\pi\sigma^2\right) \tag{6}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \mathrm{E}[u_1(x)] = \mu \\ \mathrm{E}[u_2(x)] = \sigma^2 + \mu^2 \end{bmatrix}$$
$$A(\eta) = \frac{\mu^2}{2\sigma^2} + \frac{1}{2}\ln(2\pi\sigma^2)$$

- The one-to-one correspondence between $\boldsymbol{\mu}$ and $\boldsymbol{\eta}$ stems from the following property of the exponential family:
  - The exponential family solves the maximum entropy problem (see Appendix for a brief introduction to entropy)

* For discrete random variables

$$\underset{p(\mathbf{x})}{\text{maximize}} \;\; H(\mathbf{x}) = -\sum_{\mathbf{x}} p(\mathbf{x}) \ln p(\mathbf{x}) \quad \text{entropy}$$

$$\text{s.t.} \;\; \mathrm{E}[u_k(\mathbf{x})] = \mu_k \quad \text{for all } k$$

* For continuous random variables

$$\underset{p(\mathbf{x})}{\text{maximize}} \;\; h(\mathbf{x}) = -\int p(\mathbf{x}) \ln p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \quad \text{differential entropy}$$

$$\text{s.t.} \;\; \mathrm{E}[u_k(\mathbf{x})] = \mu_k \quad \text{for all } k$$

– The natural parameters can be interpreted as Lagrange multipliers (see [2, Ch. 6-7]).

- Ex: Bernoulli distribution

$$x \sim \mathrm{Bern}(x|\mu) = \mu^x (1-\mu)^{1-x} \quad \text{with } \mu = \mathrm{E}[x] = \Pr(x=1) \tag{7}$$

– Taking the logarithm of (7), we have

$$\ln \mathrm{Bern}(x|\mu) = \ln\left(\frac{\mu}{1-\mu}\right) x + \ln(1-\mu) \tag{8}$$

and hence

$$\begin{aligned}
&u(x) = x \\
&\eta = \ln\left(\frac{\mu}{1-\mu}\right) \in \mathbb{R} \\
&\mu = \frac{1}{1+e^{-\eta}} = \sigma(\eta) \quad \text{logistic sigmoid function} \\
&A(\eta) = -\ln(1-\mu) = \ln(1+e^{\eta}) \\
&m(x) = 1
\end{aligned}$$

- Ex: Categorical or Multinoulli distribution

$$x \sim \mathrm{Cat}(x|\boldsymbol{\mu}) \tag{9}$$

where $\mu_k = \Pr[x=k]$, $k = 1, \ldots M-1$ with $\sum_{k=0}^{M-1} \mu_k = 1$, and

$$\mathrm{Cat}(x|\boldsymbol{\mu}) = \prod_{k=1}^{M-1} \mu_k^{1(x=k)} \cdot \mu_0^{1-\sum_{k=1}^{M-1} 1(x=k)} \tag{10}$$

where $1(\cdot)$ is the indication function

$$1(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases} .$$

– Taking the logarithm of (10), we have

$$\ln \mathrm{Cat}(x|\boldsymbol{\mu}) = \sum_{k=1}^{M-1} 1(x=k) \ln \frac{\mu_k}{\mu_0} + \ln \mu_0 = \sum_{k=1}^{M-1} 1(x=k) \ln\left(\frac{\mu_k}{1-\sum_{j=1}^{M-1} \mu_j}\right) + \ln\left(1-\sum_{j=1}^{M-1} \mu_j\right) \tag{11}$$

Distribution	Parameter(s)	Natural parameter(s)	Inverse parameter mapping	Base measure $h(x)$	Sufficient statistic $T(x)$	Log-partition $A(\boldsymbol{\eta})$	Log-partition $A(\boldsymbol{\theta})$
Bernoulli distribution	p	$\ln \frac{p}{1-p}$ • This is the logit function.	$\frac{1}{1+e^{-\eta}} = \frac{e^{\eta}}{1+e^{\eta}}$ • This is the logistic function.	1	x	$\ln(1+e^{\eta})$	$-\ln(1-p)$
binomial distribution with known number of trials n	p	$\ln \frac{p}{1-p}$	$\frac{1}{1+e^{-\eta}} = \frac{e^{\eta}}{1+e^{\eta}}$	$\binom{n}{x}$	x	$n\ln(1+e^{\eta})$	$-n\ln(1-p)$
Poisson distribution	λ	$\ln \lambda$	e^{η}	$\frac{1}{x!}$	x	e^{η}	λ
negative binomial distribution with known number of failures r	p	$\ln p$	e^{η}	$\binom{x+r-1}{x}$	x	$-r\ln(1-e^{\eta})$	$-r\ln(1-p)$
exponential distribution	λ	$-\lambda$	$-\eta$	1	x	$-\ln(-\eta)$	$-\ln \lambda$
Pareto distribution with known minimum value x_{m}	α	$-\alpha - 1$	$-1-\eta$	1	$\ln x$	$-\ln(-1-\eta) + (1+\eta)\ln x_{\mathrm{m}}$	$-\ln \alpha - \alpha \ln x_{\mathrm{m}}$
Weibull							

Figure 2: Distributions in the exponential family (see full list at https://en.wikipedia.org/wiki/Exponential_family).

$$\mathbf{u}(x) = \begin{bmatrix} 1(x=1) \\ \vdots \\ 1(x=M-1) \end{bmatrix}$$

$$\boldsymbol{\eta} = \begin{bmatrix} \ln\left(\frac{\mu_1}{1-\sum_{j=1}^{M-1}\mu_j}\right) \\ \vdots \\ \ln\left(\frac{\mu_{M-1}}{1-\sum_{j=1}^{M-1}\mu_j}\right) \end{bmatrix} \in \mathbb{R}^{M-1} \quad \text{natural parameters}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \frac{e^{\eta_1}}{1+\sum_{k=1}^{M-1} e^{\eta_k}} \\ \vdots \\ \frac{e^{\eta_{M-1}}}{1+\sum_{k=1}^{M-1} e^{\eta_k}} \end{bmatrix} \quad \text{mean parameters (softmax function)}$$

$$A(\boldsymbol{\eta}) = -\ln\left(1 - \sum_{k=1}^{M-1}\mu_k\right) = \ln(1 + \sum_{k=1}^{M-1} e^{\eta_k})$$

  - A categorical variable can also be represented as an $M \times 1$ vector $\mathbf{u}(x)$ (one-hot encoding).

- Other examples: Multinomial, binomial, Beta, Dirichlet, Poisson, ... (see Fig. 2)
- "Non-examples": Not in the exponential family: uniform, student t, ...

3 Maximum Likelihood Estimation for Models in the Exponential Family

- Consider N observations

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \underset{i.i.d}{\sim} p(\mathbf{x}|\boldsymbol{\eta}) \quad \text{(exponential family)} \tag{12}$$

$$\begin{aligned}\ln p(\mathbf{X}|\boldsymbol{\eta}) &= \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\boldsymbol{\eta}) \\ &= -NA(\boldsymbol{\eta}) + \boldsymbol{\eta}^T \sum_{n=1}^{N} \mathbf{u}(\mathbf{x}_n) + (\text{independent of } \boldsymbol{\eta}) \\ &= -NA(\boldsymbol{\eta}) + \sum_k \eta_k \left(\sum_{n=1}^{N} u_k(\mathbf{x}_n) \right) + (\text{independent of } \boldsymbol{\eta})\end{aligned} \tag{13}$$

- A key advantage of the exponential family is that computing the gradient of the log-likelihood is easy:

$$\frac{\partial}{\partial \eta_k} \ln p\left(\mathbf{X}|\boldsymbol{\eta}\right) = -N \frac{\partial}{\partial \eta_k} A\left(\boldsymbol{\eta}\right) + \sum_{n=1}^{N} u_k\left(\mathbf{x}_n\right), \tag{14}$$

but, as it can be verified, we have

$$\frac{\partial}{\partial \eta_k} A\left(\boldsymbol{\eta}\right) = E\left[u_k\left(\mathbf{x}\right)\right] = \mu_k \tag{15}$$

so

$$\frac{1}{N} \frac{\partial}{\partial \eta_k} \ln p\left(\mathbf{X}|\boldsymbol{\eta}\right) = \frac{1}{N} \sum_{n=1}^{N} u_k\left(\mathbf{x}_n\right) - \mu_k. \tag{16}$$

- In vector form:

$$\nabla_{\boldsymbol{\eta}} \ln p\left(\mathbf{X}|\boldsymbol{\eta}\right) = \sum_{n=1}^{N} \mathbf{u}\left(\mathbf{x}_n\right) - N \nabla_{\boldsymbol{\eta}} A\left(\boldsymbol{\eta}\right) \tag{17}$$

but

$$\nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) = \mathrm{E}\left[\mathbf{u}(\mathbf{x})\right] = \boldsymbol{\mu}, \tag{18}$$

so

$$\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{X}|\boldsymbol{\eta}) = \sum_{n=1}^{N} \mathbf{u}(\mathbf{x}_n) - N\boldsymbol{\mu} \tag{19}$$

or

$$\frac{1}{N} \nabla_{\boldsymbol{\eta}} \ln p(\mathbf{X}|\boldsymbol{\eta}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{u}(\mathbf{x}_n) - \boldsymbol{\mu}. \tag{20}$$

- Due to concavity, the ML estimate $\boldsymbol{\eta}_{\mathrm{ML}}$ is obtained (assuming no constrains on $\boldsymbol{\eta}$) by seeing the equation

$$\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{X}|\boldsymbol{\eta}) = \mathbf{0}. \tag{21}$$

- This gives

$$\boldsymbol{\mu}_{\mathrm{ML}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{u}(\mathbf{x}_n) \quad \text{(moment matching)}. \tag{22}$$

- From $\boldsymbol{\mu}_{\mathrm{ML}}$, we can also compute $\boldsymbol{\eta}_{\mathrm{ML}}$ if needed due to the one-to-one mapping between $\boldsymbol{\mu}_{\mathrm{ML}}$ and $\boldsymbol{\eta}_{\mathrm{ML}}$.

Examples:

- a) ML estimate of $\left(\mu, \sigma^2\right)$ for Gaussian model:

$$\mu_{\mathrm{ML}} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{23}$$

$$\sigma^2_{\mathrm{ML}} = \frac{1}{N} \sum_{n=1}^{N} x_n^2 - \mu^2_{\mathrm{ML}} \tag{24}$$

- b) ML estimate of μ for Bernoulli model:

$$\mathrm{E}[u(x)] = \mathrm{E}[x] = \mu \tag{25}$$

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n = \frac{N[1]}{N} \tag{26}$$

where $N[1]$ measures the number of observations equal to 1, i.e.,

$$N[1] = |\{n:\ x_n = 1\}|.$$

- c) ML estimate of $\boldsymbol{\mu}$ for Categorical model:

$$\mathrm{E}[\mathbf{u}(x)] = \begin{bmatrix} \mathrm{E}[1(x=1)] \\ \vdots \\ \mathrm{E}[1(x=M-1)] \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{M-1} \end{bmatrix} \tag{27}$$

$$\mu_{k,\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} 1(x_n = k) = \frac{N[k]}{N}, \tag{28}$$

where

$$N[k] = |\{n:\ x_n = k\}|.$$

Remark: The "black swan paradox" or zero-count problem: if we don't observe something (e.g., black swans for Europeans before 17^{th}century), we assign it zero probability with ML.

Remark: (Log-partition function and Fisher information [2, Ch. 8])

- We have seen that $\nabla_{\boldsymbol{\eta}} \mathrm{A}(\boldsymbol{\eta}) = \mathrm{E}[\mathbf{u}(\mathbf{x})]$
- The Hessian also has a special meaning, yielding the Fisher information matrix

$$\begin{aligned} \nabla^2_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) &= \mathrm{E}\left[\left(\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{X}|\boldsymbol{\eta})\right)\left(\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{X}|\boldsymbol{\eta})\right)^T\right] \\ &= -\mathrm{E}\left[\nabla^2_{\boldsymbol{\eta}} \ln p(\mathbf{X}|\boldsymbol{\eta})\right] = \mathbf{J}_{\boldsymbol{\eta}} \end{aligned} \tag{29}$$

- Furthermore, we have the relationship

$$\begin{aligned} \mathrm{KL}\left(p(\mathbf{x}|\boldsymbol{\eta}_1)||p(\mathbf{x}|\boldsymbol{\eta}_2)\right) &= A(\boldsymbol{\eta}_2) - A(\boldsymbol{\eta}_1) - \nabla A(\boldsymbol{\eta}_1)^T(\boldsymbol{\eta}_2 - \boldsymbol{\eta}_1) \\ &= \frac{1}{N}(\boldsymbol{\eta}_1 - \boldsymbol{\eta}_2)^T \mathbf{J}_{\boldsymbol{\eta}_1}(\boldsymbol{\eta}_1 - \boldsymbol{\eta}_2) + O\left(||\boldsymbol{\eta}_1 - \boldsymbol{\eta}_2||^3\right) \end{aligned} \tag{30}$$

- See also J. Duchi's notes, Ch. 8, and De Bruijn's identity.

4 Computation of Posterior and Predictive Distributions for the Exponential Family

- As we saw, in a Bayesian approach, we need to compute the posterior probability

$$p(\boldsymbol{\mu}|\mathbf{X}, \boldsymbol{\alpha})$$

given a prior probability $p(\boldsymbol{\mu}|\boldsymbol{\alpha})$ and the likelihood $p(\mathbf{X}|\boldsymbol{\mu})$.

- We have

$$p(\boldsymbol{\mu}|\mathbf{X}, \boldsymbol{\alpha}) = \frac{p(\boldsymbol{\mu}, \mathbf{X}|\boldsymbol{\alpha})}{p(\mathbf{X}|\boldsymbol{\alpha})} \propto p(\boldsymbol{\mu}|\boldsymbol{\alpha}) p(\mathbf{X}|\boldsymbol{\mu}). \tag{31}$$

- How to choose the prior?
  - Conjugate prior: choose prior $p(\boldsymbol{\mu}|\boldsymbol{\alpha})$ so that posterior $p(\boldsymbol{\mu}|\mathbf{X}, \boldsymbol{\alpha})$ has the same distribution type as $p(\boldsymbol{\mu}|\boldsymbol{\alpha})$.

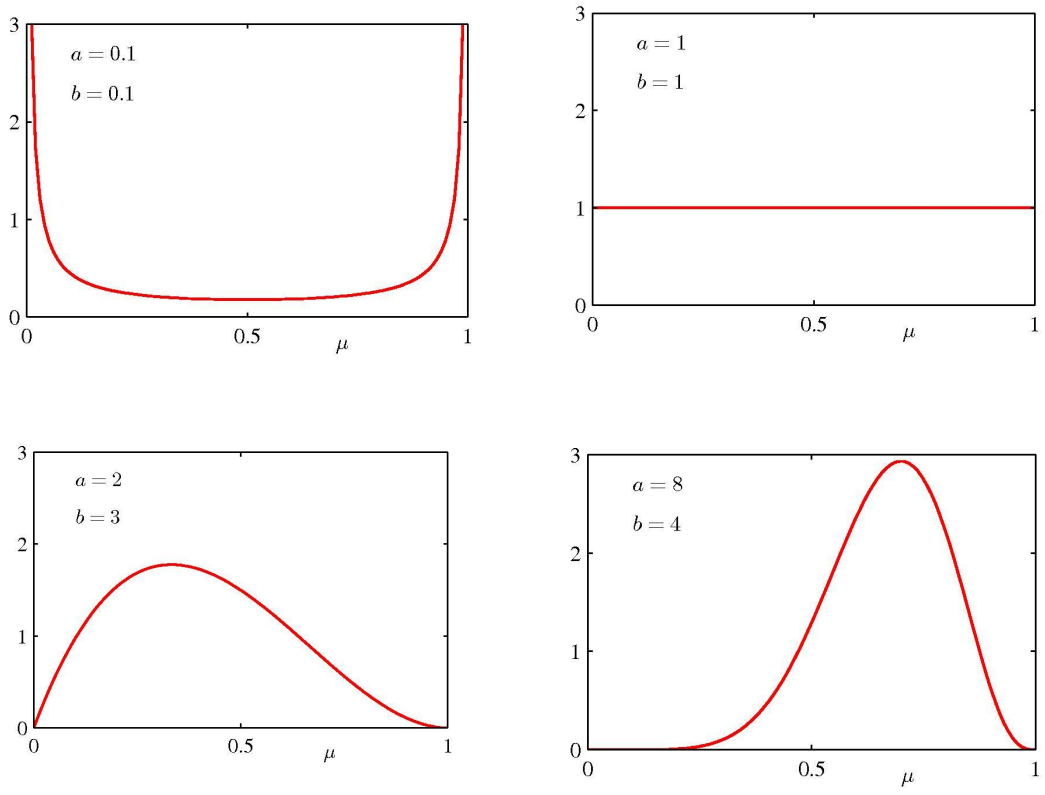


Figure 3: Plots of the Beta distribution as a function of μ for various values of the hyperparameters a and b (from [1]).

  - Non informative prior: "let the data speak for itself" (see [1, pp. 117-120])

- Exponential distributions have conjugate priors in the exponential family, as seen in the next example.

3.1. Beta-binomial Model

- Estimation of $\boldsymbol{\mu}$ for Bernoulli model:
  - Likelihood:

$$p(\mathbf{x}|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n} = \mu^{N[1]}(1-\mu)^{N[0]} \tag{32}$$

  - Conjugate prior:

$$\begin{aligned} p(\mu|a,b) &\propto \mu^{a-1}(1-\mu)^{b-1} \quad a,\ b \text{ hyperparameters (Fig. 3)} \\ &= \text{Beta}(\mu|a,b) \qquad\quad \text{Beta distribution} \end{aligned} \tag{33}$$

$$\mathrm{E}[\mu] = \frac{a}{a+b} \tag{34}$$

$$\text{mode} = \frac{a-1}{a+b-2} \quad (\text{if } a,b>1) \tag{35}$$

  - Posterior distribution:

$$\begin{aligned} p(\mu|\mathbf{x},a,b) &\propto \mu^{\sum_{n=1}^{N} x_n + a - 1}(1-\mu)^{n - \sum_{n=1}^{N} x_n + b - 1} \\ &= \text{Beta}\left(\mu|a + \sum_{n=1}^{N} x_n, b + N - \sum_{n=1}^{N} x_n\right) \quad (\text{Fig. 4}) \end{aligned} \tag{36}$$

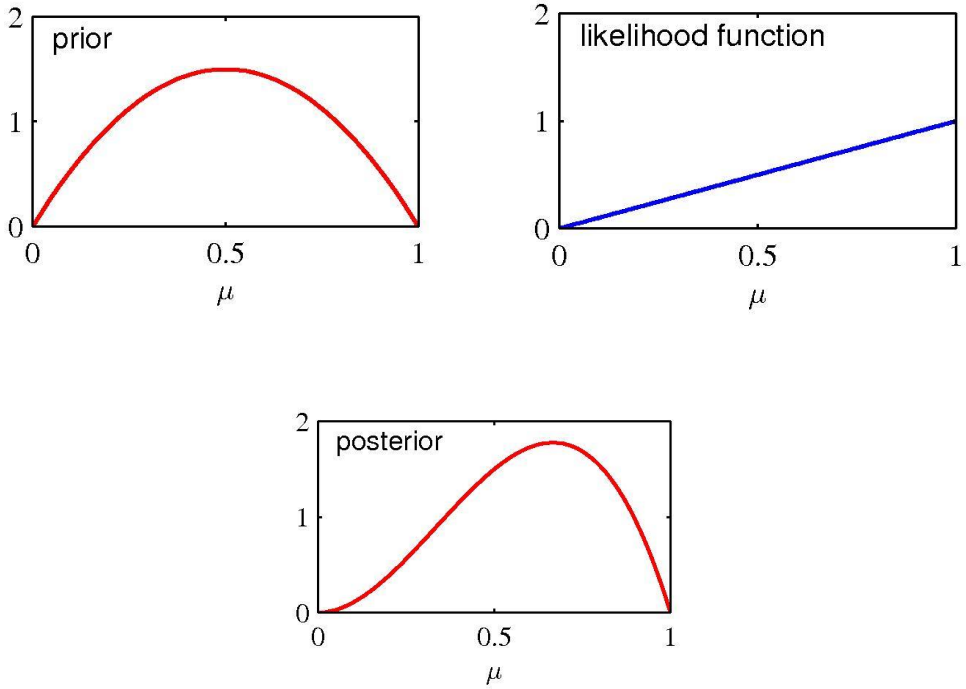


Figure 4: Illustration of the computation of the posterior distribution using (36) (from [1]).

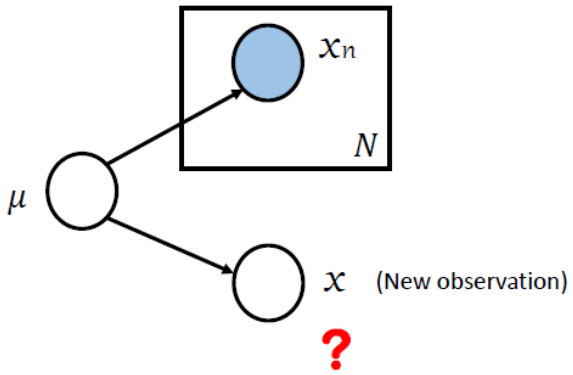


Figure 5: Bayesian network describing the joint distribution of parameter μ, data set $\{x_n\}$ and new observation x for the Beta-binomial model.

- By (36), the hyperparameter a and b can be interpreted as counts obtained from prior measurements.
- MAP estimate (if both parameters >1)

$$\mu_{\text{MAP}} = \frac{a + \sum_{n=1}^{N} x_n - 1}{a + b + N - 2} = \frac{a + N[1] - 1}{a + b + N - 2} \underset{N\to\infty}{\longrightarrow} \mu_{\text{ML}} \tag{37}$$

- Predictive distribution for Bayesian approach (see Fig. 5)

$$\begin{aligned} p(x = 1|\mathbf{x}, a, b) &= \int p(\mu|\mathbf{x}, a, b) p(x = 1|\mu) \mathrm{d}\mu \\ &= \mathrm{E}_{p(\mu|\mathbf{x},a,b)}[\mu] = \frac{N[1] + a}{N + a + b} \end{aligned} \tag{38}$$

- If N is small, $p(x = 1|\mathbf{x}, a, b) \approx \frac{a}{a+b}$
- If N is large, $p(x = 1|\mathbf{x}, a, b) \approx \frac{N[1]}{N}$

Ex: On amazon.com you have two sellers selling a product at the same price. The first has 90 positive reviews and 10 negative reviews, which the second has 2 positive reviews and 0 negative reviews. Which one to choose? We can compute the probability that the next review is positive via the predictive distribution, see Fig. 6.

3.2. Dirichlet-multinomial Model

- Estimation of $\boldsymbol{\mu}$ for multinoulli distribution:
  - Likelihood:

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=0}^{M-1} \mu_k^{N[k]} \tag{39}$$

  - Conjugate prior:

$$\begin{aligned} p(\boldsymbol{\mu}|\boldsymbol{\alpha}) &\propto \prod_{k=0}^{M-1} \mu_k^{\alpha_k - 1} \qquad \text{Dirichlet distribution} \\ &= \mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) \end{aligned} \tag{40}$$

    where α_k is the hyperparameter representing the number of "prior" observations equal to k

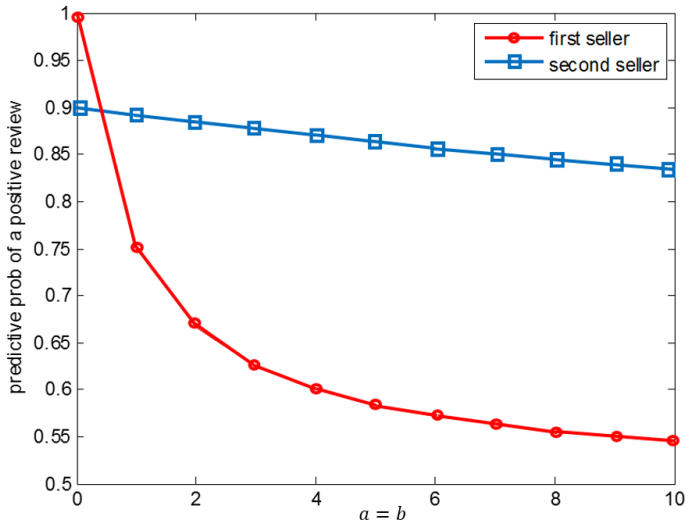


Figure 6: Probability that the next review is positive using the predictive distribution (38) for the example in Sec. 3.1.

$$\mathrm{E}[\mu_k] = \frac{\alpha_k}{\sum_{j=0}^{M-1} \alpha_j} \tag{41}$$

$$\text{mode} = \frac{\alpha_k - 1}{\sum_{j=0}^{M-1} \alpha_j - M} \quad (\text{if } \alpha > 1) \tag{42}$$

– Posterior:

$$p(\boldsymbol{\mu}|\mathbf{x}, \boldsymbol{\alpha}) \propto \prod_{k=0}^{M-1} \mu_k^{N[k]+\alpha_k} = \mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha} + N) \tag{43}$$

where $\mathbf{N} = [N[0], \cdots, N[M-1]]$.

– MAP estimate:

$$\mu_{k,\mathrm{MAP}} = \frac{\alpha_k + N[k] - 1}{\sum_{j=0}^{M-1} \alpha_j + N - M} \tag{44}$$

– Predictive distribution for Bayesian approach:

$$p(x = k|\mathbf{x}, \boldsymbol{\alpha}) = \frac{N[k] + \alpha_k}{N + \sum_{j=0}^{M-1} \alpha_j} \tag{45}$$

If N is small, $p(x = k|\mathbf{x}, \boldsymbol{\alpha}) \approx \alpha_k / \sum_{j=0}^{M-1} \alpha_j$.
If N is large, $p(x = k|\mathbf{x}, \boldsymbol{\alpha}) \approx N[k]/N$.

3.3. Gaussian-Gaussian Model

- Estimate of μ for a Gaussian distribution with σ^2 known:

– Likelihood:

$$p(\mathbf{x}|\mu) \propto \exp\left(-\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2\right) \tag{46}$$

– Conjugate prior:

$$p(\mu|\mu_0, \sigma_0^2) = \mathcal{N}(\mu|\mu_0, \sigma_0^2) \propto \exp\left(-\frac{(\mu - \mu_0)^2}{2\sigma_0^2}\right) \tag{47}$$

– Posterior:

$$\begin{aligned} p(\mu|\mathbf{x}, \mu_0, \sigma_0^2) &\propto \exp\left(-\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 - \frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\right) \\ &\propto \exp\left(-\frac{1}{2\sigma_N^2}(\mu - \mu_{\mathrm{MAP}})^2\right) \end{aligned} \tag{48}$$

Figure 2.12 Illustration of Bayesian inference for the mean μ of a Gaussian distribution, in which the variance is assumed to be known. The curves show the prior distribution over μ (the curve labelled $N = 0$), which in this case is itself Gaussian, along with the posterior distribution given by (2.140) for increasing numbers N of data points. The data points are generated from a Gaussian of mean 0.8 and variance 0.1, and the prior is chosen to have mean 0. In both the prior and the likelihood function, the variance is set to the true value.

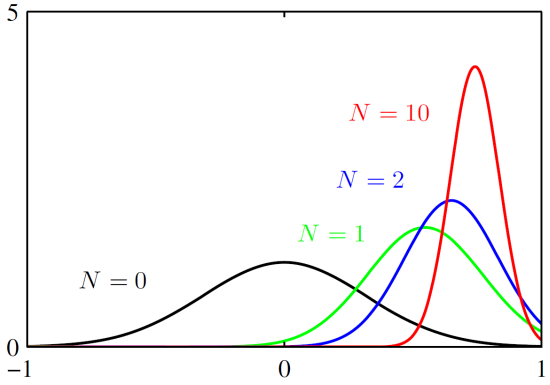


Figure 7: Posterior distribution for the Gaussian-Gaussian model with different values of N (from [1]).

where

$$\mu_{\mathrm{MAP}} = \frac{\sigma^2/N}{\sigma_0^2 + \sigma^2/N}\mu_0 + \frac{\sigma_0^2}{\sigma_0^2 + \sigma^2/N}\mu_{\mathrm{ML}} \tag{49}$$

$$\frac{1}{\sigma_N^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2} \tag{50}$$

If N is large, $\mu_N \approx \mu_{\mathrm{ML}}$ and $\sigma_N^2 \approx 0$ (see Fig. 7).

– Predictive distribution for Bayesian approach

$$p(x|\mathbf{x}, \mu_0, \sigma_0^2) = \mathcal{N}\left(x|\mu_{\mathrm{MAP}}, \sigma^2 + \sigma_N^2\right) \tag{51}$$

– For other examples of pairs of likelihood and conjugate prior, see Fig. 8 and Fig. 9.

5 Generalized linear model (GLM)

- Generalized linear models are popular probabilistic discriminative models for supervised learning.
- In GLMs with canonical link function, we have

$$p(t|\mathbf{x}, \mathbf{W}) \in \text{exponential family with } \boldsymbol{\eta} = \mathbf{W}\mathbf{x}$$

Ex:

– Linear regression

$$p(t|\mathbf{x}, \mathbf{W}) = \mathcal{N}(t|\mu = \boldsymbol{w}^T\mathbf{x}, \sigma^2) \tag{52}$$

$$(\eta = \mu/\sigma^2) \tag{53}$$

where $\eta = \mu/\sigma^2$.

Discrete distributions [edit]

Likelihood	Model parameters	Conjugate prior distribution	Prior hyperparameters	Posterior hyperparameters	Interpretation of hyperparameters[note 1]	Posterior predictive[note 2]
Bernoulli	p (probability)	Beta	α, β	$\alpha + \sum_{i=1}^{n} x_i, \beta + n - \sum_{i=1}^{n} x_i$	$\alpha - 1$ successes, $\beta - 1$ failures[note 1]	$p(\tilde{x} = 1) = \frac{\alpha'}{\alpha' + \beta'}$
Binomial	p (probability)	Beta	α, β	$\alpha + \sum_{i=1}^{n} x_i, \beta + \sum_{i=1}^{n} N_i - \sum_{i=1}^{n} x_i$	$\alpha - 1$ successes, $\beta - 1$ failures[note 1]	$\mathrm{BetaBin}(\tilde{x}\|\alpha', \beta')$ (beta-binomial)
Negative binomial with known failure number, r	p (probability)	Beta	α, β	$\alpha + \sum_{i=1}^{n} x_i, \beta + rn$	$\alpha - 1$ total successes, $\beta - 1$ failures[note 1] (i.e., $\frac{\beta - 1}{r}$ experiments, assuming r stays fixed)	
Poisson	λ (rate)	Gamma	k, θ	$k + \sum_{i=1}^{n} x_i, \frac{\theta}{n\theta + 1}$	k total occurrences in $\frac{1}{\theta}$ intervals	$\mathrm{NB}\left(\tilde{x}\|k', \frac{\theta'}{1+\theta'}\right)$ (negative binomial)
			α, β[note 3]	$\alpha + \sum_{i=1}^{n} x_i, \beta + n$	α total occurrences in β intervals	$\mathrm{NB}\left(\tilde{x}\|\alpha', \frac{1}{1+\beta'}\right)$ (negative binomial)
Categorical	$\boldsymbol{p}$ (probability vector), k (number of categories; i.e., size of $\boldsymbol{p}$)	Dirichlet	$\boldsymbol{\alpha}$	$\boldsymbol{\alpha} + (c_1, \ldots, c_k)$, where c_i is the number of observations in category i	$\alpha_i - 1$ occurrences of category i[note 1]	$p(\tilde{x} = i) = \frac{\alpha_i'}{\sum_i \alpha_i'} = \frac{\alpha_i + c_i}{\sum_i \alpha_i + n}$
Multinomial	$\boldsymbol{p}$ (probability vector), k (number of categories; i.e., size of $\boldsymbol{p}$)	Dirichlet	$\boldsymbol{\alpha}$	$\boldsymbol{\alpha} + \sum_{i=1}^{n} \mathbf{x}_i$	$\alpha_i - 1$ occurrences of category i[note 1]	$\mathrm{DirMult}(\tilde{\mathbf{x}}\|\boldsymbol{\alpha}')$ (Dirichlet-multinomial)
Hypergeometric with known total population size, N	M (number of target members)	Beta-binomial[4]	$n = N, \alpha, \beta$	$\alpha + \sum_{i=1}^{n} x_i, \beta + \sum_{i=1}^{n} N_i - \sum_{i=1}^{n} x_i$	$\alpha - 1$ successes, $\beta - 1$ failures[note 1]	
Geometric	p_0 (probability)	Beta	α, β	$\alpha + n, \beta + \sum_{i=1}^{n} x_i$	$\alpha - 1$ experiments, $\beta - 1$ total failures[note 1]	

Figure 8: Table of conjugate prior for discrete distributions (see full table at https://en.wikipedia.org/wiki/Conjugate_prior)

Continuous distributions [edit]

Likelihood	Model parameters	Conjugate prior distribution	Prior hyperparameters	Posterior hyperparameters	Interpretation of hyperparameters	Posterior predictive[note 4]
Normal with known variance σ^2	μ (mean)	Normal	μ_0, σ_0^2	$\left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{n} x_i}{\sigma^2}\right) \Big/ \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right), \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)^{-1}$	mean was estimated from observations with total precision (sum of all individual precisions)$1/\sigma_0^2$ and with sample mean μ_0	$\mathcal{N}(\tilde{x}\|\mu_0', \sigma_0^{2\prime} + \sigma^2)$[5]
Normal with known precision τ	μ (mean)	Normal	μ_0, τ_0	$\left(\tau_0\mu_0 + \tau\sum_{i=1}^{n} x_i\right) \Big/ (\tau_0 + n\tau), \tau_0 + n\tau$	mean was estimated from observations with total precision (sum of all individual precisions)τ_0 and with sample mean μ_0	$\mathcal{N}\left(\tilde{x}\|\mu_0', \frac{1}{\tau_0'} + \frac{1}{\tau}\right)$[5]
Normal with known mean μ	σ^2 (variance)	Inverse gamma	α, β [note 5]	$\alpha + \frac{n}{2}, \beta + \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{2}$	variance was estimated from 2α observations with sample variance β/α (i.e. with sum of squared deviations 2β, where deviations are from known mean μ)	$t_{2\alpha'}(\tilde{x}\|\mu, \sigma^2 = \beta'/\alpha')$[5]
Normal with known mean μ	σ^2 (variance)	Scaled inverse chi-squared	ν, σ_0^2	$\nu + n, \frac{\nu\sigma_0^2 + \sum_{i=1}^{n}(x_i - \mu)^2}{\nu + n}$	variance was estimated from ν observations with sample variance σ_0^2	$t_{\nu'}(\tilde{x}\|\mu, \sigma_0^{2\prime})$[5]
Normal with known mean μ	τ (precision)	Gamma	α, β[note 3]	$\alpha + \frac{n}{2}, \beta + \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{2}$	precision was estimated from 2α observations with sample variance β/α (i.e. with sum of squared deviations 2β, where deviations are from known mean μ)	$t_{2\alpha'}(\tilde{x}\|\mu, \sigma^2 = \beta'/\alpha')$[5]
Normal[note 6]	μ and σ^2 Assuming exchangeability	Normal-inverse gamma	$\mu_0, \nu, \alpha, \beta$	$\frac{\nu\mu_0 + n\bar{x}}{\nu + n}, \nu + n, \alpha + \frac{n}{2}, \beta + \frac{1}{2}\sum_{i=1}^{n}(x_i - \bar{x})^2 + \frac{n\nu}{\nu + n}\frac{(\bar{x} - \mu_0)^2}{2}$ • $\bar{x}$ is the sample mean	mean was estimated from ν observations with sample mean μ_0; variance was estimated from 2α observations with sample mean μ_0 and sum of squared deviations 2β	$t_{2\alpha'}\left(\tilde{x}\|\mu', \frac{\beta'(\nu'+1)}{\nu'\alpha'}\right)$[5]
Normal	μ and τ Assuming	Normal-gamma	$\mu_0, \nu, \alpha, \beta$	$\frac{\nu\mu_0 + n\bar{x}}{\nu + n}, \nu + n, \alpha + \frac{n}{2}, \beta + \frac{1}{2}\sum_{i=1}^{n}(x_i - \bar{x})^2 + \frac{n\nu}{\nu + n}\frac{(\bar{x} - \mu_0)^2}{2}$	mean was estimated from ν observations with sample mean μ_0, and precision was estimated from 2α observations with sample mean μ_0 and sum of	$t_{2\alpha'}\left(\tilde{x}\|\mu', \frac{\beta'(\nu'+1)}{\alpha'\nu'}\right)$[5]

Figure 9: Table of conjugate prior for continuous distributions (see full table at https://en.wikipedia.org/wiki/Conjugate_prior)

- Logistic regression (to be studied)
- Problem 2.

- Learning GLM can be done by means of gradient descent using (20) and the chain rule (see Problem 2).

6 Log-Linear Models and Boltzmann Machines

- These are "undirect" models based belonging to the exponential family that are typically used for unsupervised learning, as discussed in Chapter 5.

7 Energy-based Models

- An generalization of the exponential family is given by models of the form

$$p(\mathbf{x}|\theta) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-\sum_k E_k(\mathbf{x}|\theta)\right), \tag{54}$$

where $E_k(\mathbf{x}|\theta_k)$ are energy functions that identify patterns $\mathbf{x}$ of implausible variable configurations (i.e., values of $\mathbf{x}$ with high energy) as and $Z(\boldsymbol{\theta})$ is the partition function. The energy functions may depend in a more general way on variables $\mathbf{x}$ and parameters θ with respect to the exponential family. For instance, we may have

$$E_k(\mathbf{x}|\theta) = \alpha_i \ln\left(1 + (\theta_k \mathbf{x})^2\right), \tag{55}$$

with $\alpha_i > 0$, which corresponds to using a Student-t model.

- Energy-based models are typically represented via the graphical formalism of Markov networks.

- With energy-based models, the formula (20) generalizes as

$$-\frac{1}{N}\nabla_{\boldsymbol{\theta}} \ln p(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_k \nabla_{\boldsymbol{\theta}} E_k(\mathbf{x}_n|\theta) - \sum_k E_{\mathbf{x}\sim p(\mathbf{x}|\theta)}[\nabla_{\boldsymbol{\theta}} E_k(\mathbf{x}|\theta)]. \tag{56}$$

- The first term in (56) pushes down the energy of the observed samples $\mathbf{x}_n$ while the second term pushes up the energy of the "virtual" observations obtained from the model. The application of the first term is typically referred to as the positive phase, while the second is referred to as the negative phase (by some authors taken to model the working of the brain during dreams!).

- Note that, for the linear exponential family, the expectation for the negative phase readily yields the mean parameters, while, for more general models, the evaluation of this term is generally prohibitive and requires approximations.

8 Problems

1. Consider the exponential distribution

$$p(x) = \lambda \exp(-\lambda x)$$

for $x \geq 0$ and $p(x) = 0$ otherwise.

a. Show that it belongs to the exponential family by identifying sufficient statistic, mean and natural parameters.

b. Compute the ML estimate of the mean parameter $1/\lambda$.

c. Show that the derivative of the log-partition function equals the mean of the sufficient statistic.

2. Consider a regression problem in which we would like to learn a model relating a positive quantity t, representing, e.g., the time of the next update on a social network, with a vector of potentially correlated quantities $\mathbf{x} \in \mathbb{R}^D$. To this end, we focus on a probabilistic discriminative model given as

$$p(t|\mathbf{x}, \mathbf{w}) = \lambda(\mathbf{x}, \mathbf{w}) \exp(-\lambda(\mathbf{x}, \mathbf{w})t)$$

for $t \geq 0$ and $p(t|\mathbf{x}, \mathbf{w}) = 0$ otherwise, where $\lambda(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x}$ with $\mathbf{w} \in \mathbb{R}^D$. This is an example of a Generalized Linear Model (GLM), which is a class of models in which $p(t|\mathbf{x}, \mathbf{w})$ belongs to the exponential family and the natural parameters are linear functions of $\mathbf{x}$.

a. Compute the gradient $\nabla_{\mathbf{w}} \log p(t|\mathbf{x}, \mathbf{w})$ of the log-likelihood $\log p(t|\mathbf{x}, \mathbf{w})$. (Hint: Remember the chain rule for differentiation.)

We now would like to develop an algorithm to obtain the ML estimate of $\mathbf{w}$ given an i.i.d. dataset $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$. To this end, we use the stochastic gradient method. Accordingly, at each iteration of the algorithm, one sample $(\mathbf{x}_n, t_n)$ is selected from $\mathcal{D}$, e.g., uniformly with replacement. Then, we update the current estimate of $\mathbf{w}$ as

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + \gamma \nabla_{\mathbf{w}} \log p(t_n|\mathbf{x}_n, \mathbf{w}^{(i)}),$$

where i is the iteration index and γ is a parameter known as learning rate.

b. Set $\mathbf{w} = [1\ 4]^T$. Using MATLAB, generate $N = 100$ samples $\mathbf{x}_n$ with $D = 2$, in which the two entries are independent and uniformly distributed in the interval [0,1]. Then, generate the corresponding labels t_n using the given model with $\mathbf{w} = [1\ 4]^T$.

c. Implement the stochastic gradient algorithm and plot the estimates $\mathbf{w}^{(i)}$ as a function of $i = 1, 2, ..., 500$ (Set $\mathbf{w}^{(0)} = [2\ 2]^T$ and $\gamma = 0.1$.) Plot also the negative log-likelihood of $\mathbf{w}^{(i)}$ for the entire data set $\mathcal{D}$ as a function of i in a separate figure. Note that the negative log-likelihood can be taken as a measure of the training error and hence it should be decreasing with i if the learning rate is properly selected.

3. In an English text of 2000 letters, we observe the letter 'e' 260 times. Using a Dirichlet-multinomial model in which the hyperparameters of the Dirichlet prior are $\alpha_k = 10$ for $k = 1, ..., 27$ (there are 26 Roman letter plus the space character), compute the predictive probability that the next letter is 'e'.

4. We wish the use the Gaussian-Gaussian model to estimate the mean of a Gaussian distribution. To this end, assume a prior distribution with mean 0 and variance 0.1. Generate $N = 10$ independent Gaussian random variables x_n with mean 0.8 and variance 0.1. Plot in the same figure the prior distribution and the posterior distribution for $N = 1, ..., 10$. What happens when $N \to \infty$?

5. Consider the distribution

$$p(x|b) = \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right),$$

where the mean μ is a known number, while $b \geq 0$ is unknown and needs to be learned.

a. Show that $p(x|b)$ belongs to the exponential family by identifying sufficient statistic, natural parameters and log-partition function.

b. Compute the ML estimate of the mean parameter b (Note that the mean parameter is not the same as the mean of the random variable $X \sim p(x)$ and that we have $\int_{-\infty}^{\infty} |x - \mu| p(x) dx = b$).

c. Derive a stochastic gradient algorithm to obtain the ML estimate of the mean parameter b.

d. Describe how to obtain the posterior $p(b|\mathbf{x})$ based on observations $\mathbf{x} = (x_1, ..., x_N) \sim_{i.i.d.} p(x)$ under a Gaussian prior distribution on b with zero mean and variance σ^2 (known). What is the main complication in computing $p(b|\mathbf{x})$?

9 Appendix: Entropy and Differential Entropy

- The entropy $H(x)$ of a discrete random variable $x \sim p(x)$ taking values over a finite alphabet is defined as

$$H(x) = -\sum_x p(x) \ln p(x) \tag{57}$$

  and measures the "randomness" or "unpredictability" of x.

- Some properties:
  - $H(x) \geq 0$ with equality iff x is deterministic.
  - $H(x) \leq \ln(\text{cardinality of } x)$ with equality iff x is uniform (maximum entropy).

- One can think of $H(x)$ as measuring the logarithm of the "effective" number of possible realizations for x.

- The differential entropy $h(x)$ of a continuous random variable $x \sim p(x)$ is defined as

$$h(x) = -\int p(x) \ln p(x) \, dx \tag{58}$$

  and measures the "randomness" or "unpredictability" of x.

- Unlike the entropy, $h(x)$ can be negative. One can think of $h(x)$ as measuring the logarithm of the "effective" support of the pdf of x.

References

[1] C. Bishop, *Pattern recognition and Machine Learning,*, Springer, 2006.

[2] J. Duchi, Lecture notes, http://stanford.edu/class/stats311/Lectures/full-notes.pdf

3. Linear Models for Classification

1 Introduction

- Classification is a supervised learning problem characterized as:
  - Given: A training set $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}$, where $(x_n, t_n) \underset{i.i.d.}{\sim} p_0(\mathbf{x}, t)$ and

$p_0(\mathbf{x}, t)$		true (unknown) distribution
$(\mathbf{x}_1, \ldots, \mathbf{x}_N)$	$(\mathbf{x}_n \in \mathbb{R}^D)$	domain variables
$(t_1, \ldots, t_N)$		labels, where t_n indicates the class C_k, with $k = 1, ..., K$, to which x_n belongs.

  - Goal: Assign a new vector $\mathbf{x}$ to a class C_k, (see Fig. 1)

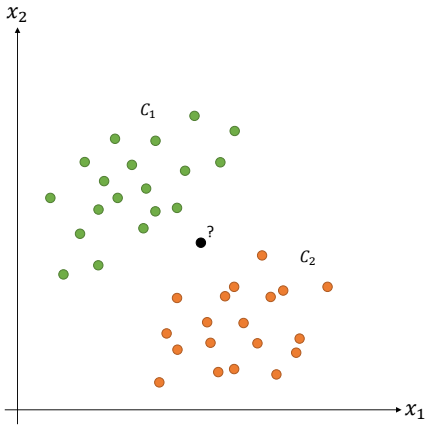


Figure 1: Definition of the classification problem ($D = 2$ in this figure).

- For two classes ($K = 2$), e.g., for spam detection, the mapping between class and label is given as

$$C_0 \rightarrow t = 0 \text{ or } = -1$$
$$C_1 \rightarrow t = 1$$

- For multiple classes, e.g., for credit scoring, text classification, we instead typically use one-hot encoding

$$C_k \rightarrow t = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix},$$

  where the 1 element corresponds to position $k + 1$.

- In this chapter, we focus on $K = 2$. The extension to $K > 2$ will be shortly discussed in Sec. 5.

- Three modeling approaches:
  - Deterministic (discriminative) model:
    - $t = y(\mathbf{x}, \mathbf{w})$
  - Probabilistic discriminative model:
    - $p(C_k|\mathbf{x})$
  - Probabilistic generative model:
    - $p(C_k)$ and $p(\mathbf{x}|C_k)$

2 Deterministic Discriminative Model

2.1 Model

- Definition:

$$t = y(\mathbf{x}, \tilde{\mathbf{w}}) = f\left(\sum_{d=1}^{D} w_d x_d + w_0\right) = f\left(\mathbf{w}^T\mathbf{x} + w_0\right) = f\left(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}\right) \tag{1}$$

where

$$t \in \{\pm 1\}$$
$$f(\cdot) \quad \text{activation function}$$
$$f(a) = \begin{cases} 1 & \text{if } a > 0 \\ -1 & \text{if } a < 0 \end{cases} = \text{sign}(a)$$
$$a(\mathbf{x}, \tilde{\mathbf{w}}) = \mathbf{w}^T\mathbf{x} + w_0 \quad \text{activation}$$
$$\tilde{\mathbf{w}} = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$
$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$

- Interpretations:
  - Project $\mathbf{x}$ into direction $\mathbf{w}$ and then apply threshold rule

$$\mathbf{w}^T\mathbf{x} \gtrless -w_0 \tag{2}$$

  - Hyperplane decision surface (see Fig. 2 with $D = 2$)

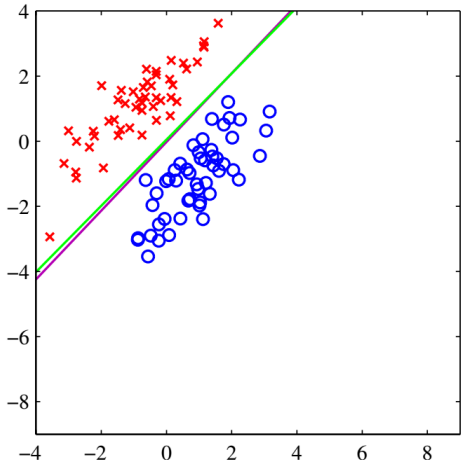


Figure 2: Training data from two classes. The decision hyperplanes are obtained here via the least squares (magenta) and logistic regression (green) learning rules (to be discussed).

- Linear deterministic models can be interpreted as deterministic single neuron, or McCulloch–Pitts, models as illustrated in Fig. 3).

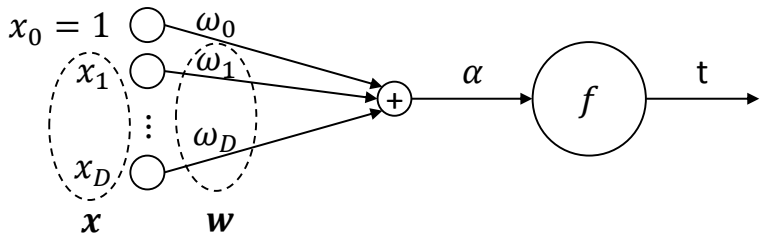


Figure 3: Deterministic single neuron model.

- Important facts about the geometry of the problem (see Fig. 4):
  - $a(\mathbf{x}, \tilde{\mathbf{w}}) = 0$ defines the decision hyperplane (or surface)
  - $-\frac{w_0}{||\mathbf{w}||}$ = bias of the decision surface in the direction $\mathbf{w}$
  - $\frac{||a(\mathbf{x},\tilde{\mathbf{w}})||}{||\mathbf{w}||}$ = distance of $\mathbf{x}$ to the decision surface (see)

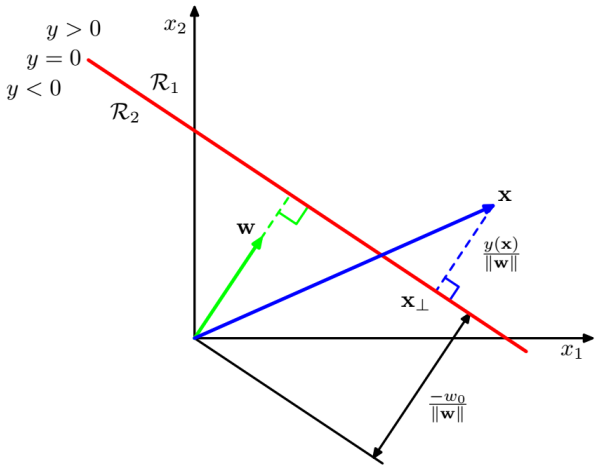


Figure 4: Geometry of the binary classification problem using deterministic linear models [Bishop, 2006].

- $|a(\mathbf{x}, \tilde{\mathbf{w}})|$ can be interpreted as a measure of the *confidence* with which the model classifies $\mathbf{x}$ in either class.
- In many problems, it is convenient to work with features

$$\phi_k(\mathbf{x}), \quad k = 1, \ldots, D'$$

in lieu of $\mathbf{x}$ as inputs to the classifier. This is, for instance, the case when:

  - D is large, and hence learning may be prone to overfitting;
    ex.: email spam classification ($\mathbf{x}$ = email text, $\phi_k(\mathbf{x})$ = count of a "suspicious" words).
  - D changes from sample to sample
    ex.: email spam classification.
  - $p_0(\mathbf{x}, t)$ is not well separable by a hyperplane, and hence learning directly using a linear model would cause bias
    ex.: Fig. 5.

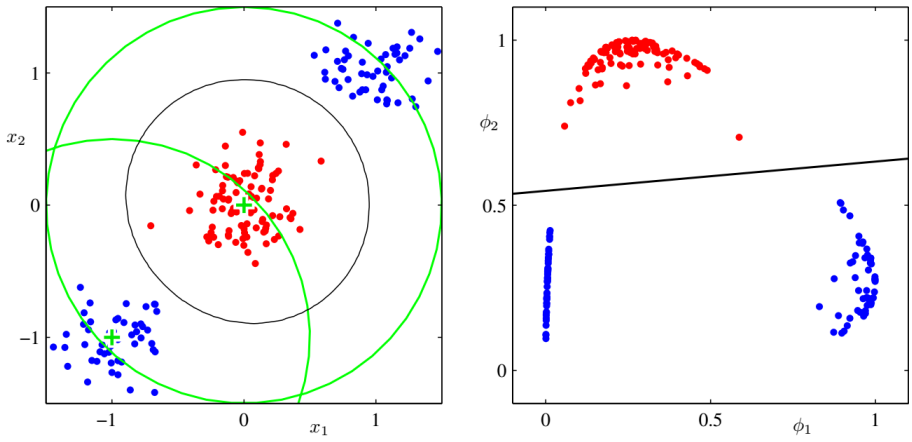


Figure 5: Example of the usefulness of using non-linear features as inputs to a linear classifier. Here the features are Gaussian pdfs centered at the green crosses [Bishop, 2006].

- Feature-based model:

$$t = y\left(\boldsymbol{\phi}(\mathbf{x}), \tilde{\mathbf{w}}\right) = f\left(\sum_{k=1}^{D'} w_k \phi_k(\mathbf{x})\right) = f\left(\tilde{\mathbf{w}}^T \boldsymbol{\phi}(\mathbf{x})\right) \tag{3}$$

where $\phi_0(\mathbf{x}) = 1$ and $\tilde{\mathbf{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{D'} \end{bmatrix}$. Note that (1) is a special case of (3) with $\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$.

- We can have:
  - $D > D'$ over-complete representation
  - $D < D'$ dimensionality reduction
- The feature-base model can be interpreted as a three-layer neural network in which the operation between the first two layers is fixed while the weights between second and third layers needs to be learned (see Fig. 6).

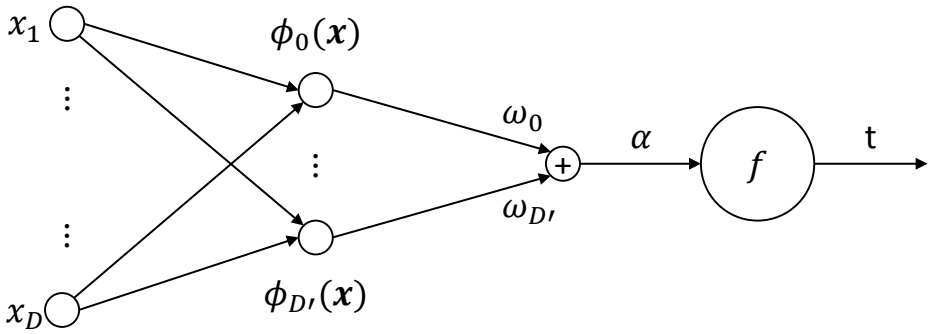


Figure 6: Three-layer neural network representing the feature-based model (3).

2.2 Learning

- As seen in Chapter 1, with deterministic discriminative models, inference is done by minimizing directly the empirical risk:

$$\underset{\tilde{\mathbf{w}}}{\text{minimize}} \quad \frac{1}{N}\sum_{n=1}^{N} \ell\left(\tilde{\mathbf{w}}, (\mathbf{x}_n, t_n)\right), \tag{4}$$

This is called empirical risk minimization (ERM) with loss function $\ell(\cdot)$.

- The empirical loss is an approximation of the average loss

$$E_{(\mathbf{x},t)\sim p_0(\mathbf{x},t)}\left[\ell\left(\tilde{\mathbf{w}}, (\mathbf{x}, t)\right)\right] \tag{5}$$

- It is also typical to add a regularization term to combat overfitting, yielding regularized ERM

$$\underset{\tilde{\mathbf{w}}}{\text{minimize}} \quad \frac{1}{N}\sum_{n=1}^{N} \ell\left(\tilde{\mathbf{w}}, (\mathbf{x}_n, t_n)\right) + R\left(\tilde{\mathbf{w}}\right) \tag{6}$$

The regularization term is typically convex but possibly not differentiable, e.g.,$R\left(\tilde{\mathbf{w}}\right) = ||\tilde{\mathbf{w}}||_1$.

- Given a sample $(\mathbf{x}_n, t_n)$, we define the following quantities illustrated in Fig. 7:

$$\begin{array}{ll} \frac{t_n a(\mathbf{x}_n, \tilde{\mathbf{w}})}{||\mathbf{w}||} & \text{geometric margin} \\ t_n a\left(\mathbf{x}_n, \tilde{\mathbf{w}}\right) & \text{functional margin} \end{array} \Bigg\} \begin{array}{ll} > 0 & \text{if correctly classified} \\ < 0 & \text{otherwise} \end{array}$$

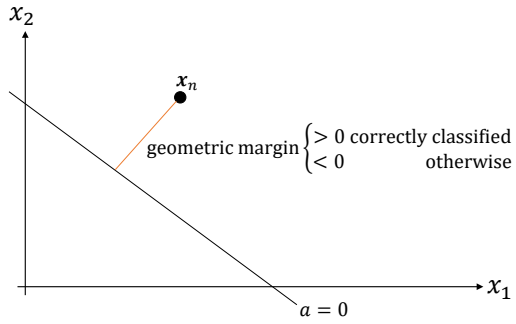


Figure 7: Illustration of the geometric margin.

- A natural choice for the loss function would be the 0-1 loss illustrated in Fig. 8:

$$\ell\left(\tilde{\mathbf{w}}, (\mathbf{x}, t)\right) = \begin{cases} 0 & \text{if } t\, a\left(\mathbf{x}, \tilde{\mathbf{w}}\right) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

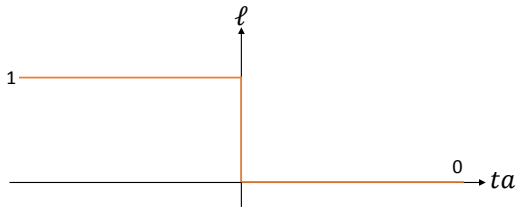


Figure 8: The 0-1 loss.

but the direct solution of the ERM problem, or regularized ERM problem, with the 0-1 loss is prohibitively complex.

2.3 Learning Algorithms

- We will review the following algorithms:
  - Least squares
  - Perceptron algorithm
  - Support vector machine (SVM)

2.3.1 Least Squares (LS)

- Least Squares (LS) solves the ERM problem with squared error loss $\ell\left(\tilde{\mathbf{w}}, (\mathbf{x}, t)\right) = \left(t - \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}\right)^2$:

$$\underset{\tilde{\mathbf{w}}}{\text{minimize}} \quad \sum_{n=1}^{N} \left(t_n - \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_n\right)^2 \tag{8}$$

$$\Leftrightarrow \underset{\tilde{\mathbf{w}}}{\text{minimize}} \quad \left|\left|\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{t}\right|\right|^2 \tag{9}$$

$$\text{where } \tilde{\mathbf{X}} = \begin{pmatrix} \tilde{\mathbf{x}}_n^T \\ \vdots \\ \tilde{\mathbf{x}}_N^T \end{pmatrix} \text{ and } \mathbf{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix} \tag{10}$$

- Solution: Assuming $N \geq D + 1$ and full-rank $\tilde{\mathbf{X}}$, we have

$$\tilde{\mathbf{w}} = \underbrace{\left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^T}_{\text{pseudoinverse of } \tilde{\mathbf{X}}} \mathbf{t} \tag{11}$$

$$\Rightarrow a\left(\tilde{\mathbf{w}}, \mathbf{x}\right) = \mathbf{t}^T \tilde{\mathbf{X}} \left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}\right)^{-1} \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \tag{12}$$

- The matrix to be inverted is of size $(D + 1) \times (D + 1)$.
- LS is sensitive to outliers: squared error penalizes predictions that are "too correct" as illustrated in Fig. 9 $\Rightarrow$ overfitting

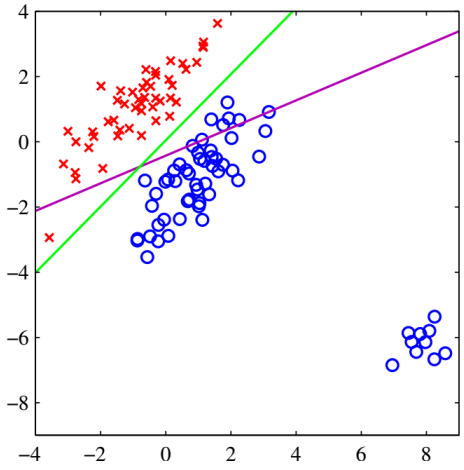


Figure 9: Illustration of the overfitting problem for LS. The data in the bottom right corner represent outliers that cause the decision region of LS (purple line) to tilt in a way that is likely to affect negatively the generalization error [Bishop, 2006].

- LS can be interpreted as ML estimation under the model $t \sim \mathcal{N}\left(t|\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}, \beta^{-1}\right)$, which does not even account for binary nature of t. We can hence think of the limitations of LS as a problem of bias.
- Ridge regression is regularized ERM with quadratic loss and regularization function $R\left(\tilde{\mathbf{w}}\right) = \lambda\left|\left|\tilde{\mathbf{w}}\right|\right|^2$, which yields

$$\tilde{\mathbf{w}} = \left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} + \lambda\mathbf{I}\right)^{-1}\tilde{\mathbf{X}}^T\mathbf{t} \tag{13}$$

$$\Rightarrow a\left(\tilde{\mathbf{w}}, \mathbf{x}\right) = \mathbf{t}^T\tilde{\mathbf{X}}\left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} + \lambda\mathbf{I}\right)^{-1}\begin{bmatrix}1\\ \mathbf{x}\end{bmatrix} \tag{14}$$

- Considering a generalized model with features $\boldsymbol{\phi}\left(\mathbf{x}\right)$ requires to substitute $\tilde{\mathbf{X}}$ with $\boldsymbol{\Phi} = \begin{pmatrix}\boldsymbol{\phi}\left(\mathbf{x}_1\right)^T\\ \vdots\\ \boldsymbol{\phi}\left(\mathbf{x}_N\right)^T\end{pmatrix}$ and hence the LS solution becomes:

$$a\left(\mathbf{x}, \tilde{\mathbf{w}}\right) = \mathbf{t}^T\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\phi}(\mathbf{x}) \tag{15}$$

and with ridge regression we have:

$$a\left(\mathbf{x}, \tilde{\mathbf{w}}\right) = \mathbf{t}^T\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{1}\right)^{-1}\boldsymbol{\phi}(\mathbf{x}) \tag{16}$$

- The matrix to be inverted is of size $(D'+1)\times(D'+1)$.

2.3.2 Kernel Methods

- Equation (16) can also be written as

$$a\left(\mathbf{x}\right) = \mathbf{t}^T\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\mathbf{I}\right)^{-1}\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}), \tag{17}$$

by using the *matrix inversion lemma*, where we dropped for convenience the dependence on the optimal $\tilde{\mathbf{w}}$.

- The matrix to be inverted is of size $N \times N$, making the approach attractive if $D+1 > N$, i.e., for high-dimensional data.
- Define

$$K(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y}), \qquad \text{for } \mathbf{x}, \mathbf{y} \in \mathbb{R}^D \tag{18}$$

which we will refer to as a Kernel function. Note that $K\left(\mathbf{x}, \mathbf{y}\right)$ measures the correlation of $\mathbf{x}$ and $\mathbf{y}$ in the feature space.

- Then, the equation above can be written as a function of the correlation between pairs of training points, namely $K\left(\mathbf{x}_n, \mathbf{x}_m\right)$, and between training points and the new data point, namely $K\left(\mathbf{x}_n, \mathbf{x}\right)$. In fact, we can write (17) as

$$a\left(\mathbf{x}\right) = \sum_{n=1}^{N}\alpha_n K\left(\mathbf{x}, \mathbf{x}_n\right) \tag{19}$$

where

$$\boldsymbol{\alpha} = \left(\mathbf{K} + \lambda\mathbf{I}\right)^{-1}\mathbf{t} \qquad \text{with } \left[\mathbf{K}\right]_{mn} = K\left(\mathbf{x}_m, \mathbf{x}_n\right) \tag{20}$$

- When most α_n are non-zero, the equation above requires the sum over N terms, which may be expensive.
- The key point, however, is that one can use (19) with any other kernel function, where a kernel function is any symmetric function measuring the correlation of two data points in some feature space, of possibly infinite dimension. This is known as the *kernel trick*.
- Examples:

1. The polynomial kernel

$$K(\mathbf{x}, \mathbf{y}) = \left(\gamma \mathbf{x}^T \mathbf{y} + r\right)^M \qquad \text{, where } r > 0 \tag{21}$$

corresponds to a correlation $\boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{y})$ in a high-dimensional space D'. For instance, with $M = 2$, we have $D' = 6$ with feature vector:

$$\phi(x) = \left[1,\ \sqrt{2}x_1,\ \sqrt{2}x_2,\ x_1^2 x_2^2,\ \sqrt{2}x_1 x_2\right]^T \tag{22}$$

2. The Gaussian kernel

$$K(\mathbf{x}, \mathbf{y}) = e^{-\gamma ||x-y||^2} \tag{23}$$

corresponds to an inner product in an infinite dimensional space!

- The issue of complexity due to the large number of $\alpha_n > 0$ is addressed by *sparse* kernel machines, such as SVM.
- Murphy [Murphy, 2012] provides an extensive discussion about kernel methods.
- As a generalization of kernel methods, we can consider learning rules of the form [Friedman et al., 2001]

$$a(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}_n),$$

where the kernel $K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}_n)$ depends on $\mathcal{X} = \{x_n\}_{n=1}^N$. An important example is the k-Nearest Neighbor (k-NN), which chooses a($\mathbf{x}$) as the average of the labels of the k closest data points, that is $\alpha_n = t_n/k$ and $K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}_n) = 1$ if $\mathbf{x}_n$ is one of the k closest points to $\mathbf{x}$ while otherwise we have $K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}_n) = 0$. An example of decision regions obtained with k-NN is shown in Fig. 10.

- k-NN is an example of a non-parametric learning rule in the sense that, in contrast to the other schemes studied so far and in the rest of the course, it does not rely on a parametric model of the (probabilistic) relationship between input and output. Instead, k-NN relies on the assumption that the labels of nearby points $\mathbf{x}$ should be similar, i.e., it assumes that the relationship between $\mathbf{x}$ and t is smooth. The non-parametric nature of k-NN makes it susceptible to the course of dimensionality: the number N of training samples required to maintain a certain performance level scales exponentially with the data dimension D (or D' for feature-based models) [Friedman et al., 2001].

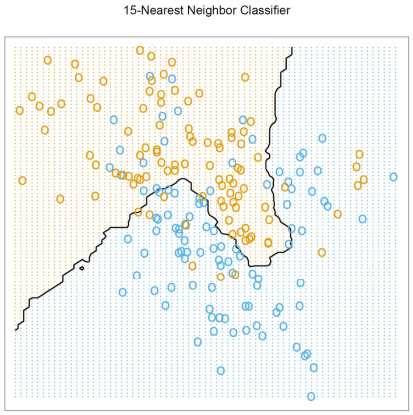


Figure 10: Illustration of decision regions for the k-NN algorithm [Friedman et al., 2001].

2.3.3 Perceptron Algorithm

- The perceptron algorithm was invented in 1957 by Frank Rosenblatt at the Cornell Aeronautical Laboratory. In figure 11 the first implementation in hardware of the "Mark 1 perceptron" is illustrated.

Figure 11: Mark 1 perceptron. The inputs were obtaining using a simple camera shown in the left photograph. The middle photograph shows the perceptron's patch board for trying different configurations of input features. Racks of adaptive weights, which could be adjusted automatically by the learning algorithm, shown in the right photograph [Bishop, 2006].

- For generality, we use the feature-base model

$$a\left(\mathbf{x}, \tilde{\mathbf{w}}\right) = \tilde{\mathbf{w}}^T \boldsymbol{\phi}(\mathbf{x}) \tag{24}$$

- The perceptron algorithm is an ERM scheme with perceptron loss defined as:

$$\ell\left(\tilde{w}, (\mathbf{x}, t)\right) = \begin{cases} -t\left(\tilde{\mathbf{w}}^T \boldsymbol{\phi}\left(\mathbf{x}\right)\right) & \text{if } t\left(\tilde{\mathbf{w}}^T \boldsymbol{\phi}\left(\mathbf{x}\right)\right) < 0 \text{ (incorrectly classified)} \\ 0 & \text{if } t\left(\tilde{\mathbf{w}}^T \boldsymbol{\phi}\left(\mathbf{x}\right)\right) \geq 0 \text{ (correctly classified)} \end{cases} \tag{25}$$

$$= \max\left(-t\left(\tilde{\mathbf{w}}^T \boldsymbol{\phi}\left(\mathbf{x}\right)\right), 0\right) \tag{26}$$

and illustrated in Fig. 12 along with the 0-1 loss.

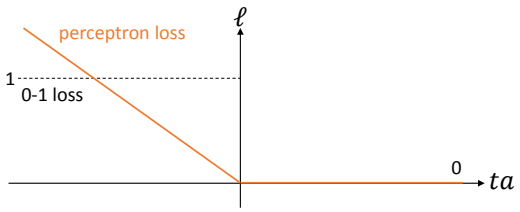


Figure 12: Perceptron loss.

- ERM problem:

$$\underset{\tilde{\mathbf{w}}}{\text{minimize}} \quad \underbrace{\sum_{n=1}^{N} \max\left(0, -t_n\left(\tilde{\mathbf{w}}^T \boldsymbol{\phi}\left(\mathbf{x}_n\right)\right)\right)}_{\text{sum-functional margin over misclassified examples}} \tag{27}$$

- Non-differentiable objective.
- The perceptron algorithm uses a stochastic gradient approach:
  - initialize $\tilde{\mathbf{w}}^{(0)}$
  - for each training example n (sequentially or randomly selected):
    - if correctly classified: $\tilde{\mathbf{w}}^{(n)} \leftarrow \tilde{\mathbf{w}}^{(n-1)}$

* otherwise

$$\tilde{\mathbf{w}}^{(n)} \leftarrow \tilde{\mathbf{w}}^{(n-1)} - \nabla_{\tilde{\mathbf{w}}} \underbrace{\ell\left(\tilde{\mathbf{w}}, (\mathbf{x}_n, t_n)\right)}_{-\tilde{\mathbf{w}}^T \phi(\mathbf{x}_n) t_n} = \tilde{\mathbf{w}}^{(n-1)} + \boldsymbol{\phi}\left(\mathbf{x}_n\right) t_n \tag{28}$$

* repeat until convergence

- Fig. 13 shows a visualization of the first steps of the perceptron algorithm.

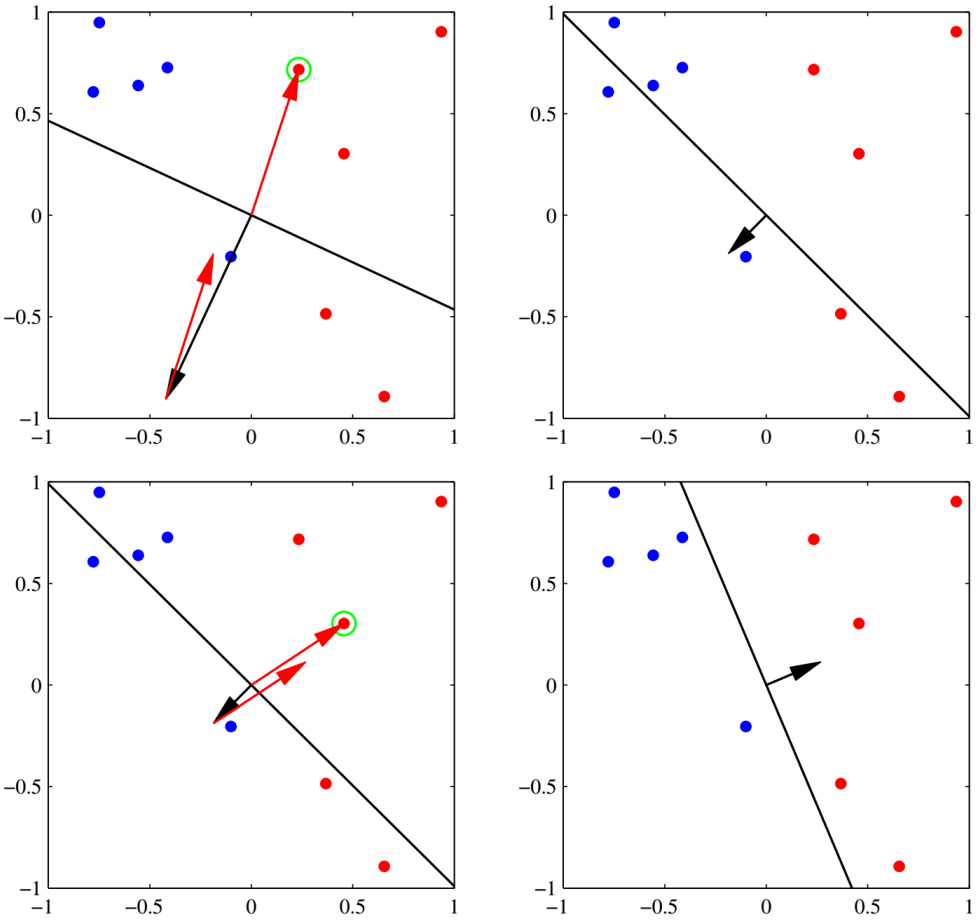


Figure 13: Illustration of the operation of the perceptron algorithm: the black arrow is the perpendicular $\mathbf{w}$ to the decision hyperplane [Bishop, 2006].

- At each step, the algorithm reduces the term in the perceptron loss related to training example n.
- It can be proved that, if the training set is linearly separable, then the perceptron algorithm will find a n that separates the two classes in a finite number of steps.
- Convergence can be slow.

2.3.4 Support Vector Machine (SVM)

- Assume first that the data set $\mathcal{D}$ is linearly separable in the given feature space.
- In this section, we write the activation as: $a(\mathbf{x}, \tilde{\mathbf{w}}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0$ to emphasize offset w_0, where $\phi(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_{D'}(\mathbf{x}) \end{bmatrix}$.

- By assumption of linear separability, there is hence a $\tilde{\mathbf{w}}$ such that the geometric margin satisfies

$$\underbrace{t_n \frac{a(\mathbf{x}, \tilde{\mathbf{w}})}{\|\mathbf{w}\|}}_{\text{geometric margin}} > 0 \text{ for all } n \tag{29}$$

- SVM attempts to maximize the minimum margin

$$\text{maximize} \min_{n=1,\ldots,N} t_n \frac{a(\mathbf{x_n}, \tilde{\mathbf{w}})}{\|\mathbf{w}\|} \tag{30}$$

which is not an ERM problem (at least not yet!). This is illustrated in Fig. 14.

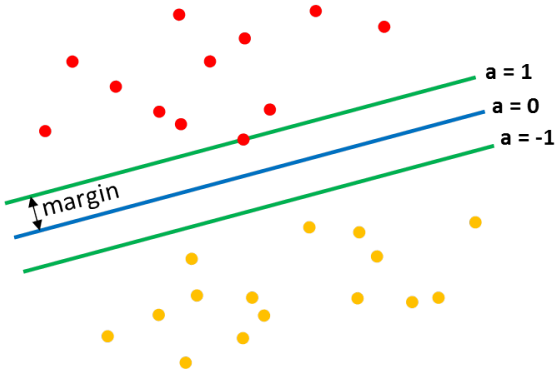


Figure 14: The margin is defined as the distance between the decision boundary and the closest of the data points.

- We can always choose a value of $\tilde{\mathbf{w}}$ such that

$$\min_{n=1,\ldots,N} t_n a(\mathbf{x_n}, \tilde{\mathbf{w}}) = 1 \tag{31}$$

i.e., such that the functional margin equals 1 for the support vectors $\mathbf{x_n}$ and is larger than 1 for all other vectors $\mathbf{x_n}$. As a result of this choice, we have

$$\min_{n=1,\ldots,N} t_n \frac{a(\mathbf{x_n}, \tilde{\mathbf{w}})}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}. \tag{32}$$

- The problem (30) can hence be reformulated as

$$\begin{aligned} \underset{\mathbf{w}, w_o}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & t_n(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_n}) + w_o) \geq 1 \quad \text{for } n = 1, ..., N \end{aligned} \tag{33}$$

- At an optimum value $\tilde{\mathbf{w}}$, there are at least two support vectors as seen in Fig. 15.

- This is a convex quadratic program: it can be solved by means of solvers such as CVX.
- Using Lagrange duality, the Lagrange multipliers associated to the support vectors can be identified because they are positive.
- The dual problem can be formulated in terms of the kernel function $k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{y})$ and is a quadratic problem ([Bishop, 2006, p. 329]). Furthermore, the resulting classifier is given as

$$a(\mathbf{x}, \tilde{\mathbf{w}}) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}, \mathbf{x_n}) \tag{34}$$

where only the α_n variables corresponding to support vectors are non-zero: This is a sparse kernel machine.

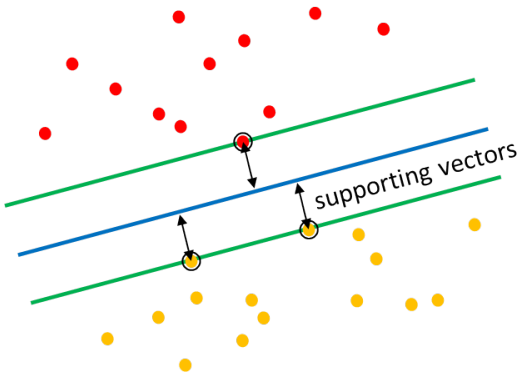


Figure 15: Maximizing the minimum margin leads to a decision boundary that is determined by a subset of at least two data points, known as supporting vectors, which are indicated by the circles.

- For non-linearly separable classes, SVM solves the problem

$$\underset{\mathbf{w},w_0,\mathbf{z}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} z_n \tag{35}$$

$$\text{s.t.} \quad t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_n}) + w_0) \geq 1 - z_n \tag{36}$$

$$z_n \geq 0 \text{ for } n{=}1, ..., N \tag{37}$$

  where C is a positive constant and z_n is a slack variable for the nth training value.

- If the data set is linearly separable, we have $z_n = 0$ for all n as the optimal solution. Instead, points for which $0 < z_n < 1$ are correctly classified but inside the margin and those data points for which $z_n > 1$ are incorrectly classified as illustrated in Fig. 16.

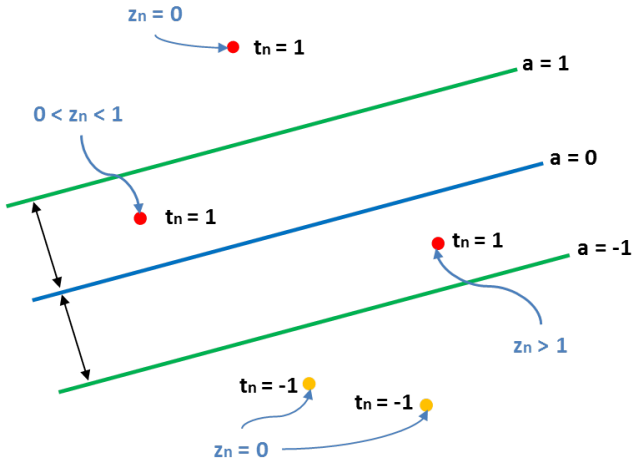


Figure 16: Illustration of the significance of the slack variables z_n in SVM.

- The problem above is equivalent to the regularized ERM:

$$\underset{\mathbf{w},w_0}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \ell(\tilde{\mathbf{w}}, (\mathbf{x_n}, t_n)) \tag{38}$$

  where the hinge loss is defined as

$$\ell(\tilde{\mathbf{w}}, (\mathbf{x}, t)) = \max(0, 1 - t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_o)). \tag{39}$$

- The hinge loss is a convex surrogate of the 0-1-loss as illustrated in Fig. 18.
- C can be set by validation or cross-validation.
- From the discussion above, we can conclude that the restriction to maximum margin classifiers can be thought of as a form of regularization, hence reducing overfitting.

3 Probabilistic Discriminating Models

3.1 Model

- Model $p(C_k|\mathbf{x})$ as a function of parameters $\mathbf{w}$, as well as $p(\mathbf{w})$ for MAP and Bayesian approaches.
- Logistic regression:

$$p(C_1|\mathbf{x}) = p(t = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})) \tag{40}$$

$$\text{and } p(C_0|\boldsymbol{x}) = 1 - \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})) \tag{41}$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{42}$$

is the logistic sigmoid function illustrated in Fig. 15.

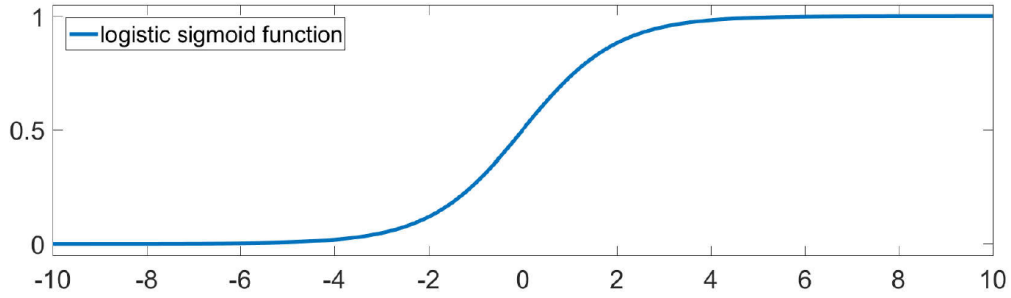


Figure 17: Logistic sigmoid function $\sigma(a)$.

- $\sigma(a)$ can be thought of as a "soft" version of the threshold function $f(a)$ used by the deterministic models studied in the previous section.
- More usefully, we can observe that logistic regression is a Generalized Linear Model (GLM) with

$$t|\mathbf{x}, \mathbf{w} \sim \text{Bern}(t|\eta = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})) \tag{43}$$

with $t \in \{0, 1\}$. We refer to Chapter 2 for discussion on GLMs.

3.2 Learning

- As seen in Chapter 1, for probabilistic models, learning consists of two phases:
  1. *Inference*: Estimate $\mathbf{w}$.
  2. *Optimal decision*: Given a new point $\mathbf{x}$, to minimize the probability of error, the prediction of the class is:

$$p(C_1|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})) = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})) \underset{C_0}{\overset{C_1}{\gtrless}} \frac{1}{2} \tag{44}$$

$$\text{or} \qquad \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) \underset{C_0}{\overset{C_1}{\gtrless}} 0 \tag{45}$$

- Now we focus on inference using ML.

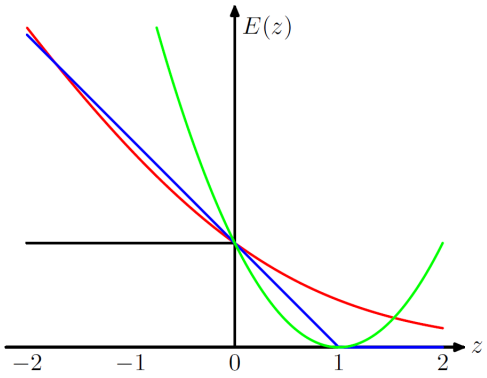


Figure 18: Plot of the 'hinge' error function used in SVM, shown in blue, along with the error function for logistic regression, rescaled by a factor of $\frac{1}{\ln(2)}$ so that it passes through the point $(0, 1)$, shown in red. Also shown are the 0-1 error in black and the squared error in green [Bishop, 2006].

- Likelihood function:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} \underbrace{\sigma(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_n}))}_{\hat{=} y_n}{}^{t_n} (1 - \underbrace{\sigma(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_n}))}_{\hat{=} y_n})^{1-t_n} \tag{46}$$

$$\Rightarrow -\ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln(y_n) + (1 - t_n) \ln(1 - y_n)\} \tag{47}$$

  where equation (47) is the *cross entropy*.

- The ML problem is convex:
  - can be solved via standard solvers, such as CVX;
  - or by using iterative methods such as a gradient or Newton (the latter yields iterative reweighed least square algorithm ([Bishop, 2006, p. 207]).

- The expression of the gradient follows directly from the general formula seen in Chapter 2 for exponential models and by the chain rule for derivatives:

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = (t_n - y_n) \boldsymbol{\phi}(\mathbf{x_n}) \tag{48}$$

- A gradient method for MAP can be implemented in a similar way.

- The Bayesian approach, instead, is generally intractable due to the difficulty in normalizing the posterior

$$p(\mathbf{w}|\mathbf{X}, \mathbf{t}) \propto p(\mathbf{w}) \prod_{n=1}^{N} p(t_n|\mathbf{x_n}, \mathbf{w}) \tag{49}$$

- See [Bishop, 2006, p. 217-220] for an approximate approach based on Laplace approximation.

- As a remark, with $\tilde{t} \in \{-1, +1\}$, the cross-entropy can be written as

$$-\ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^{N} \ln(1 + \exp(\underbrace{-\tilde{t_n} \mathbf{w}^T \phi(\mathbf{x_n})}_{\text{functional margin}})). \tag{50}$$

  This formulation shows that logistic regression can be thought of as an ERM method with a loss $\ell(\mathbf{w}, (\mathbf{x}, t)) = \ln(1 + \exp(-\tilde{t} \mathbf{w}^T \phi(\mathbf{x}))$. The latter is seen in Fig. 18 to be a convex surrogate loss of the 0-1 loss.

4 Probabilistic Generative Models

4.1 Model

- Model:
  - $p(C_k)$ prior distribution over classes
  - $p(\mathbf{x}|C_k)$ class-conditional densities
- Typical choice:
  - $t \sim \text{Bern}(\pi)$
  - $\mathbf{x}|t \sim p(\mathbf{x}|\boldsymbol{\eta}_t)$ Exponential family with natural parameters η_t
  - The two choices above yield the joint distribution for $(\mathbf{x}, t)$

$$p(\mathbf{x}, t|\pi, \eta_0, \eta_1) = p(t|\pi)p(\mathbf{x}|\boldsymbol{\eta_t}).$$

- Generative models typically have more parameters than discriminative models, and they make more assumptions on the model by attempting to learn also the distribution of $\mathbf{x}$. As such, generative models may suffer from both bias and overfitting. However, the capability to capture the properties of the distribution of $\mathbf{x}$ can improve learning if $p(\mathbf{x}|t)$ has significant structure.

4.2 Learning

- As for discriminative models, we have two phases:
  1. *Inference*: Estimate $\boldsymbol{\pi}, \boldsymbol{\eta_0}, \eta_1$.
  2. *Optimal decision*: Given a new point $\mathbf{x}$, to minimize the probability of error, the prediction of the class is:

$$p(C_1|\mathbf{x}) = \frac{\boldsymbol{\pi} p(\mathbf{x}|\eta_1)}{\boldsymbol{\pi} p(\mathbf{x}|\eta_1) + (1-\pi)p(\mathbf{x}|\eta_0)} \underset{C_0}{\overset{C_1}{\gtrless}} \frac{1}{2}. \tag{51}$$

- We now focus on ML inference.

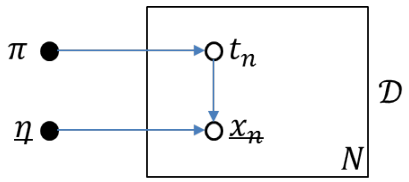


Figure 19: Bayesian network representation for the joint distribution $p(\mathbf{X}, \mathbf{t}|\pi, \boldsymbol{\eta_0}, \boldsymbol{\eta_1})$ in the probabilistic generative model ($\mathcal{D}$=$(\mathbf{X}, \mathbf{t})$ is the dataset).

- Log-likelihood function:

$$\ln p(\mathbf{X}, \mathbf{t}|\pi, \boldsymbol{\eta_0}, \boldsymbol{\eta_1}) = \sum_{n=1}^{N} \ln p(t_n|\pi) + \sum_{\substack{n=1: \\ t_n=0}}^{N} \ln p(\mathbf{x_n}|\boldsymbol{\eta}_0) + \sum_{\substack{n=1: \\ t_n=1}}^{N} \ln p(\mathbf{x_n}|\boldsymbol{\eta}_1). \tag{52}$$

- A graphical representation of the joint distribution of the dataset $(\mathbf{X}, \mathbf{t})$ via a Bayesian network is shown in Fig. 19.
- Given the decomposition of the log-likelihood in (52), we con optimize over π, $\boldsymbol{\eta}_0$ and $\boldsymbol{\eta}_0$ via separate ML estimates.
- As an example, we consider *Quadratic Discriminant Analysis* (QDA), which chooses the class-dependent distributions as

$$\mathbf{x}|C_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{53}$$

- From the general rules derived in Chapter 2 for the exponential family, the ML estimates are given as:

$$\pi_{ML} = \frac{N[1]}{N} \tag{54}$$

$$\boldsymbol{\mu}_{k,ML} = \frac{1}{N[k]} \sum_{\substack{n=1: \\ t_n=k}}^{N} \mathbf{x}_n \tag{55}$$

$$\boldsymbol{\Sigma}_{k,ML} = \frac{1}{N[k]} \sum_{\substack{n=1: \\ t_n=k}}^{N} (\mathbf{x_n} - \boldsymbol{\mu}_k)(\mathbf{x_n} - \boldsymbol{\mu}_k)^T \tag{56}$$

- The resulting predictive distribution for the label of a new sample is

$$p(C_1|\mathbf{x}) = \frac{\pi_{ML}\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{1,ML}, \boldsymbol{\Sigma}_{1,ML})}{\pi_{ML}\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{1,ML}, \boldsymbol{\Sigma}_{1,ML}) + (1-\pi_{ML})\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{0,ML}, \boldsymbol{\Sigma}_{0,ML})}. \tag{57}$$

- Setting $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$ for $k = 1, 2$, which is an example of *parameter tying or sharing*, yields Linear Discriminant Analysis (LDA).

- Under the assumption of decoupled priors, MAP and Bayesian approaches can be directly derived by using the same analysis and models discussed in Chapter 2.

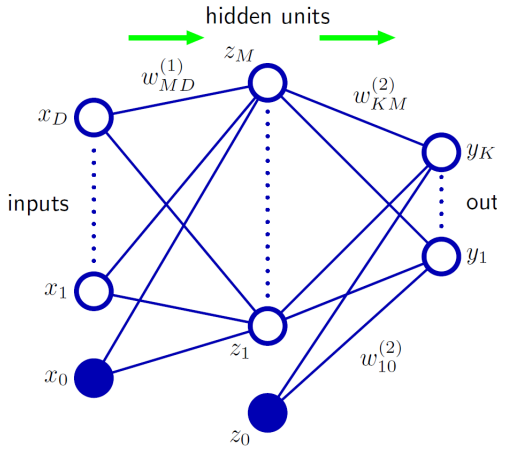


Figure 20: Network diagram for a two-layer neural network. The input, hidden, and output variables are represented by nodes, and the weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links coming from additional input and hidden variables x_0 and z_0 . Arrows denote the direction of information flow through the network during forward propagation [Bishop, 2006].

5 Extensions

5.1 Multi-class Classification

- Two general strategies can be used to combine binary classifiers for multiclass classification: *one-versus-one* and *one-versus-the-rest* classifier [Bishop, 2006].

- The architecture of a one-versus-the-rest solution is shown in Fig. 21 for a deterministic model. Feature-based models can be similarly defined following Fig. 6.

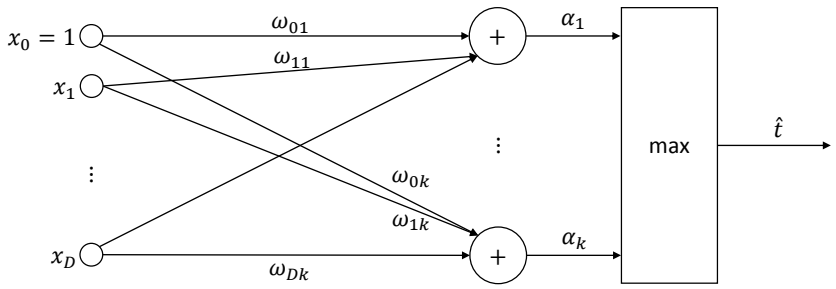


Figure 21: Linear deterministic model for a multi-class classifier.

- An illustration of the outcome of learning for a multi-class classifier for digit classification can be seen in Fig. 22. The image is the input to the classifier, where the dimension D equals the number of pixels. For each class k, the figure shows the learned weights $\mathbf{w}$. Note that there is one weight for each pixel and hence the weight vector can be represented as an image.

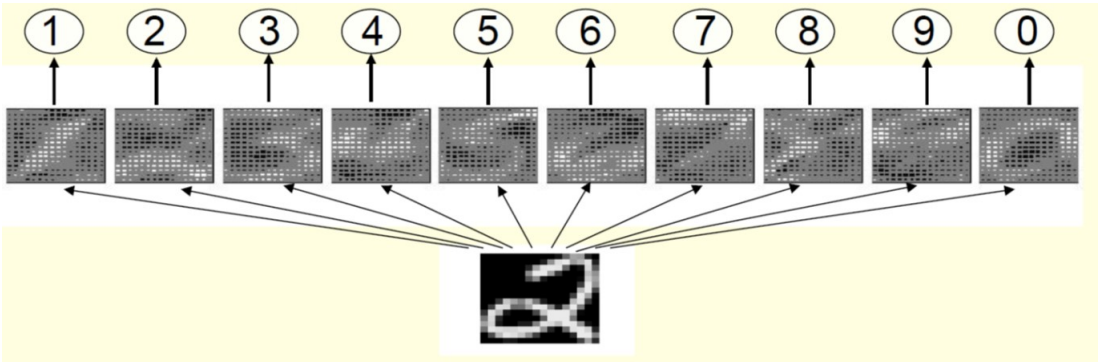


Figure 22: Learned weights for a multi-class classifier as in Fig. 21 trained for character recognition (@2016 Hinton).

5.2 Non-linear Classification

- Feed-forward multi-layer neural networks are non-linear generalizations of the two-layer models $y(\mathbf{x}, \mathbf{w})$ or $p(t|\mathbf{x}, \mathbf{w})$ used in linear models (or three-layer models, with fixed relationship between first and second layers as in Fig. 6), which admit efficient gradient based learning algorithms.
- As illustrated in Fig. 23, feed-forward multilayer neural networks are characterized by:
  - hidden units with non-linear activation functions

$$z_j = h(a_j), \tag{58}$$

    where function h is differentiable and non-linear and a_j the activation of the jth hidden unit. Examples are

$$h(a) = \sigma(a)$$
$$h(a) = \tanh(a)$$
$$h(a) = \max(0, a);$$

  - output units defined as for the two-layer networks

$$y_k = f(a_k) \tag{59}$$

where a_k is the activation of the kth output unit. Examples are

$$f(a) = \sigma(a)$$

$$f(a) = \text{sign}(a).$$

- We can think of the intermediate layers as learning the features that are used by the output layer to perform classification. Multilayer networks are hence particularly useful when good features are hard to derive.

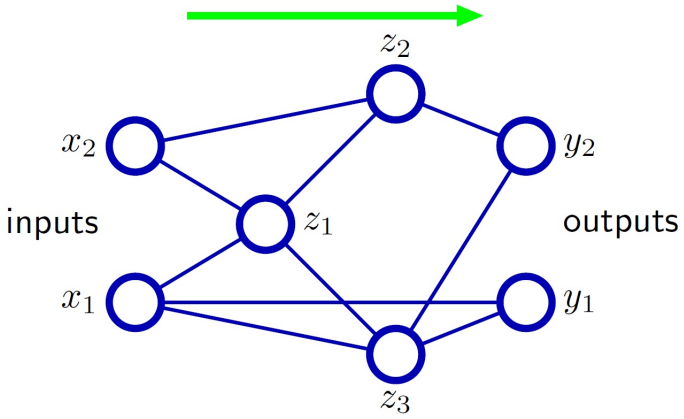


Figure 23: Example of a multi-layer feed-forward neural network [Bishop, 2006].

- There are many variants of neuronal networks such as:
  - Deep neural networks (Fig. 24);

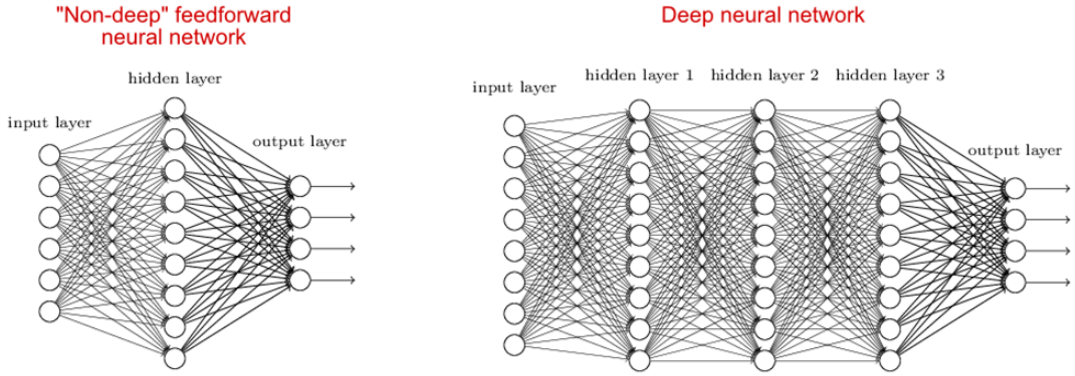


Figure 24: Conventional vs. deep neural networks [Olah, 2014b].

  - Recursive neuronal networks (Fig. 25);

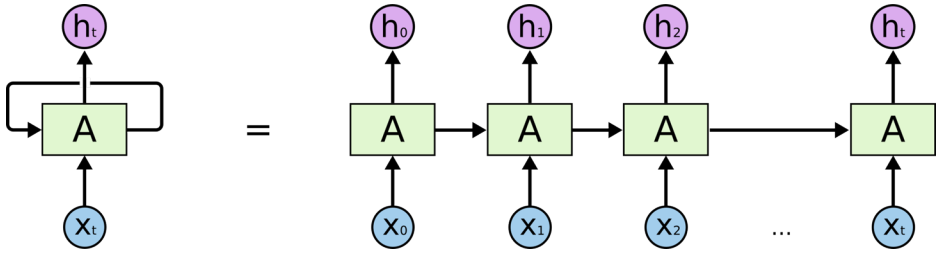


Figure 25: Recursive neural networks [Olah, 2015].

– Convolutional neuronal networks (Fig. 26);

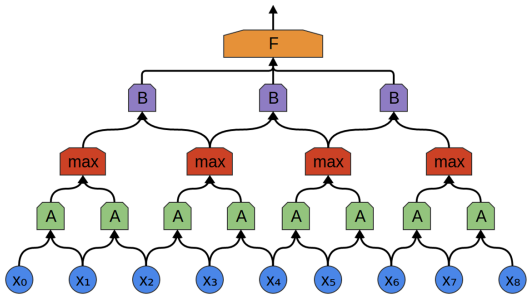


Figure 26: Convolutional neuronal network [Olah, 2014a].

– Spiking neuronal networks.

⋮

6 Problems

1. In this problem, we implement and evaluate the performance of the considered binary classification schemes. We will do so on the dataset `datasetiris`, which presents $N = 80$ training examples divided into 40 examples corresponding to a type of iris flowers called setosa (`setosa_tr`) and 40 examples corresponding to a different type known as virginica (`virginica_tr`). The two variables measured in each example are the sepal length (first variable) and sepal width (second variable). Note that $D = 2$ and $N = 80$. There are also $N_{test} = 40$ test examples, half for setosa and half for virginica families (`setosa_test` and `virginica_test`).

a. Make a scatterplot of the training data set. Use different colors for the two classes. (Hint: You can use `scatter(setosa_tr(1,:),setosa_tr(2,:)) hold on; scatter(virginica_tr(1,:),virginica_tr(2,:),'r'); xlabel('sepal length'); ylabel('sepal width').`)

b. Defining the setosa type as class 0, which is encoded as $t = -1$ and the virginica type as class 1, which is encoded as $t = 1$, find the separating hyperplane obtained via the least squares solution. (Hint: Format the training data as `X=[ones(N,1) [setosa_tr';virginica_tr']]; t=[-ones(N/2,1);ones(N/2,1)];` and then compute `w=pinv(X)*t.`)

c. Plot the separating hyperplane obtained at the previous point and compute the corresponding test misclassification error, i.e., the fraction of samples in the test set that are incorrectly classified.

d. Implement now the perceptron algorithm. Perform N iterations by selecting one training sample uniformly at random from the training set. Use `w=[-5.5 1 0.01]'` as the initial solution. (Hint: You can use the function `randperm`.) Plot the obtained separating hyperplane and compute the test misclassification error. Try running the code again: you will obtain a different hyperplane due to the randomization of the order in which the samples are used for training.

e. (Optional) Implement SVM by using cvx (http://cvxr.com/cvx/). You can set $C = 1$. Plot the obtained separating hyperplane and compute the test misclassification error.

f. Implement logistic regression via the stochastic gradient method. Set the learning rate as η=1; the same initialization as for the perceptron algorithm; and select samples at each iteration randomly as discussed above. Plot the contour lines of the predictive distribution (Hint: You can use `xaxis=[4:0.01:8.5]; yaxis=[0.5:0.01:4.5]; [Xaxis,Yaxis]=meshgrid(xaxis,yaxis); Plog=log(1./(1+exp(-(w(1)+w(2)*Xaxis+w(3)*Yaxis)))); contour(Xaxis,Yaxis,Plog,[-100:1:0]);`). Evaluate the test misclassification error.

g. Implement Quadratic Discriminant Analysis. Plot the contour lines of the two class-conditioned distributions and evaluate the error.

2. We consider the following generative model for binary classification

$$p(t|\pi) = \text{Bern}(t|\pi)$$
$$p(\mathbf{x}|t, p_t) = \prod_{d=1}^{D} \text{Bern}(x_d|p_t)$$

Note that $\mathbf{x} \in \{0, 1\}^D$ is a binary vector of dimension D and that the parameters of the model are π, p_0, p_1.

a. Write the log-likelihood function for a data set $\{(\mathbf{x}_n, t_n)\}$, $n = 1, ..., N$, generated from the model. Use the notation $\mathbf{x}_n = (x_{1n}, ..., x_{Dn})$.

b. Show that the ML estimates of π, p_0, p_1 can be carried out separately and write the ML estimate of these parameters.

c. Assume now that there is an independent Beta prior for π, p_0, p_1. Assume for simplicity that the hyperparameters of the three Beta distributions are the same, namely $a > 1$ and $b > 1$. What is the MAP estimate of the model parameters?

d. Derive the optimal decision t for a new data point $\mathbf{x} =(x_1, ..., x_D)$ as a function of π, p_0, p_1 and $\mathbf{x}$.

References

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin.

[Murphy, 2012] Murphy, K. P. (2012). *Machine Learning*. MIT Press.

[Olah, 2014a] Olah, C. (2014a). *http://colah.github.io/posts/2014-07-Conv-Nets-Modular/*.

[Olah, 2014b] Olah, C. (2014b). *http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/*.

[Olah, 2015] Olah, C. (2015). *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*.

4. Statistical Learning

1 Introduction

- In supervised learning, we wish to learn a model h from a class $\mathcal{H}$ given a dataset

$$\mathcal{D} = \{(x_n, t_n)\}_{n=1}^{N} \underset{i.i.d.}{\sim} p_0(\boldsymbol{x}, t) \tag{1}$$

- The learned model $\mathcal{D} \to h_{\mathcal{D}}$ is random due to randomness of the dataset $\mathcal{D}$.
- Ex: For linear regression, we have $\mathcal{H} = \{t = h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x})\}$, for a given feature vector $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{D'}$, and hence the model is parametrized by a vector $\boldsymbol{w} \in \mathbb{R}^{D'}$, which is learned from $\mathcal{D}$.
- **Goal:** Minimize the generalization error

$$L_p(h) = E_{(\boldsymbol{x},t)\sim p_0(\boldsymbol{x},t)}[\ell(h, (\boldsymbol{x}, t))], \tag{2}$$

obtaining an optimal model $h^* \in \underset{h\in\mathcal{H}}{\operatorname{argmin}} L_p(h)$ without knowing $p_0(\boldsymbol{x}, t)$. Note that there may be multiple optimal hypotheses, but, to fix the ideas, it may be useful to think of the case in which there is a unique optimal hypothesis (which is true when the loss is strictly convex).

- Since $p_0(\boldsymbol{x}, t)$ is unknown, we can only minimize $L_p(h)$ approximately and with some probability, as illustrated in Fig.1.

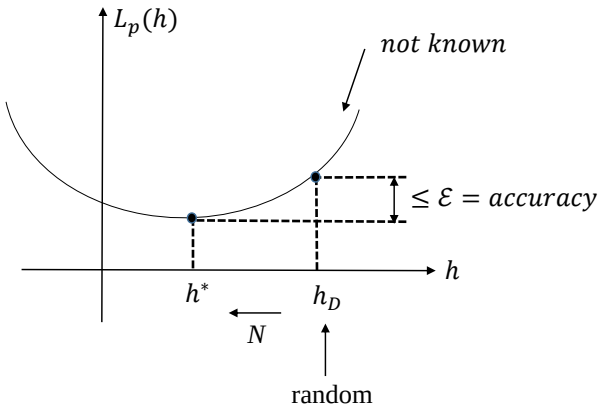


Figure 1: The goal of learning is obtaining the minimum generalization error $L_p(h^*)$, which is, however, unknown. If learning is successful, $L_p(h_D)$ converges to $L_p(h^*)$ when N increases.

- **Goal (restated):** We wish to find a hypothesis $h \in \mathcal{H}$ that is *Probably Approximately Correct (PAC)*, that is

  $L_p(h_\mathcal{D}) \approx L_p(h^*)$ with high probability or, more precisely,

  $L_p(h_\mathcal{D}) \leq L_p(h^*) + \epsilon$ with probability $\geq 1 - \delta$

  where we define ϵ as the accuracy parameter and δ as the confidence parameter.

- The key question is: How large should N be to ensure the existence of a learning scheme $\mathcal{D} \rightarrow h_\mathcal{D}$ such that $h_\mathcal{D}$ is $\epsilon-$ accurate with probability $1 - \delta$? See Fig. 2 for an illustration.

- We know that a large $|\mathcal{H}|$ implies the need for a larger N to avoid overfitting. In this chapter, we study the relationship between the "size" of $\mathcal{H}$ and N.

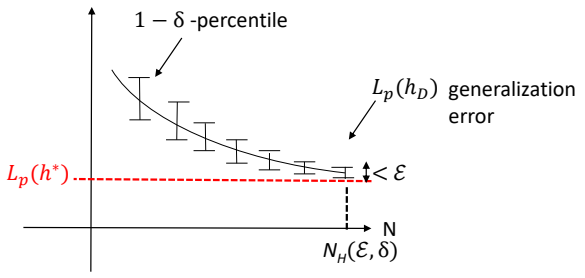


Figure 2: Generalization error $L_p(h_\mathcal{D})$ versus the number N of data points for a PAC learnable class $\mathcal{H}$.

2 Empirical Risk Minimization (ERM)

- In this chapter, we focus on deterministic discriminative models (or hypotheses) $t = h(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^D$ and $h \in \mathcal{H}$, where $\mathcal{H}$ is the model (or hypothesis) class.

- As we have seen in the previous chapters, learning is typically done by solving the ERM problem as illustrated in Fig. 3:

$$h_\mathcal{D}^{ERM} = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\, L_\mathcal{D}(h) = \frac{1}{N} \sum_{n=1}^{N} \ell(h, (\boldsymbol{x}_n, t_n)) \tag{3}$$

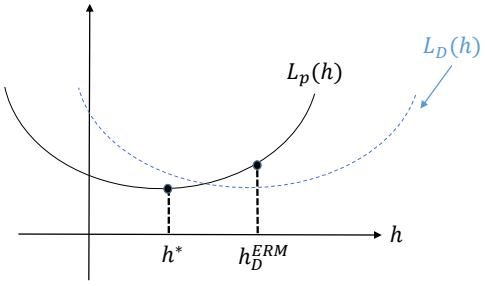


Figure 3: Illustration of the ERM solution $h_\mathcal{D}^{ERM}$.

- Intuitively, if N is large enough, we should have $|L_{\mathcal{D}}(h) - L_p(h)| \approx 0$ for all $h \in \mathcal{H}$ and hence also $L_p(h^*) \approx L_p(h_{\mathcal{D}}^{ERM})$.
- An illustration of the typical behavior of $L_p(h_{\mathcal{D}})$ and $L_{\mathcal{D}}(h_{\mathcal{D}})$ is shown in Fig. 4 (see also Chapter 1).

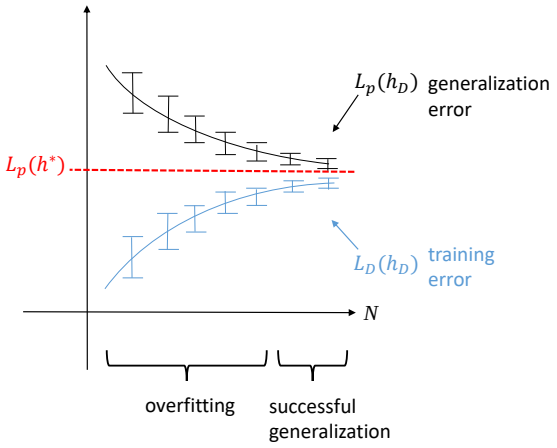


Figure 4: Illustration of the typical behavior of $L_p(h_{\mathcal{D}})$ and $L_{\mathcal{D}}(h_{\mathcal{D}})$.

- As an example, consider the problem of binary classification using threshold functions, which is defined as
  - $x \in \mathbb{R}$ $(D = 1)$
  - $\mathcal{H} = \left\{ h_a(x) = \begin{cases} 0, & \text{if } x < a \\ 1, & \text{if } x \geq a \end{cases} = 1(x \geq a) \right\}$
  - $\ell(h_a, (x, t)) = 1(h_a(x) \neq t)$ 0-1 loss
- Make the *realizability* assumption that $p_0(x, t) = p_0(x)1(t = h_0(x))$ and hence the optimal hypothesis is $h^* = h_0$ or $a^* = 0$.
- The generalization error and the training error with $p_0(x) = \mathcal{U}([-1, 1])$ are shown in Fig. 5.

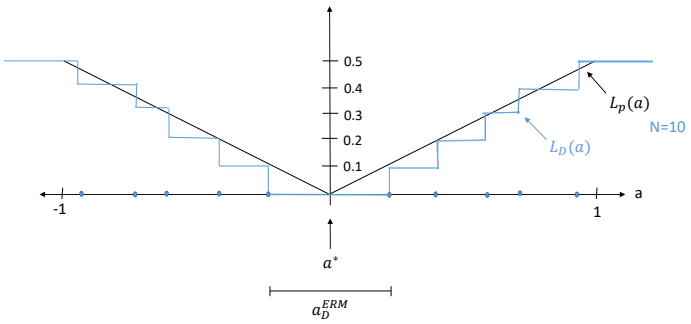


Figure 5: Generalization error $L_p(a)$ and training error $L_{\mathcal{D}}(a)$ for binary classification using threshold functions. The training error is computed from the data $\mathcal{D}$ indicated by the dots on the horizontal axis.

3 PAC Learnability and Sample Complexity

- **Definition (PAC learnability):** A hypothesis class $\mathcal{H}$ is PAC learnable under a loss function $\ell(h,(\boldsymbol{x},t))$ if there exists a function $N_{\mathcal{H}}(\epsilon,\delta)<\infty$ and a learning algorithm $\mathcal{D}\rightarrow h_{\mathcal{D}}\in\mathcal{H}$ such that, for any $\epsilon,\delta\in(0,1)$ and for every $p_0(\boldsymbol{x},t)$, we have

$$\Pr_{\mathcal{D}\underset{i.i.d.}{\sim}p_0(\boldsymbol{x},t)}(L_p(h_{\mathcal{D}})\leq L_p(h^*)+\epsilon)\geq 1-\delta \quad \text{for } N\geq N_{\mathcal{H}}(\epsilon,\delta). \tag{4}$$

- Realizability assumption: A less strong definition of PAC learnability requires (4) to hold only for every $p_0(\boldsymbol{x},t)$ such that

$$p_0(\boldsymbol{x},t)=p_0(\boldsymbol{x})1(t=h_0(\boldsymbol{x})) \quad \text{with } h_0\in\mathcal{H} \tag{5}$$

  under a loss function ℓ for which $L_p(h_0)=0$. In other words, under the realizability assumption, the data is generated from some mechanism that is included in the hypothesis class.

- Not all classes $\mathcal{H}$ are PAC learnable!

- **Examples:**
  - 1) The class $\mathcal{H}=\{\textbf{set of all functions } h:\ \mathbb{R}^{\mathbf{D}}\rightarrow\{0,1\}\}$ under $0-1$ loss is not PAC learnable. This is also known as the *no free lunch theorem*. Given any amount of data, we can always find a distribution $p_0(\boldsymbol{x},t)$ under which the PAC condition are not satisfied. Intuitively, even in the realizable case, knowing $t=h^*(\boldsymbol{x})$ for any number of $\boldsymbol{x}\in\mathbb{R}^{\mathrm{D}}$ yields us no information on the value of h_0 for other values of $\boldsymbol{x}$.
  - 2) The class $\mathcal{H}=\{h_w(x)=1(sin(wx)>0)\},\ x\in\mathbb{R}$ under $0-1$ loss is not PAC learnable (see Fig. 6) [2].

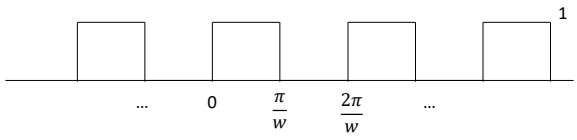


Figure 6: The class $\mathcal{H}=\{h_w(x)=1(sin(wx)>0)\}$ is not PAC learnable.

- **Definition (Sample Complexity):** The sample complexity $N^*_{\mathcal{H}}(\epsilon,\delta)$ is the minimal value of $N_{\mathcal{H}}(\epsilon,\delta)$ that satisfies the requirements of PAC learning for class $\mathcal{H}$.

- The sample complexity depends on the "size" of $\mathcal{H}$.

- The sample complexity of example 1) and 2) above is $N^*=\infty$ since they are not PAC learnable.

- PAC learnability can also be defined under the additional condition that $N^*_{\mathcal{H}}(\epsilon,\delta)$ be polynomial in $1/\epsilon$, $1/\delta$, $\mathcal{D}$ and "size" of $\mathcal{H}$ (to be discussed).

- A class $\mathcal{H}$ is efficiently PAC learnable if the learning algorithm in the definition has polynomial complexity in the same variables discussed above.

4 PAC Learnability for Finite Hypothesis Classes

- **Theorem:** If $\mathcal{H}$ is a finite hypothesis class and the loss function is bounded, w.l.o.g. $\ell(h,(\boldsymbol{x},t))\leq 1$, then $\mathcal{H}$ is PAC-learnable under ℓ with sample complexity

$$N^*_{\mathcal{H}}(\epsilon,\delta)\leq\left\lceil\frac{2\ln(2|\mathcal{H}|)/\delta}{\epsilon^2}\right\rceil\triangleq N_H^{ERM}(\epsilon,\delta) \tag{6}$$

  Moreover, the ERM algorithm achieves the upper bound $N_{\mathcal{H}}^{ERM}(\epsilon,\delta)$.

- The upper bound $N_{\mathcal{H}}^{ERM}(\epsilon, \delta)$ in (6) depends on
  - the "capacity" of the hypothesis class $\ln |\mathcal{H}|$ (nats) or $\log_2 |\mathcal{H}|$ (bits): number of bits required to index the hypothesis in $\mathcal{H}$;
  - the logarithm $\ln(1/\delta)$ of the confidence parameter;
  - the inverse square $1/\epsilon^2$ of the accuracy parameter.
- The theorem applies only to finite hypothesis classes. But can infinite classes be learned? For example, can we learn a binary linear classifier $\mathcal{H} = \left\{h_{\tilde{\boldsymbol{w}}}(\mathbf{x}) = 1(\boldsymbol{w}^T\mathbf{x} + w_0 > 0)\right\}$, where $\tilde{\boldsymbol{w}} \in \mathbb{R}^{D+1}$ under the $0-1$ loss? One approach is to try to learn a "quantized" version of $\mathcal{H}$, where each weight is represented by b bits (e.g., $b = 16$). Then, an upper bound on the sample complexity is

$$N_{\mathcal{H}}^{ERM}(\epsilon, \delta) = \left\lceil \frac{2\ln(2^{b(D+1)}/\delta)}{\epsilon^2} \right\rceil = \left\lceil \frac{2b(D+1)\ln 2 + 2\ln(1/\delta)}{\epsilon^2} \right\rceil, \tag{7}$$

  which scale proportionally to the number of parameters $D+1$ and to the resolution b. This is known as the *quantization trick*.
- The assumption of boundedness of ℓ is important. For example, the problem of linear regression with squared loss is generally not PAC learnable (see [2, p. 132]). See also [1, Ch. 4] for extensions.
- From the theorem, we have that if $N \geq \frac{2\ln(2|\mathcal{H}|/\delta)}{\epsilon^2}$, then we can achieve the accuracy and confidence levels (ϵ, δ) with ERM. This implies that, conversely, with N data points, we can achieve the precision

$$\epsilon = \sqrt{\frac{2\ln(2|\mathcal{H}|/\delta)}{N}} \propto \frac{1}{\sqrt{N}}, \text{ with probability } 1-\delta, \tag{8}$$

  that is, with N data points, we can upper bound the generalization error as

$$L_p(h_{\mathcal{D}}^{ERM}) \leq L_p(h^*) + \sqrt{\frac{2\ln|\mathcal{H}|/\delta}{N}} \tag{9}$$

  with probability $1-\delta$.
- It can be inferred from (7) that increasing N brings the generalization error closer to the minimum generalization error for finite hypothesis classes.
- Under the realizability assumption, the theorem can be modified to yield the upper bound

$$N_{\mathcal{H}}^{*}(\epsilon, \delta) \leq \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil = N_{\mathcal{H}}^{ERM}(\epsilon, \delta). \tag{10}$$

4.1 Proof of Theorem

- **Lemma:** For any $N \geq N_{\mathcal{H}}^{ERM}(\epsilon, \delta)$, we have

$$\Pr_{\mathcal{D} \underset{i.i.d.}{\sim} p_0(\boldsymbol{x},t)} \left[|L_p(h) - L_{\mathcal{D}}(h)| \leq \frac{\epsilon}{2} \; \forall h \in \mathcal{H} \right] \geq 1 - \delta \tag{11}$$

- This lemma says that $L_{\mathcal{D}}(h)$ is a uniformly accurate estimate of $L_p(h)$ with high probability. The proof is given below.

- Assuming that the Lemma is true, then the Theorem follows because

$$| L_p(h) - L_{\mathcal{D}}(h) | \leq \frac{\epsilon}{2} \quad \forall h \in \mathcal{H} \text{ with probability} \geq 1-\delta$$

$$\implies L_p(h_{\mathcal{D}}^{ERM}) \underset{\substack{\uparrow \\ \text{by Lemma}}}{\leq} L_{\mathcal{D}}(h_{\mathcal{D}}^{ERM}) + \frac{\epsilon}{2} \underset{\substack{\uparrow \\ \text{by defn. of ERM}}}{\leq} L_{\mathcal{D}}(h^*) + \frac{\epsilon}{2}$$

$$\underset{\substack{\uparrow \\ \text{by Lemma}}}{\leq} L_p(h^*) + \underbrace{\frac{\epsilon}{2} + \frac{\epsilon}{2}}_{\epsilon} = L_p(h^*) + \epsilon,$$

  which concludes the proof.

- Now, we only need to prove the Lemma. To do that, we will use Hoeffding's inequality.

- Hoeffding's Inequality: For $\theta_1, \theta_2, \cdots, \theta_M \underset{i.i.d.}{\sim} p(\theta)$ **such that** $E[\theta_i] = \mu$ and $\Pr(a \leq \theta_i \leq b) = 1$,

$$\Pr\left[\left|\frac{1}{M}\sum_{m=1}^{M}\theta_m - \mu\right| > \epsilon\right] \leq 2\exp\left(-\frac{2M\epsilon^2}{(b-a)^2}\right). \tag{12}$$

- **Proof of Lemma:** We have

$$\begin{aligned}
&\Pr_{\mathcal{D} \underset{i.i.d.}{\sim} p_0(\boldsymbol{x},t)}\left[\exists h \in \mathcal{H} : |L_p(h) - L_{\mathcal{D}}(h)| > \frac{\epsilon}{2}\right] \leq \delta \\
&\quad = \Pr_{\mathcal{D} \underset{i.i.d.}{\sim} p_0(\boldsymbol{x},t)}\left[\bigcup_{h\in\mathcal{H}}\left\{|L_p(h) - L_{\mathcal{D}}(h)| > \frac{\epsilon}{2}\right\}\right] \\
&\quad \underset{\substack{\uparrow \\ \text{union bound}}}{\leq} \sum_{h\in\mathcal{H}} \Pr_{\mathcal{D} \underset{i.i.d.}{\sim} p_0(\boldsymbol{x},t)}\left[\left|\underbrace{L_p(h)}_{E[\ell(h,(\boldsymbol{x},t))]} - \underbrace{L_{\mathcal{D}}(h)}_{\frac{1}{N}\sum_{h=1}^{N}\ell(h,(\boldsymbol{x_n},t_n))}\right| > \frac{\epsilon}{2}\right] \\
&\quad \underset{\substack{\uparrow \\ \text{by Hoeffding}}}{\leq} 2\sum_{h\in\mathcal{H}} \exp\left(-\frac{N\epsilon^2}{2}\right) = 2|\mathcal{H}|\exp\left(-\frac{N\epsilon^2}{2}\right) \\
&\quad \leq \delta \quad \text{if } N \geq \left\lceil \frac{2\ln(2|\mathcal{H}|/\delta)}{\epsilon^2} \right\rceil = N_{\mathcal{H}}^{ERM}(\epsilon,\delta).
\end{aligned}$$

5 Choosing a Model Class Using PAC Learning Theory

- A smaller class leads to a smaller sample complexity, but is it always a good choice to choose a small class? As we have seen in the previous chapter, the choice depends on a trade-off between richness of the model and the estimation error, where the latter can be decreased by increasing N as quantified by PAC learning theory.

- To make this precise, we can write the generalization error of the learned hypothesis $h_{\mathcal{D}}^{ERM}$ via ERM (or any other learning scheme) as

$$L_p(h_{\mathcal{D}}^{ERM}) = L_p(h^*) + \left(L_p(h_{\mathcal{D}}^{ERM}) - L_p(h^*)\right),$$

  where:

  - the minimum generalization error $L_p(h^*)$ generally decreases with $|\mathcal{H}|$
  - the estimation error $\left(L_p(h_D^{ERM}) - L_p(h^*)\right)$ is upper bounded by $\sqrt{\frac{2\ln(2|H|/\delta)}{N}}$ and it generally increases with $|\mathcal{H}|$ and decreases with N.

- In the previous chapters, we have seen that the estimation error can be evaluated via validation or cross-validation. Structural Risk Minimization (SRM) is an alternative approach based on the direct minimization of an upper bound on the generalization error evaluated only on the data set $\mathcal{D}$ (and not on test data). Specifically, from the Lemma in the previous section, we have the following bound

$$L_p(h) \leq L_{\mathcal{D}}(h) + \frac{\epsilon}{2} = L_{\mathcal{D}}(h) + \sqrt{\frac{\ln(2|\mathcal{H}|/\delta)}{2N}} \qquad \forall h \in \mathcal{H} \text{ with probability } 1-\delta. \tag{13}$$

- SRM minimizes the upper bound on $L_p(h)$ given in (13), which is a pessimistic estimate of the generalization error, over both the model order and the hypothesis class. The approach hence enables joint model selection and inference for nested hypothesis classes.
- More practical variants of SRM include AIC and MDL.

6 VC Dimension and Fundamental Theorem of PAC Learning

- We have seen that finite classes are PAC learnable with sample complexity $\propto \ln|\mathcal{H}|$ by using ERM.
- Is this the smallest sample complexity? What about infinite hypothesis classes?
- As an example, consider the problem of binary classification on the real line using threshold functions, which is characterized as $x \in \mathbb{R}$, $\mathcal{H} = \{h_a(x) = 1(x \geq a)\}$, and $\ell(h_a(x,t)) = 1(h_a(x) = t)\}$ $0-1$ loss. If we use the quantization trick, the sample complexity under the realizability assumption is bounded as

$$N^*_{\mathcal{H}}(\epsilon,\delta) \leq N^{ERM}_{\mathcal{H}}(\epsilon,\delta) = \left\lceil \frac{b\ln(2/\delta)}{\epsilon} \right\rceil, \tag{14}$$

  where b is the number of bits used to quantize a. However, in Appendix A, we show that, under the realizability assumption, we have

$$N^{ERM}_{\mathcal{H}}(\epsilon,\delta) \leq \left\lceil \frac{\ln(2/\delta)}{\epsilon} \right\rceil \ll \left\lceil \frac{b\ln(2/\delta)}{\epsilon} \right\rceil \tag{15}$$

  as if the class $\mathcal{H}$ had "capacity" $\log_2|\mathcal{H}| = 1$ bit. Note that the "capacity" for this model equals the number of parameters defining the hypothesis class.
- In the previous example, the "capacity" of $\mathcal{H}$ was 1 bit. How to generalize this concept?
- We focus on binary classification $t \in \{0,1\}$ with $0-1$ loss.
- **Definition:** A hypothesis class $\mathcal{H}$ is said to shatter a training set $\mathcal{X} = \{x_n\}_{n=1}^N$ if no matter how the labels $\{t_n \in \{0,1\}\}_{n=1}^N$ are selected, there exist a hypothesis $h \in \mathcal{H}$ that perfectly separates them (i.e., $h(\boldsymbol{x}_n) = t_n$ for all $n = 1,\ldots,N$).
- **Definition (Vapnik-Chervonenkis, or VC, Dimension)**: VCdim($\mathcal{H}$) is the size of the largest training set $\mathcal{X}$ that is shattered by $\mathcal{H}$.
- To prove that VCdim($\mathcal{H}$) $= N$, we need
  - to demonstrate the existence of $\mathcal{X}$ with $|\mathcal{X}| = N$, that is shattered by $\mathcal{H}$ (easier);
  - prove that no set $\mathcal{X}$ of dimension $N+1$ exists that is shattered by $\mathcal{H}$ (more difficult).
- We will see that the VC dimension is the right definition of "capacity" of a hypothesis class $\mathcal{H}$ (measured in bits).
- **Examples:**

- **a)** Threshold functions: $\text{VCdim}(\mathcal{H}) = 1$ since there exists no $\mathcal{X}$ with $|\mathcal{X}| = 2$ that is shattered by $\mathcal{H}$, see Fig. 7. Note that the VC dimension is equal to the number of parameters.
- **b)** Intervals $\mathcal{H} = \{h_{a,b}(x) = 1\,(a \le x \le b) \text{ with } a \le b\}$: $\text{VCdim}(\mathcal{H}) = 2$, see Fig. 8. Again, the VC dimension equals the number of parameters.
- **c)** Axis aligned rectangles $\mathcal{H} = \left\{h_{(a_1,a_2,b_1,b_2)}(\boldsymbol{x}) = 1\,(a_1 \le x_1 \le a_2 \text{ and } b_1 \le x_2 \le b_2) \text{ with } a \le b\right\}$ (see Fig. 9): $\text{VCdim}(\mathcal{H}) = 4$, as illustrated in Fig. 10 and 11. The VC dimension equal the number of parameters.
- **d)** Hyperplanes $\mathcal{H} = \left\{h_{\tilde{\boldsymbol{w}}}(\boldsymbol{x}) = 1\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + w_o > 0\right)\right\}$ $(x \in \mathbb{R})$: $\text{VCdim}(\mathcal{H}) = D + 1$, see Fig. 12-13.
- **e)** Finite classes: $\text{VCdim}(\mathcal{H}) \le \log_2 |\mathcal{H}|$ since $|\mathcal{H}|$ hypothesis can create at most $2^{|\mathcal{H}|}$ different label configurations.
- **f)** There exist hypothesis classes with a finite number of parameters but infinite VCdim, such as

$$\mathcal{H} = \{h_w(x) = 1(\sin(wx) > 0)\} \tag{16}$$

- **g)** Class of hypotheses with margin $\ge m$: VCdim $\propto \frac{1}{m^2}$.

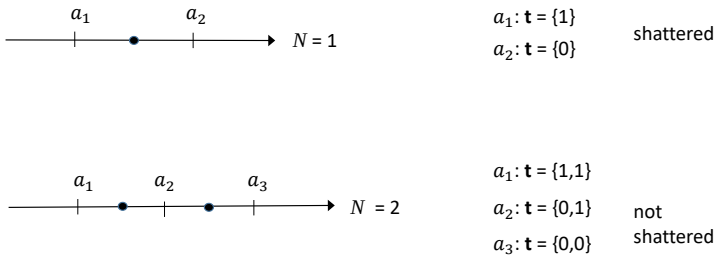


Figure 7: Example a.

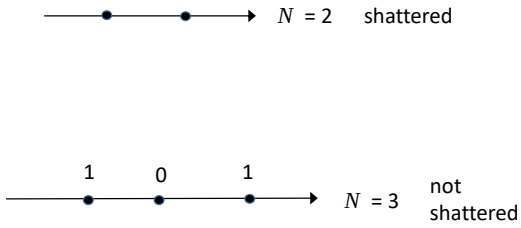


Figure 8: Example b.

- **Theorem:** (**Fundamental Theorem of PAC Learning**) For a binary classification problem with $0-1$ loss, a class $\mathcal{H}$ with $\text{VCdim}(\mathcal{H}) = d < \infty$ is PAC learnable with sample complexity

$$C_1 \frac{d + \ln(1/\delta)}{\epsilon^2} \le N^*_{\mathcal{H}}(\epsilon, \delta) \le C_2 \frac{d + \ln(1/\delta)}{\epsilon^2} \tag{17}$$

for some (absolute) constants C_1 and C_2. Moreover, the ERM algorithm achieves the upper bound above.

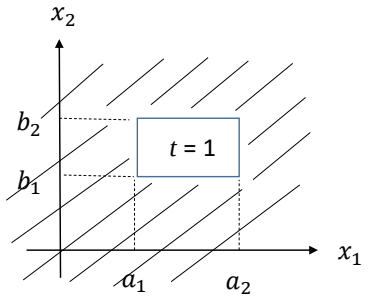


Figure 9: Example c: Axis aligned rectangles.

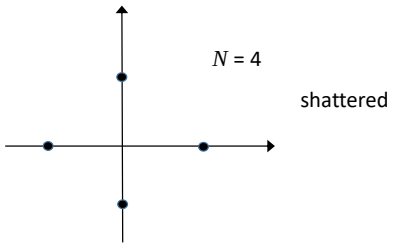


Figure 10: Example c: This training set with $N = 4$ is shattered.

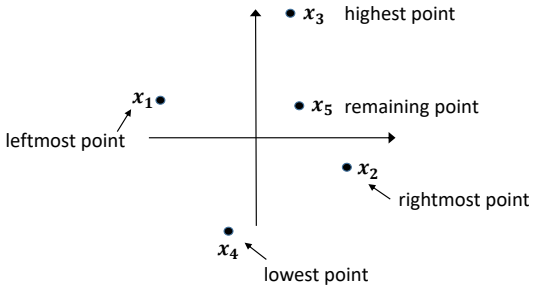


Figure 11: Example c: There is no training set with $N = 5$ that is shattered. The labeling (1,1,1,1,0) is not realizable since the rectangle must contain x_1, x_2, x_3, x_4 and hence also x_5.

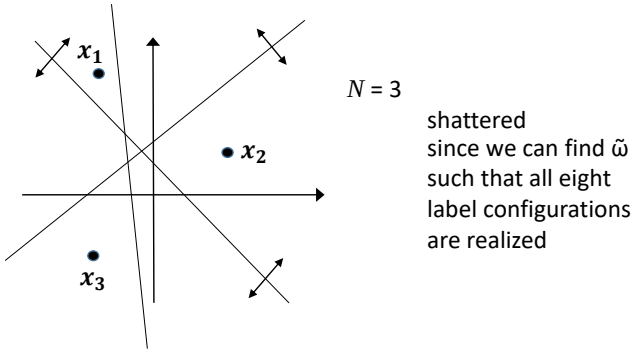


Figure 12: Example d.

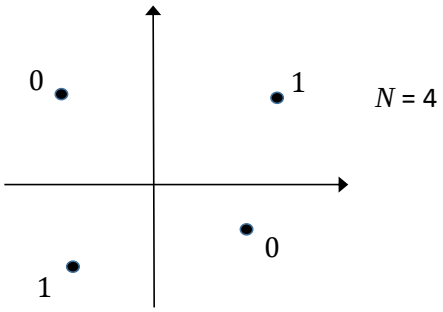


Figure 13: Example d.

- The theorem allows us to conclude that the VC dimension is the "right" definition for the "capacity" of hypothesis class $\mathcal{H}$.
- Extensions exist for other learning tasks (loss functions) (see [2, p.48]).

Proof: See [2].

7 Problems

1. Consider the class of binary classifiers defined on a finite set $\mathcal{X}$

$$\mathcal{H} = \{h_z(x) = 1(x = z) \text{ for all } z \in \mathcal{X}\} \bigcup \{h_\emptyset(x) = 0\} \tag{18}$$

and assume the standard 0-1 misclassification loss.

a. Write down the ERM problem. Derive the ERM predictor given a sample $\mathcal{D}$ of N training pairs (x, t) from an unknown true distribution $p_0(x, t) = p_0(x)1(t = h(x))$ with $h(x) \in \mathcal{H}$. Note that $p_0(x)$ and $h(x)$ are both unknown and that the given restriction on the true distribution $p_0(x, t)$ implies that we are in the realizable case.

b. Is this class PAC learnable? If so, obtain an upper bound on the sample complexity.

2. Calculate the VC dimension of the hypothesis class $\mathcal{H} = \{h_r(\mathbf{x}) = 1(||\mathbf{x}|| \leq r) \text{ for } r \geq 0\}$ for $\mathbf{x} \in \mathbb{R}^2$.

3. Compute the VC dimension of the following hypothesis class: $\mathcal{H} = \{h_{a,b,s}(x) \in \{0, 1\} : \ a \leq b \text{ and } s \in \{0, 1\}\}$, where $h_{a,b,s}(x) = s$ if $x \in [a, b]$ and $h_{a,b,s}(x) = 1 - s$ if $x \notin [a, b]$.

References

[1] J. Duchi, Lecture notes, http://stanford.edu/class/stats311/Lectures/full-notes.pdf

[2] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: from theory to algorithm*, Cambridge University Press, 2014.

8 Appendix A: Proof of (15)

- By the realizability assumption, we have (see Fig. 14)

$$p_0(x, t) = p_0(x)1(t = 1(x \geq a^*)) \tag{19}$$

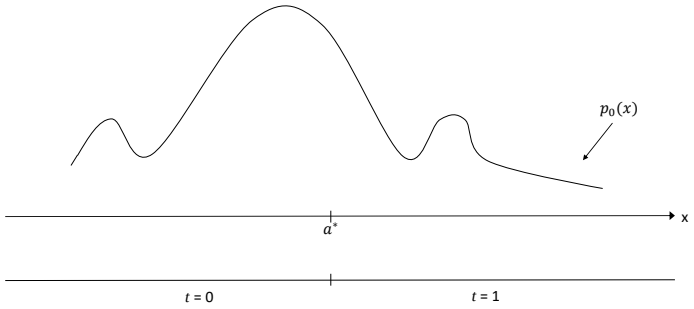


Figure 14: Illustration of a possible data distribution for the example studied in Appendix A.

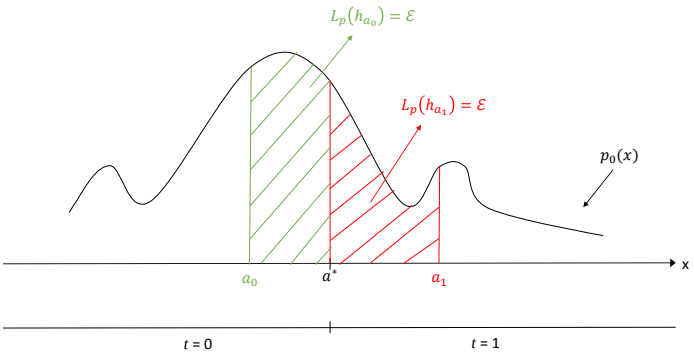


Figure 15: Illustration of a_0 and a_1.

- Define $a_1 = \sup\{a:\ L_p(h_a) \le \epsilon\}$ and $a_0 = \inf\{a : L_p(h_a) \le \epsilon\}$ (see Fig. 15).

- Referring to Fig. 16 and 17, we have

$$\begin{aligned}
&\Pr_{\boldsymbol{x} \underset{i.i.d.}{\sim} p_0(x)}\left[L_p(a_{\mathcal{D}}^{ERM}) \le \epsilon\right] \ge \Pr_{\boldsymbol{x} \underset{i.i.d.}{\sim} p_0(x)}\left[\left\{\bigcup_{n=1}^{N}\{x_n \in (a_0, a^*)\}\right\} \bigcap \left\{\bigcup_{n=1}^{N}\{x_n \in (a^*, a_1)\}\right\}\right] \\
&\underset{\substack{\uparrow \\ \text{De Morgan}}}{=} 1 - \Pr_{\boldsymbol{x} \underset{i.i.d.}{\sim} p_0(x)}\left[\left\{\bigcap_{n=1}^{N}\{x_n \notin (a_o, a^*)\}\right\} \bigcup \left\{\bigcap_{n=1}^{N}\{x_n \notin (a^*, a_1)\}\right\}\right] \\
&\underset{\text{union bound}}{\ge} 1 - \Pr_{\boldsymbol{x} \underset{i.i.d.}{\sim} p_0(x)}\left[\bigcap_{n=1}^{N}\{x_n \notin (a_0, a^*)\}\right] - 1 - \Pr_{\boldsymbol{x} \underset{i.i.d.}{\sim} p_0(x)}\left[\bigcap_{n=1}^{N}\{x_n \notin (a^*, a_1)\}\right] \\
&= 1 - (1-\epsilon)^N - (1-\epsilon)^N \\
&\underset{\substack{\uparrow \\ \text{see Fig. 18}}}{\ge} 1 - 2\exp(-N\epsilon) \underset{\substack{\uparrow \\ N \ge \frac{\ln(2/\delta)}{\epsilon}}}{\ge} 1 - 2\exp\left(-\frac{\ln(2/\delta)}{\epsilon}\epsilon\right) = 1 - \delta
\end{aligned}$$

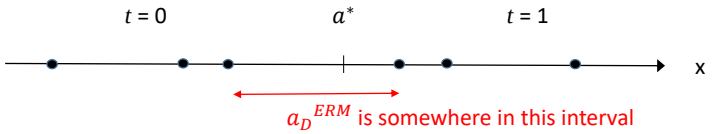


Figure 16: Illustration of ERM for the example in Appendix A.

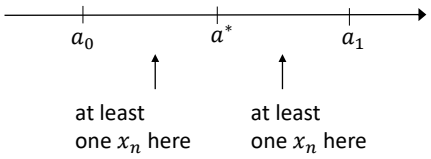


Figure 17: Illustration of the first inequality in the proof in Appendix A.

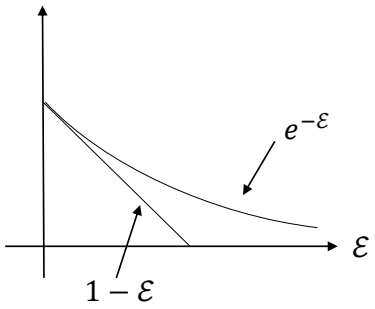


Figure 18: Graph of $1-\epsilon$ and e^{-t}.

5. Unsupervised Learning

1 Introduction

- Unsupervised learning: Given a data set consisting of input data without labeled responses,

$$\mathcal{D} = \{\boldsymbol{x}_n\} \underset{\text{i.i.d.}}{\sim} p_0(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^D, \tag{1}$$

  where $p_0(\boldsymbol{x})$ is the true unknown distribution, the goal is to learn the properties of $p_0(\boldsymbol{x})$

- Applications:
  - Density estimation
  - Classification
  - Dimensionality reduction
  - Feature extraction (for supervised learning, search, etc.)
  - Generation of new samples from $p_0(\boldsymbol{x})$ (e.g, computer graphics)
  - Outlier detection and monitoring
  - Compression

2 Type of Models

1. *Generative directed:* Generative directed models are illustrated in Fig. 1 and are characterized by models of the form

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}|\boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}) \tag{2}$$

   where the variables $\boldsymbol{z}_n$ known as latent or hidden, identify classes or features that define the data point $\boldsymbol{x}$. Directed models capture the cause-effect nature of the relationship between $\boldsymbol{z}$ and $\boldsymbol{x}$.

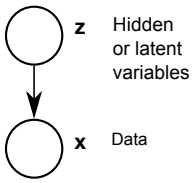


Figure 1: Generative directed models.

2. *Generative undirected models*: Generative undirected models parametrize directly the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{z}$ as

$$p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}) \tag{3}$$

Unlike directed models, as sketched in Fig. 2, undirected models capture the affinity, or compatibility, of given configurations of values for $\boldsymbol{z}$ and $\boldsymbol{x}$.

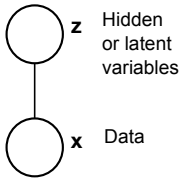


Figure 2: Generative undirected models.

3. *Autoencoders*: Autoencoders are (typically deterministic) machines that convert unsupervised learning problems into supervised learning problems by using the data $\boldsymbol{x}_n$, define both the domain point and the label.

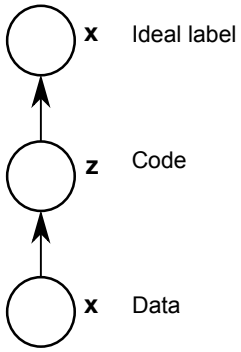


Figure 3: Autoencoders.

4. *Multiple hidden layers*: As shown in Fig. 4, there exist multiple layer extensions of all models discussed above.

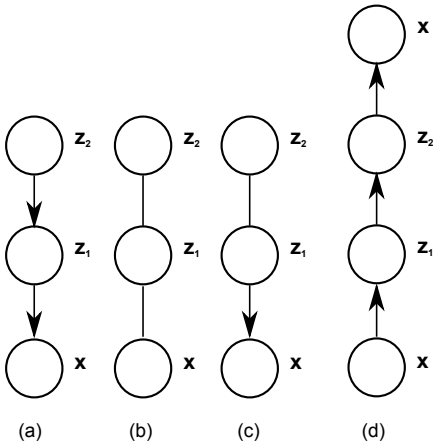


Figure 4: Multiple hidden layer models (a) Helmholtz machine; (b) Deep Boltzmann Machine; (c) Deep Belief Network; (d) Deep autoencoder.

- There are also discriminative models, such Regularized Information Maximization (RIM), in which one optimizes $p(z|x)$.
- Unlike supervised learning, there is no general theoretical framework for unsupervised learning given that it encompasses a variety of different tasks.

3 Warm-Up: K-Means Clustering

- Dataset $\mathcal{D} = \{\boldsymbol{x}_n\}$, $\boldsymbol{x}_n \in \mathbb{R}^{\mathrm{D}}$
- Goal: Assign each vector $\boldsymbol{x}_n$ to a class, or a cluster, C_k encoded by a categorical variable $\boldsymbol{z}_n$ via one hot encoding:
  - $[\boldsymbol{z}_n]_k = \boldsymbol{z}_{nk} = \begin{cases} 1 & \text{if } x_n \text{ belongs to cluster } C_k \\ 0 & \text{otherwise} \end{cases}$

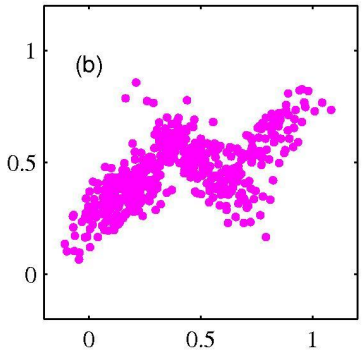


Figure 5: Example of a dataset $\mathcal{D}$ for $D = 2$ (from [1]).

- An example of a dataset $\mathcal{D}$ is shown in Fig. 5. Note that the labels $\{\boldsymbol{z}_n\}$ are not observed.
- K-means is a heuristic method that assigns each cluster a "prototype" vector $\boldsymbol{\mu}_k$ and aims at solving

$$\underset{\{\boldsymbol{z}_n\},\{\boldsymbol{\mu}_k\}}{\text{maximize}} \sum_{n=1}^{N}\sum_{k=1}^{K} z_{n,k} d(\boldsymbol{x}_n, \boldsymbol{\mu}_k) = f\left(\{\boldsymbol{z}_n\}, \{\boldsymbol{\mu}_k\}\right) \tag{4}$$

where

$$d(\boldsymbol{x}_n, \boldsymbol{\mu}_k) = ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2 \tag{5}$$

is the squared Euclidean distance used by the K-means algorithm. A more general distance metric yields the K-medoids algorithm. It is in fact possible to apply clustering to discrete data as long as the distance d is properly defined (most generally, by a matrix of pairwise dissimilarities).

- The K-means algorithm performs alternatively optimization of $\{\boldsymbol{z}_n\}$ and $\{\boldsymbol{\mu}_k\}$:

1. Initialize cluster positions $\left\{\boldsymbol{\mu}_k^{old}\right\}$.

2. E step: (minimization over $\{\boldsymbol{z}_n\}$):

$$z_{n,k}^{new} = \begin{cases} 1 & k = \underset{j}{\text{argmin}}\ d(\boldsymbol{x}_n, \boldsymbol{\mu}_j^{old}) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

   - Each training point is assigned to the cluster with the closest prototype.
   - Requires the computation of K distances for each $\boldsymbol{x}_n$.

3. M step: (minimization over $\{\boldsymbol{\mu}_k\}$):

$$\nabla_{\boldsymbol{\mu}_k} f(\{\boldsymbol{z}_k^{\text{new}}\}\{\boldsymbol{\mu}_k\}) = 0 \tag{7}$$

$$\Rightarrow \boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{n=1}^{N} z_{n,k}^{\text{new}} x_n}{\sum_{n=1}^{N} z_{n,k}^{\text{new}}} \tag{8}$$

   where $\boldsymbol{\mu}_{\text{k}}^{\text{new}}$is the mean of the $\sum_{\text{n=1}}^{\text{N}} z_{n,k}^{\text{new}}$ data points assigned to cluster k.

4. If convergence criterion is not satisfied, set $\{\boldsymbol{\mu}_k^{old}\} \leftarrow \{\boldsymbol{\mu}_k^{new}\}$ and return to 2.

- An example of the evolution of the K-means algorithm is shown in Fig. 6 and the corresponding value of the objective function in equation (4) is shown in Fig. 7.
- K-means alternates between making inferences $\{\boldsymbol{z}_n\}$ about the hidden variables based on the model $\{\boldsymbol{\mu}_k\}$ and updating the model to match data $\{\boldsymbol{x}_n\}$ and the inferred variables $\{\boldsymbol{z}_n\}$. As a consequence, the objective function in (4) is non-increasing across the iterations.
- How to choose K?
  - Add/remove clusters until certain (heuristic) criteria are satisfied (e.g., purity of clusters).
  - Hierarchical clustering: Build a tree with clusters of increasing sizes moving to the root, see Fig. 7 for a dendrogram.
  - Let K be selected automatically by adopting a non-parametric Bayesian approach: Use a Dirichlet process prior for $p(z)$ that allows for any number of clusters [2].

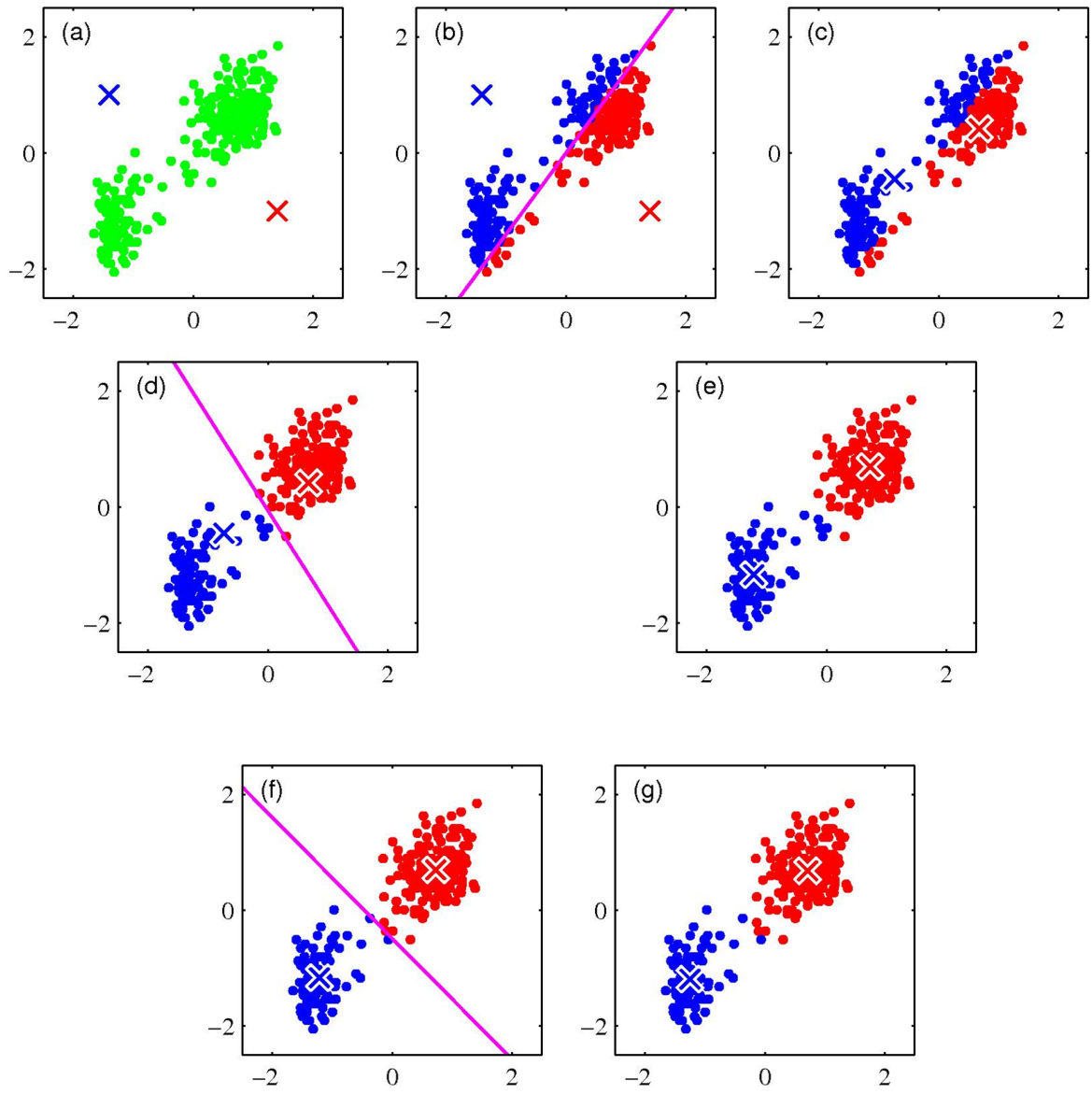


Figure 6: Illustration of three iterations of the K-means algorithm (from [1]).

4 Non-Convexity of Unsupervised Learning

- ML inference in unsupervised learning based on probabilistic models requires the solution of the problem,

$$\underset{\theta}{\text{maximize}} \ln p(x|\theta) = \ln \left(\sum_z p(x, z|\theta) \right) \tag{9}$$

  where x denotes the data and z the hidden or latent variables.

- In most problems of interest, $p(x, z|\theta)$ is from the exponential family and hence, $\ln(p(x, z|\theta))$ is concave in θ. Therefore, the supervised learning problem

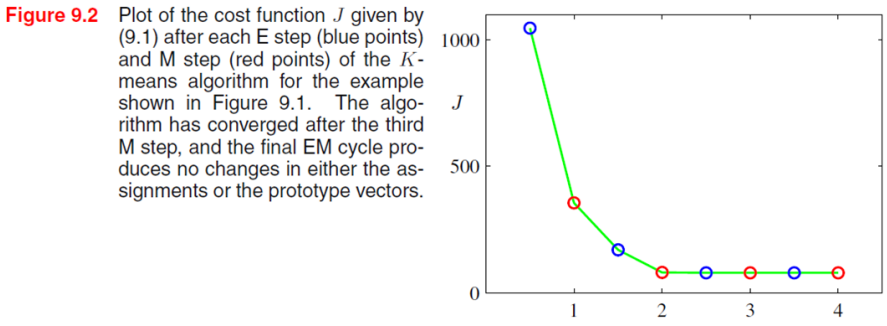


Figure 7: Evolution of (4) across the iterations of K-means (from [1])

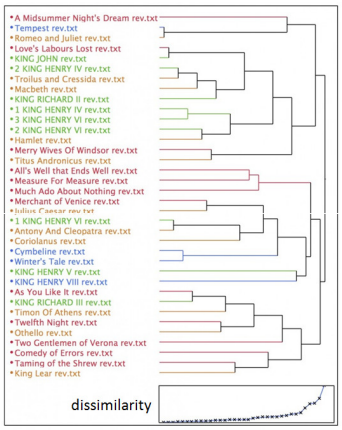


Figure 8: A dendrogram illustrates a number of possible clustering solutions with increasing K, and hence decreasing similarity within each cluster, as we move towards the root.

$$\underset{\theta}{\text{maximize}} \ln p(x|\theta) \tag{10}$$

concave and can be solved as seen in Chapter 2.

- The problem (9) is instead non-convex, see Fig. 9 for an illustration.
- Non-convex problems may be tackled via gradient-based methods or variations thereof. However, unlike convex problems, gradient-based methods can typically guarantee only convergence to stationary points, that is, to local minima, maxima or to saddle points. Various tricks exist to ensure that local minima are reached, see, e.g., [5]. However, these approaches are generic and do not leverage the special structure of problems such as (9). In the rest of this chapter, we study tools that can potentially outperform general-purpose non-convex optimizers by exploiting the special structure of the problem. The key idea will be that of performing inference of the latent variables (as in the E step of K-means) in order to tackle convex problems with fully observed variables (as in the M step of K-means).

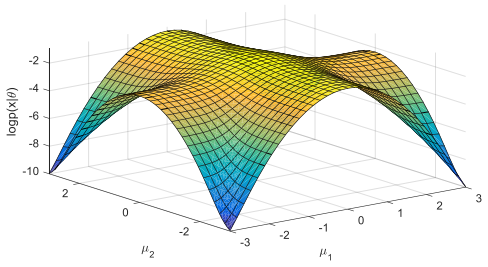


Figure 9: Illustration of the non-convex nature of ML in unsupervised learning.

5 Tools: ELBO and EM Algorithm

- Many methods to tackle the problem are based on the maximization of the Evidence Lower Bound (ELBO) (or variational free energy or Helmholtz energy).
- Definition: For any distribution $q(z)$, the ELBO $\mathcal{L}(q,\theta)$is defined as

$$\mathcal{L}(q,\theta) = \underbrace{E_{z\sim q(z)}[\ln p(x,z|\theta)]}_{\text{energy term}} + \underbrace{H(q)}_{\text{entropy term}} \tag{11}$$

$$= E_{z\sim q(z)}[\ln p(x|z,\theta)] - \underbrace{\mathrm{KL}\left(q(z)||p(z|\theta)\right)}_{\text{variational regularization}} \tag{12}$$

$$= -\mathrm{KL}\left(q(z)||p(x,z|\theta)\right) \tag{13}$$

- Note that in (12) we have used the convention of defining $\mathrm{KL}(p||q)$ even when q is not normalized (see Chapter 1). A key property is that $\mathcal{L}(q,\theta)$ is convex in θ and in $q(z)$ (see Fig. 10 and Fig. 11).

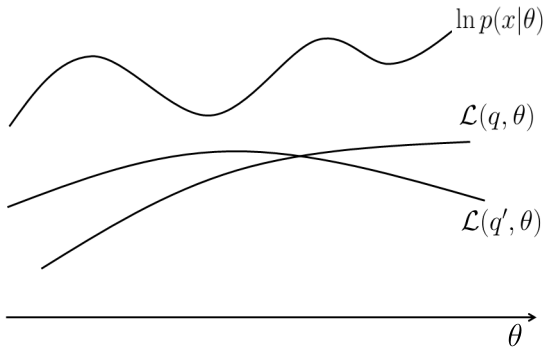


Figure 10: Illustration of the ELBO for two distributions $q(z)$ and $q'(z)$

- Theorem (ELBO): We have the equality

$$\ln p(x|\theta) = \mathcal{L}(q,\theta) - \mathrm{KL}(q(z)||p(z|x,\theta)) \tag{14}$$

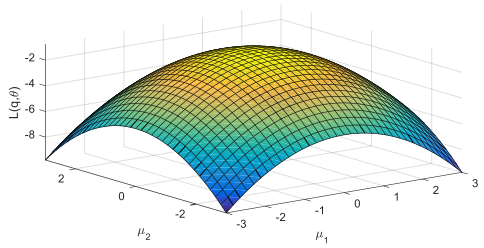


Figure 11: Example of an ELBO for the log-likelihood in Fig. 9.

and hence the inequality

$$\ln p(x|\theta) \;\geq \mathcal{L}(q,\theta) \tag{15}$$

holds for all θ, with equality at a value $\theta^{'}$ if and only if $q(z) = p(z|x,\theta^{'})$.

- The ELBO can be in principle easily optimized over θ, since it is a concave function, and also over q and the optimal q is $p(z|x,\theta)$ (see Fig. 12).

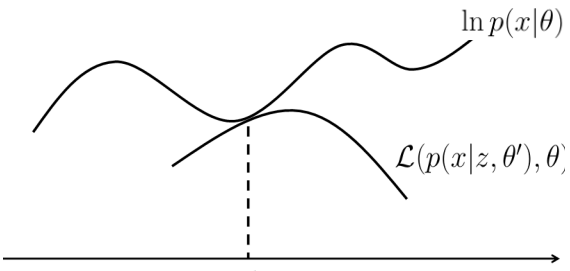


Figure 12: Illustration of a tight ELBO at a parameter value $\theta = \theta^{'}$

- Proof :

$$\ln p(x|\theta) = \ln \frac{p(x,z|\theta)}{p(z|x,\theta)} \quad \text{for every } z \tag{16}$$

$$= \ln \left(\frac{p(x,z|\theta)}{q(z)} \frac{q(z)}{p(z|x,\theta)} \right) \quad \text{for every } z \tag{17}$$

$$= \underbrace{\sum_z q(z) \ln \frac{p(x,z|\theta)}{q(z)}}_{\mathcal{L}(q,\theta)} + \underbrace{\sum_z q(z) \ln \frac{q(z)}{p(z|x,\theta)}}_{\mathrm{KL}(q(z)||p(z|x,\theta))} \tag{18}$$

- Alternative proof:

$$\ln p(x|\theta) = \ln\left(\sum_z p(x,z|\theta)\right) \tag{19}$$

$$= \ln\left(\sum_z q(z)\frac{p(x,z|\theta)}{q(z)}\right) \tag{20}$$

$$\underset{\text{Jensen's inequality}}{\geq} \sum_z q(z)\ln\left(\frac{p(x,z|\theta)}{q(z)}\right) = \mathcal{L}(q,\theta) \tag{21}$$

- The ELBO can be generalized as the multi-sample ELBO-weighted bound

$$\ln p(x|\theta) \geq E_{\boldsymbol{z} \underset{i.i.d}{\sim} q(\boldsymbol{z})}\left[\ln\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p(x,z_k|\theta)}{q(z_k)}\right)\right] \tag{22}$$

Note that, as $K \to \infty$, the inequality becomes tight by the law of large numbers.

- As mentioned, a large number of schemes is based on the maximization of ELBO $\mathcal{L}(q,\theta)$.
- A classical algorithm is the EM which performs the following alternate optimization (see Fig. 13):

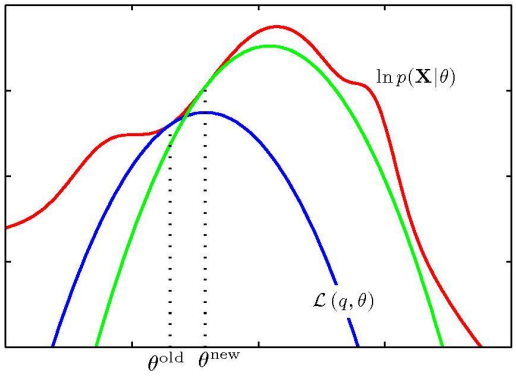


Figure 13: Illustration of the EM algorithm (from [1]).

E step:

$$\underset{q}{\text{maximize}}\ \mathcal{L}(q,\theta^{old}) \Rightarrow p(z|x,\theta^{old}) = q^{new}(z) \tag{23}$$

M step:

$$\underset{\theta}{\text{maximize}}\ \mathcal{L}(q^{new},\theta) \to \theta^{new} \tag{24}$$

- More formally:

  EM Algorithm

  1. Initialize θ^{old}
  2. E step: Evaluate $p(z|x,\theta^{old})$;

3. M step: Solve

$$\underset{\theta}{\text{maximize}}\ Q(\theta,\theta^{old}) = \sum_z p(z|x,\theta^{old}) \ln p(x,z|\theta) \tag{25}$$

$$= E_{z\sim p(z|x,\theta^{old})} \left[\ln p(x,z|\theta)\right] \tag{26}$$

producing θ^{new}.

4. If convergence is not satisfied, set $\theta^{old} \leftarrow \theta^{new}$ and return to step 2.

- If $\{\boldsymbol{x}_n, \boldsymbol{z}_n\}$ are i.i.d. (given $\boldsymbol{\theta}$), then the E and M steps can be carried out separately for each data point $\boldsymbol{x}$:

$$p(\boldsymbol{z},\boldsymbol{x}|\boldsymbol{\theta}) = \frac{p(\boldsymbol{x},\boldsymbol{z}|\theta)}{p(\boldsymbol{x}|\theta)} = \prod_{n=1}^{n} \frac{p(\boldsymbol{x}_n,\boldsymbol{z_n}|\boldsymbol{\theta})}{p(\boldsymbol{x}_n|\boldsymbol{\theta})} \tag{27}$$

$$= \prod_{n=1}^{n} p(\boldsymbol{z}_n|\boldsymbol{x}_n,\boldsymbol{\theta}) \text{ (E step)} \tag{28}$$

and

$$E_{\boldsymbol{z}}\left[\ln p(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\theta})\right] = \sum_{n=1}^{N} E_{z_n\sim p(\boldsymbol{z}_n|\boldsymbol{x}_n,\boldsymbol{\theta})}\left[\ln p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta})\right] \text{ (M step).} \tag{29}$$

- An illustration is shown in Fig. 13.
- The EM algorithm is guaranteed to increase $\ln p(x|\theta)$ at each iteration unless a local maximum is attained.
- Both steps can be approximated when too expensive:
  - E step: Instead of the exact posterior $p(z|x,\theta^{old})$, a parametrized approximation can be learned instead (approximate inference, see Chapter 7);
  - M step:
    - The expectation can be approximated via an empirical (Monte Carlo) average (see Chapter 7).
    - Instead of the exact maximization, one can apply one of more steps from the gradient $\nabla_\theta \mathcal{L}\left(p(z|x,\theta^{old}),\theta\right) = E_{z\sim p(z|x,\theta^{old})}\left[\nabla_\theta \ln p(x,z|\theta)\right]$.
    - The gradient can also be approximated via an empirical (Monte Carlo) average (see Chapter 7).
- For extensions of EM algorithm, see [4, p. 247-248].
- It can be proven that

$$\nabla_\theta \mathcal{L}\left(p(z|x,\theta^{old}),\theta\right)|_{\theta=\theta^{old}} = \nabla_\theta \ln p(x|\theta)|_{\theta=\theta^{old}}. \tag{30}$$

6 Generative Directed Models

6.1 Mixture of Gaussians Model

- Mixture of Gaussians models (Fig. 14):

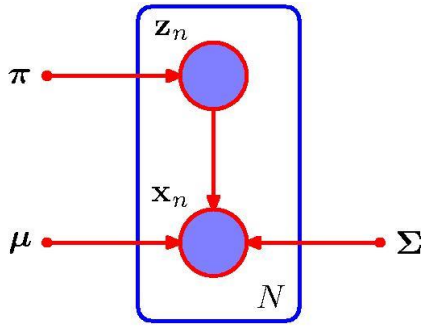


Figure 14: Bayesian network for mixture of Gaussians model (from [1]).

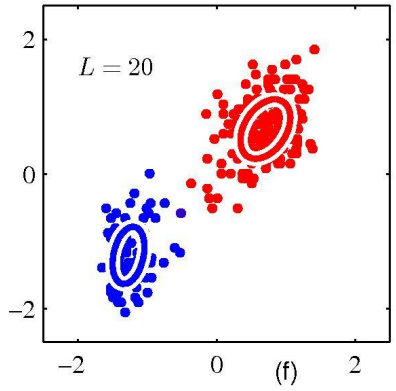


Figure 15: Example of a trained mixture of Gaussians model (from [1]).

- Example of the result of a mixture of Gaussians model whose parameters are selected by means of the EM algorithm can be found in Fig. 15.
- This model can be thought as the unsupervised version of the QDA model for the fully observed (or supervised) case studied in Chapter 3.
- Applications: classification, density estimation, etc.
- We have

$$\ln p(\boldsymbol{x}_n, \boldsymbol{z}_n = k|\boldsymbol{\theta}) = \ln \pi_k + \ln \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{31}$$

and

$$\ln p(\boldsymbol{x}_n|\boldsymbol{\theta}) = \ln \left(\sum_k \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right). \tag{32}$$

- The ML problem can hence, be formulated as

$$\underset{\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}}{\text{maximize}} \ln p(\boldsymbol{x}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right) \right\} \tag{33}$$

which is non-convex.

- Instead, the problem with complete data coincides with QDA:

$$\underset{\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}}{\text{maximize}} \ln p\left(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}\right) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \left\{ \ln \pi_k + \ln \mathcal{N}\left(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right) \right\} \tag{34}$$

which yields the ML solutions:

$$\pi_k = \frac{N_k}{N} \text{ with } N_k = \sum_{n=1}^{N} z_{nk} \tag{35}$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} \boldsymbol{x}_n \text{ mean of cluster } k \tag{36}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_n)(\boldsymbol{x}_n - \boldsymbol{\mu}_n)^T \tag{37}$$

- EM Algorithm:
  - E step:

$$p(z_{nk} = 1|\boldsymbol{x}_n,\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}) = E\left[z_{nk}|\boldsymbol{x}_n,\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}\right] = \frac{\pi_k \mathcal{N}\left(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{x}_n|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j\right)} = E\left[z_{nk}\right], \tag{38}$$

    also known as responsibility of class k for data in n.
  - M step:

$$E_{\boldsymbol{z}_n}\left[\ln p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta})\right] = \sum_{k=1}^{K} E\left[z_{nk}\right] \left\{ \ln \pi_k + \ln \mathcal{N}\left(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right) \right\}. \tag{39}$$

- Therefore, the M step is equivalent to QDA, if we substitute $z_{nk} \leftarrow E[z_{nk}]$:

$$\pi_k = \frac{N_k}{N} \text{ with } N_k = \sum_{n=1}^{N} E\left[z_{nk}\right] \tag{40}$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} E\left[z_{nk}\right] \boldsymbol{x}_n \tag{41}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} E\left[z_{nk}\right] (\boldsymbol{x}_n - \boldsymbol{\mu}_n)(\boldsymbol{x}_n - \boldsymbol{\mu}_n)^T \tag{42}$$

- See Fig. 16 for an example.

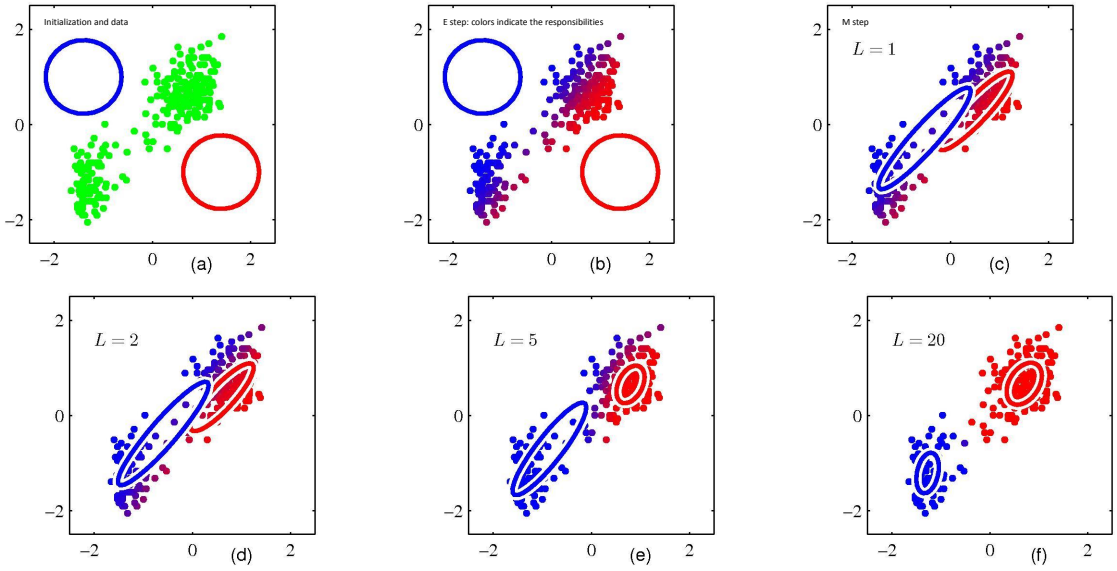


Figure 16: Illustration of EM for the Gaussian mixture model (from [1]).

- Relationship between EM and K-means: Setting $\boldsymbol{\Sigma}_k = \epsilon \boldsymbol{I}$ as known and letting $\epsilon \to 0$ makes K-means a special case of EM (see [1, p. 443]).

6.2 Mixture of Bernoulli Model

- Consider a set of N binary images $\mathcal{D} = \{\boldsymbol{x}_n\}$ representing handwritten digits as seen in Fig. 17.

Figure 17: Example of a dataset $\mathcal{D}$ for handwritten digit classification.

- We wish to cluster images (unsupervised learning) into K classes.
- We use a mixture of Bernoulli model (see Fig. 18):

$$\boldsymbol{z}_n \sim \text{Cat}(\boldsymbol{\pi}) \tag{43}$$

$$\boldsymbol{x}_n | \boldsymbol{z}_n = k \sim \prod_{i=1}^{D} \text{Bern}\left(x_{ni} | \mu_{ki}\right) \tag{44}$$

$$\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k) \tag{45}$$

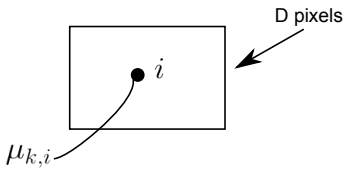


Figure 18: The mixture of Bernoulli model defines a probability μ_{ki} for each class k and pixel i.

- E step :

$$p(z_{nk}=1|\boldsymbol{x}_n,\boldsymbol{\pi},\boldsymbol{\mu})=\frac{\pi_k\prod_{i=1}^{D}\text{Bern}\left(\boldsymbol{x}_{ni}|\mu_{ki}\right)}{\sum_{j=1}^{K}\pi_j\prod_{i=1}^{D}\text{Bern}\left(\boldsymbol{x}_{ni}|\mu_{ji}\right)}=E\left[z_{nk}\right] \tag{46}$$

- M step:

$$E_{\boldsymbol{z}_n}\left[\ln p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta})\right]=\sum_{k=1}^{K}E\left[z_{nk}\right]\left(\ln\pi_k+\sum_{i=1}^{D}\left(x_{ni}\ln\mu_{ki}+(1-x_{ni})\ln\left(1-\mu_{ki}\right)\right)\right) \tag{47}$$

and

$$\boldsymbol{\mu}_k=\frac{1}{N_k}\sum_{n=1}^{N}E\left[\boldsymbol{x}_{nk}\right]\boldsymbol{x}_n \text{ vector of effective number of samples with } \{x_{ni}=1\} \text{ inclass } K \tag{48}$$

$$N_k=\sum_{n=1}^{N}E\left[\boldsymbol{x}_{nk}\right] \text{ effective number of observations in class } K \tag{49}$$

$$\pi_k=\frac{N_k}{N} \tag{50}$$

- An illustration is found in Fig. 19.

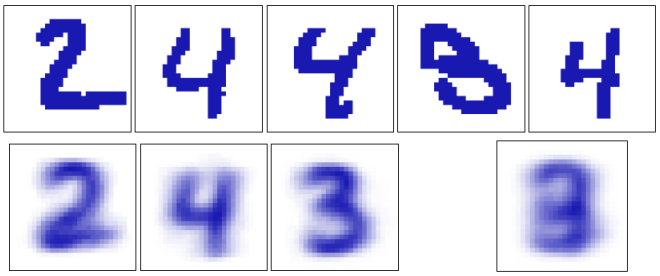


Figure 9.10 Illustration of the Bernoulli mixture model in which the top row shows examples from the digits data set after converting the pixel values from grey scale to binary using a threshold of 0.5. On the bottom row the first three images show the parameters μ_{ki} for each of the three components in the mixture model. As a comparison, we also fit the same data set using a single multivariate Bernoulli distribution, again using maximum likelihood. This amounts to simply averaging the counts in each pixel and is shown by the right-most image on the bottom row.

Figure 19: Illustration of a learned mixture of Bernoulli model (from [1]).

6.3 Probabilistic Principal Component Analysis

- In the models considered so far, hidden variables indicate the cluster or class for any data point. More generally, latent variables can identify the features that define each data point.
- Probabilistic PCA uses a linear factor generative model with $M < D$ features (see Fig. 20):

$$\begin{aligned} \boldsymbol{z}_n &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) && M \times 1 \\ \boldsymbol{x}_n &= \boldsymbol{W}\boldsymbol{z}_n + \boldsymbol{\mu} + \boldsymbol{\epsilon}, && \epsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \boldsymbol{I}\right) \Rightarrow \boldsymbol{x}_n | \boldsymbol{z}_n \sim \mathcal{N}\left(\boldsymbol{W}\boldsymbol{z}_n + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right) \\ \boldsymbol{\theta} &= \begin{pmatrix} \boldsymbol{W} \\ \boldsymbol{\mu} \\ \sigma^2 \end{pmatrix} && \begin{matrix} D \times M \\ D \times 1 \end{matrix} \end{aligned}$$

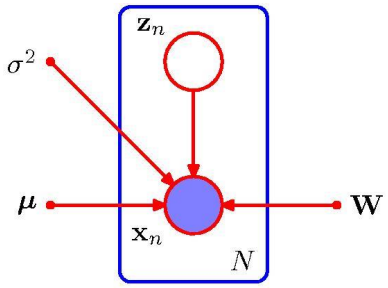


Figure 20: Bayesian network for Probabilistic PCA (from [1]).

- Applications: Dimensionality reduction, feature extraction, density estimation.
- The columns of $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2 ... \boldsymbol{w}_M]$ can be interpreted as features since (see Fig. 21):

$$\boldsymbol{x}_n = \sum_{m=1}^{M} \boldsymbol{w}_m z_{mn} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{51}$$

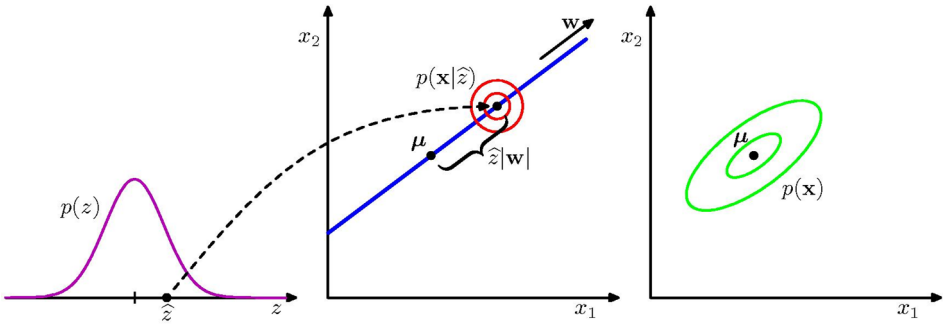


Figure 21: Illustration of Probabilistic PCA (from [1]).

- An EM Algorithm can be designed to estimate $\boldsymbol{W}$, $\boldsymbol{\mu}$,$\boldsymbol{\sigma}^2$ ([1, p. 577]), see Fig. 22.

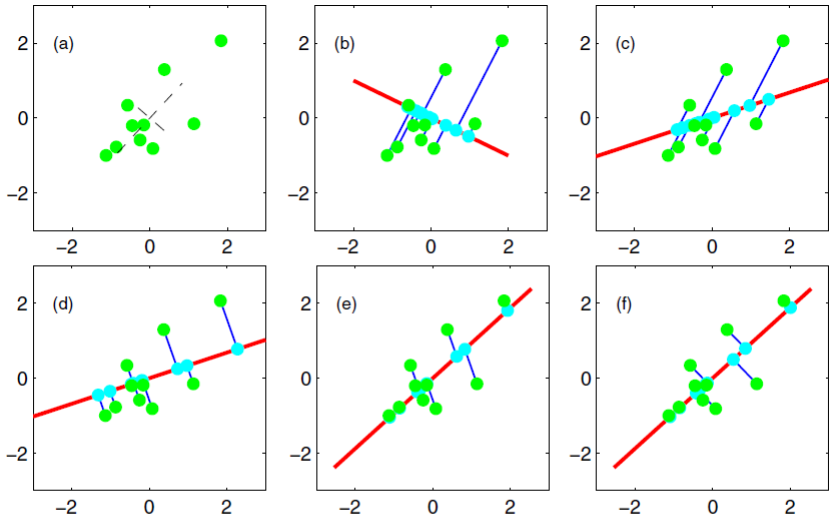


Figure 12.12 Synthetic data illustrating the EM algorithm for PCA defined by (12.58) and (12.59). (a) A data set $\mathbf{X}$ with the data points shown in green, together with the true principal components (shown as eigenvectors scaled by the square roots of the eigenvalues). (b) Initial configuration of the principal subspace defined by $\mathbf{W}$, shown in red, together with the projections of the latent points $\mathbf{Z}$ into the data space, given by $\mathbf{ZW}^{\mathrm{T}}$, shown in cyan. (c) After one M step, the latent space has been updated with $\mathbf{Z}$ held fixed. (d) After the successive E step, the values of $\mathbf{Z}$ have been updated, giving orthogonal projections, with $\mathbf{W}$ held fixed. (e) After the second M step. (f) After the second E step.

Figure 22: Illustration of the EM algorithm for Probabilistic PCA (from [1]).

- Via Bayes theorem

$$p\left(\boldsymbol{z}_n|\boldsymbol{x}_n,\boldsymbol{\theta}\right)=\mathcal{N}\left(\boldsymbol{z}_n|\boldsymbol{M}^{-1}\boldsymbol{W}^T(\boldsymbol{x}-\boldsymbol{\mu}),\sigma^2\boldsymbol{M}\right) \tag{52}$$

  where $\boldsymbol{M}=\boldsymbol{W}^T\boldsymbol{W}+\sigma^2\boldsymbol{I}$.

- When $\boldsymbol{z}\sim\prod_j p(z_j)$ but not Gaussian, we obtain Independent Component Analysis (ICA), (see [1, p. 591]). It is also possible to impose structure on each $p(z_j)$, e.g., sparsity via t-student or Laplace priors. A general discussion on linear factor models can be found [3].

- Techniques that attempt to extract features or more generally to infer the structure in $p_0(\boldsymbol{x})$ by fitting a generative directed model, are also known as "analysis by synthesis".

7 Generative Undirected Models

7.1 Restricted Boltzmann Machines (RBM)

- Bernoulli - Bernoulli RBM:

$$p(\boldsymbol{x},\boldsymbol{z}|\underbrace{\boldsymbol{W},\boldsymbol{a},\boldsymbol{b}}_{\boldsymbol{\theta}})=\frac{1}{Z(\boldsymbol{\theta})}\exp\left(-E(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\theta})\right), \tag{53}$$

  with $\boldsymbol{x}\in\{0,1\}^D$, $\boldsymbol{z}\in\{0,1\}^M$ and with the energy function

$$E(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) = -\boldsymbol{a}^T\boldsymbol{z} - \boldsymbol{b}^T\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{z} \tag{54}$$

and the partition function $\boldsymbol{Z}(\theta) = \sum_{\boldsymbol{x},\boldsymbol{z}} \exp\left(-E\left(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}\right)\right)$.

- It belongs to the exponential family.
- Applications: feature extraction for binary images ($M < D$).
- The columns of $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, .., \boldsymbol{w}_M]$ can be interpreted as features, as illustrated in Fig. 23.

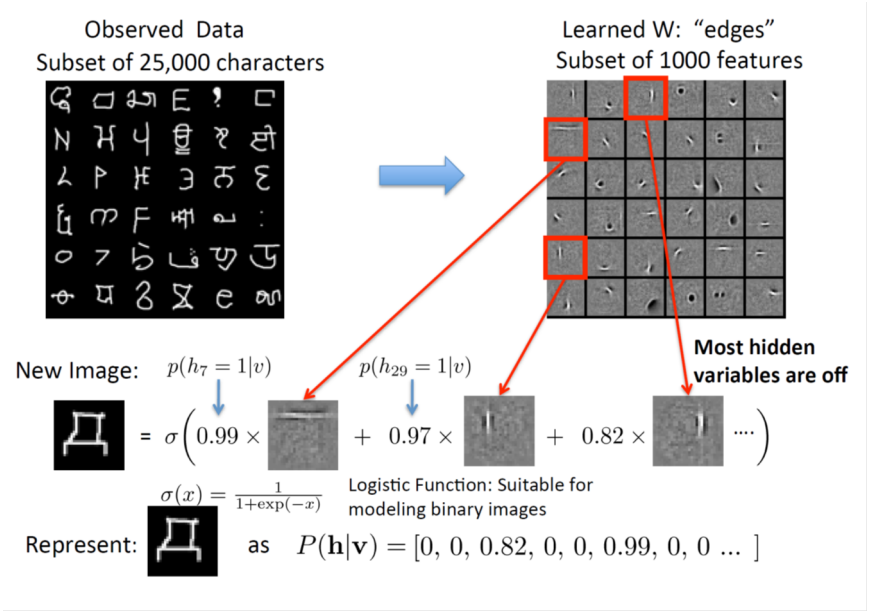


Figure 23: In RBMs, the columns of $\boldsymbol{W}$ can be interpreted as features (from [7]).

- From the model, it is easy to compute

$$p(z_j = 1|\boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{w}_j^T\boldsymbol{x} + a_j\right) \tag{55}$$

$$p(x_i = 1|\boldsymbol{z}, \boldsymbol{\theta}) = \sigma\left(\widetilde{\boldsymbol{w}}_i\boldsymbol{z} + b_i\right) \tag{56}$$

  where $\widetilde{\boldsymbol{w}}_i$ is the i^{th} row of $\boldsymbol{W}$.
- Note that z_j tends to equal 1 if $\boldsymbol{x}$ is correlated with the feature $\boldsymbol{w}_j$.
- Learning is typically done by means of an approximate EM algorithm based on gradient ascent:
  M step: Apply gradient ascent instead of exact optimization (see Fig. 24):

$$\nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{old}} = E_{\boldsymbol{z}\sim p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\theta}^{old})}\left[\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{old}}\right] \tag{57}$$

- The gradients can be computed as

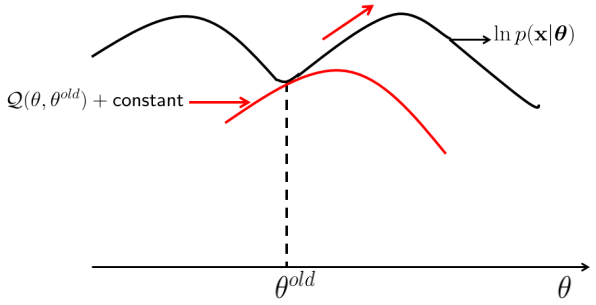


Figure 24: Illustration of gradient based learning of RBMs.

$$\begin{aligned}\nabla_{w_{ij}} Q(\theta, \theta^{old})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{old}} &= E_{z_j \sim p(\boldsymbol{z}_j|\boldsymbol{x}, \boldsymbol{\theta}^{old})}\left[x_i z_j\right] - E_{\boldsymbol{x},\boldsymbol{z} \sim p(\boldsymbol{x},\boldsymbol{z}|\theta^{old})}\left[x_i' z_j'\right] \\ &= x_i \sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x} + a_j^{old}\right) - E_{\boldsymbol{x}' \sim p(\boldsymbol{x}|\boldsymbol{\theta}^{old})}\left[x_i' \sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x}' + a_j^{old}\right)\right] \end{aligned} \tag{58}$$

and similarly for $\boldsymbol{a}$ and $\boldsymbol{b}$:

$$\begin{aligned}\nabla_{a_j} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) &= E_{z_j \sim p(z_j|\boldsymbol{x}, \boldsymbol{\theta}^{old})}\left[z_j\right] - E_{z_j' \sim p(z_j, \boldsymbol{\theta}^{old})}\left[z_j'\right] \\ &= \sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x} + a_j^{old}\right) - E_{\boldsymbol{x}' \sim p(\boldsymbol{x}|\boldsymbol{\theta}^{old})}\left[\sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x}' + a_j^{old}\right)\right] \end{aligned} \tag{59}$$

$$\nabla_{b_i} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = x_i - E_{\boldsymbol{z}' \sim p(\boldsymbol{z}|\boldsymbol{\theta}^{old})}\left[\sigma\left(\widetilde{\boldsymbol{w}}_i^{old} \boldsymbol{z}' + b_i^{old}\right)\right] \tag{60}$$

- E step: Generate samples $\boldsymbol{x}'$,$\boldsymbol{z}' \sim p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}^{old})$ so as to compute the averages in (58)-(60). This is typically done using Contrastive Divergence, an example of Markov Chain Monte Carlo techniques:
  - Start with $\boldsymbol{x}$ (data)
  - Generate the independent variables:

$$\begin{aligned} \widetilde{z}_j &\underset{\text{i.i.d.}}{\sim} \text{Bern}\left(\sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x} + a_j^{old}\right)\right) \\ x_i' &\underset{\text{i.i.d.}}{\sim} \text{Bern}\left(\sigma\left(\widetilde{\boldsymbol{w}}_i^{old} \widetilde{\boldsymbol{z}} + b_i^{old}\right)\right) \\ z_j' &\underset{\text{i.i.d.}}{\sim} \text{Bern}\left(\sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x}' + a_j^{old}\right)\right) \end{aligned}$$

  - Use the resulting $\boldsymbol{x}'$ and $\boldsymbol{z}'$ to approximate the expectations:

$$E_{\boldsymbol{x}' \sim p(\boldsymbol{x}|\boldsymbol{\theta}^{old})}\left[x_i' \sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x}' + a_j^{old}\right)\right] \approx x_i^{'}\left(\sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x}^{'} + a_j^{old}\right)\right) \tag{61}$$

$$E_{\boldsymbol{x}' \sim p(\boldsymbol{x}|\boldsymbol{\theta}^{old})}\left[\sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x}' + a_j^{old}\right)\right] \approx \left(\sigma\left((\boldsymbol{w}_j^{old})^T \boldsymbol{x}^{'} + a_j^{old}\right)\right) \tag{62}$$

$$E_{\boldsymbol{z}' \sim p(\boldsymbol{z}|\boldsymbol{\theta}^{old})}\left[\sigma\left(\widetilde{\boldsymbol{w}}_i^{old} \boldsymbol{z}' + b_i^{old}\right)\right] \approx \left(\sigma\left(\widetilde{\boldsymbol{w}}_i^{old} \boldsymbol{z}' + b_i^{old}\right)\right) \tag{63}$$

- As illustrated in Fig. 24, it can be proven that

$$\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta})|_{\theta=\theta'} = \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}')|_{\theta=\theta'} \tag{64}$$

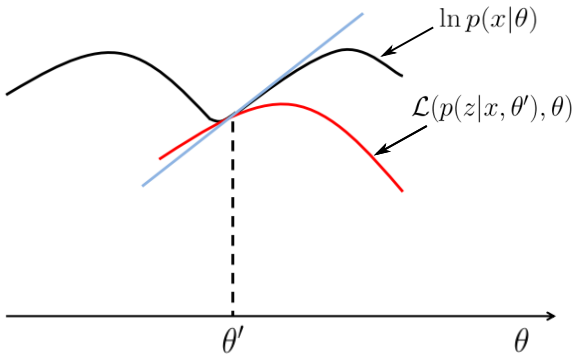


Figure 25: The ELBO at $\theta = \theta'$ has the same gradient at that point of the log-likelihood function and hence the approach can be directly justified as gradient ascent on the log-likelihood.

- There are other RBMs such as:
  - Gaussian-Bernoulli RBM, ex: non-binary images;
  - Categorical-Bernoulli RBM, ex: collaborative filtering for movie ratings (see Fig. 26).

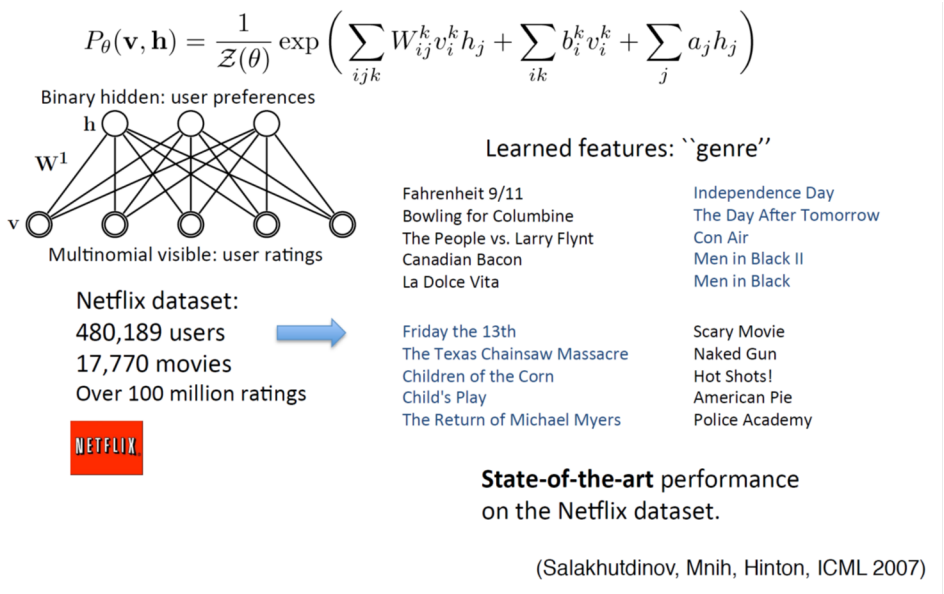


Figure 26: Categorical-Bernoulli RBM for collaborative filtering (from [7]).

8 Autoencoders

- Autoencoders include parametric models for both "encoder" (discriminative model) and "decoder" (generative model) as seen in Fig. 27.

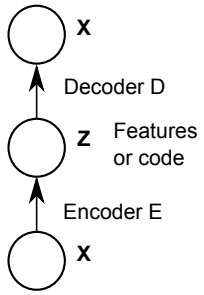


Figure 27: An autoencoder includes encoder E and decoder D.

- Autoencoders are based on deterministic models and convert the unsupervised problem to a supervised problem.
- A typical formulation of the training criterion is

$$\underset{E,D}{\textbf{minimize}} \sum_{n=1}^{N} \|\boldsymbol{x}_n - D\left(E(\boldsymbol{x}_n)\right)\|^2, \tag{65}$$

where constraints, such as dimensionality or sparsity, are imposed on $\boldsymbol{z}$ in order to avoid learning the identity mapping.

- Examples: a) PCA

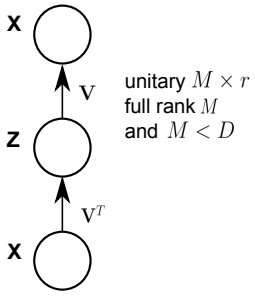


Figure 28: PCA.

  - Training:

$$\underset{\boldsymbol{V}}{\text{minimize}} \sum_{n=1}^{N} \|\boldsymbol{x}_n - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}_n\|^2 \tag{66}$$

$$\Rightarrow \widehat{\boldsymbol{V}} = M \text{ principal eigenvectors of the co-variance matrix } \tfrac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T$$

- See Fig. 27 for an example.
- PCA can be generalized to kernel PCA.
- It is a special case of probabilistic PCA with $\mu = 0$ and $\sigma^2 \to 0$.

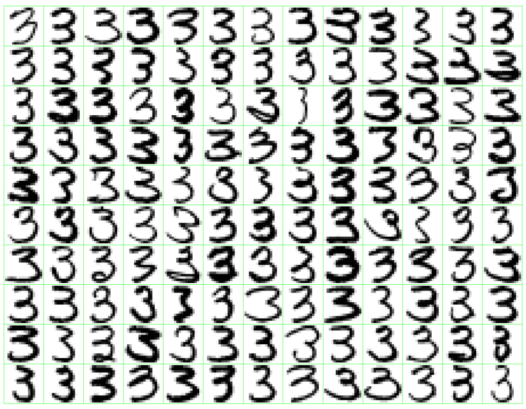


FIGURE 14.22. *A sample of 130 handwritten 3's shows a variety of writing styles.*

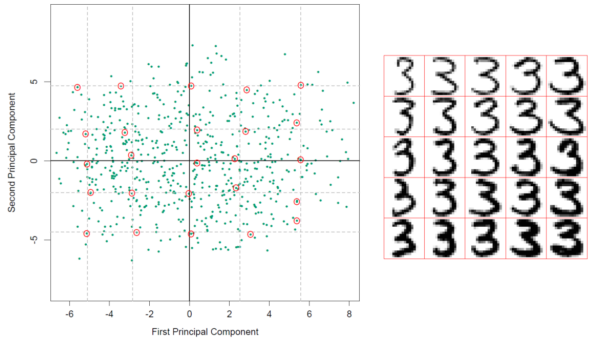


FIGURE 14.23. *(Left panel:) the first two principal components of the hand-written threes. The circled points are the closest projected images to the vertices of a grid, defined by the marginal quantiles of the principal components. (Right panel:) The images corresponding to the circled points. These show the nature of the first two principal components.*

Figure 29: An example of PCA (from [8]).

(b) Dictionary learning (see Fig. 29)

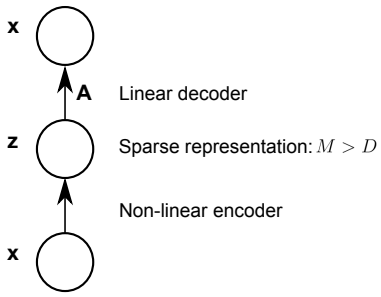


Figure 30: Dictionary Learning

$$\boldsymbol{z} \leftarrow \underset{\boldsymbol{z}}{\text{minimize}} \|\boldsymbol{x} - A\boldsymbol{z}\|^2 \text{subject to constraints such as sparsity } (\mathcal{C})$$

- Training:

$$\underset{\boldsymbol{A}}{\text{minimize}} \min_{\{\boldsymbol{z}_n\}\in\mathcal{C}} \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{x}_n - \boldsymbol{A}\boldsymbol{z}_n\|^2 \tag{67}$$

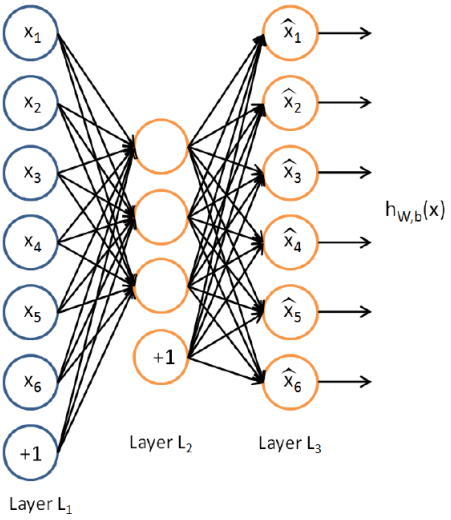


Figure 31: An autoencoder based on a multilayer neural network.

(c) Non-linear encoder-decoder as neural networks (see Fig. 31).

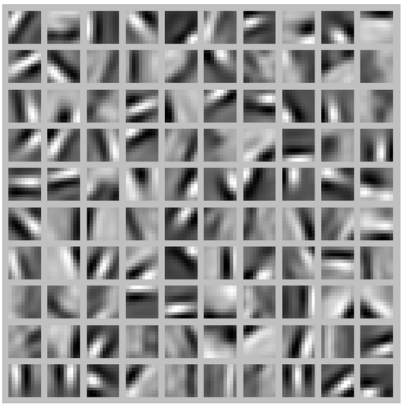
- Training:

$$L(\boldsymbol{w}) = \sum_{n=1}^{N} \|\boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w}) - \boldsymbol{x}_n\|^2, \tag{68}$$

where $\boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w})$ is the input - output relationship of the autoencoder, and $\boldsymbol{x}_n$ are the desired "labels" equal to the input.

Example:

  - input: x_n, 10×10 pixel images (D =100)
  - $M = 100$ hidden units ($M = D$ just for illustration)
  - output shown in Fig. 32: images that are matched to each hidden neuron (i.e., proportional to the corresponding vector $\boldsymbol{w}$)



Each square in the figure above shows the (norm bounded) input image x that maximally actives one of 100 hidden units. We see that the different hidden units have learned to detect edges at different positions and orientations in the image.

These features are, not surprisingly, useful for such tasks as object recognition and other vision tasks. When applied to other input domains (such as audio), this algorithm also learns useful representations/features for those domains too.

Figure 32: Example of a trained autoencoder based on a neural network.

9 Multiple Hidden Layers

- Different types of models with multiple layers exist as illustrated in Fig. 4:
  - Helmholtz machines
  - Deep Boltzmann Machines
  - Deep Belief networks
- Training is often done layer by layer.

10 A Different Type of Unsupervised Learning: PageRank

- Given a set of webpages, how should we rank them?
- Define a graph with a vertex for each webpage

$$L_{i,j} = \begin{cases} 1 & \text{if page } j \text{ links to page } i \\ 0 & \text{otherwise} \end{cases} \tag{69}$$

$$C_j = \sum_{n=1}^{N} L_{i,j} \tag{70}$$

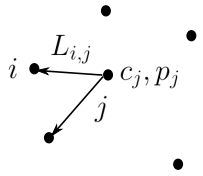


Figure 33: Key quantities in the PageRank algorithm.

- The rank p_i of a webpage i is given as

$$p_i = (1-d) + d \sum_{j \neq i} \frac{L_{ij}}{C_j} p_j \tag{71}$$

where $0 < d < 1$ is a parameter.

- In the formula above, the "vote" of a webpage j to the linked webpages is proportional to the rank of a page: pages with a lot of incoming links from important pages are highly ranked.
- The equation (70) can be solved recursively to obtain the pageranks of all pages.

11 Problems

1. For the dataset $\mathbf{x}$ used in the previous assignment (this file), we wish to implement the EM algorithm to train a mixture of Gaussians model with $K = 2$. To this end, pick K points randomly in the data set for the class means μ_k, initialize the covariance matrices as $\mathbf{\Sigma}_k = 10\mathbf{I}$ ($\mathbf{I}$ is the identity matrix) and the a priori probability of each class as $\pi_k = 1/K$. At each step, show the current cluster means and represent with colors the current a posteriori probabilities, also known as responsibilities, $p(z_n = k|\mathbf{x}_n, \theta)$. You use `scatter3(x(1,:),x(2,:),x(3,:),50*ones(1,N),[pz; zeros(1,N)]')`, where `pz` is the $K \times N$ vector containing the current a posteriori probabilities for each class. You can repeat the experiment to check the impact of the random initalization of cluster centers.

2. (Adapted from [4]) Consider the following generative directed model

$$z = \text{Bern}(0.5) + 1$$

and

$$x|z = k \sim \mathcal{N}(k\theta, 0.5).$$

Note that z takes values 1 and 2 with equal probability and that the model parameter is θ. We observe a sample $x = 2.75$. We wish to develop the EM algorithm to tackle the problem of ML estimation of the parameter θ given this single observation of $x = 2.75$.

a. Write the likelihood function $p(x = 2.75|\theta)$ of θ for the given observation. Plot the likelihood $p(x = 2.75|\theta)$ for θ in the interval $[-5, 5]$ using MATLAB and check that it is not concave.

b. In the E-step, we need to compute $p(z|x = 2.75, \theta^{old})$. Write the probability $p(z = 2|x = 2.75, \theta^{old})$ as a function of θ^{old} (note that $p(z = 1|x = 2.75, \theta^{old}) = 1 - p(z = 2|x = 2.75, \theta^{old})$).

c. In the M-step, we instead need to solve the problem

$$\text{maximize } E_{z\sim p(z|x=2.75,\theta^{old})}[\ln p(x = 2.75, z|\theta)]$$

over θ. Formulate and solve the problem to obtain θ^{new} as a function of θ^{old}.

3. Load this dataset. It contains $N = 1000$ examples of size $D = 10$ in a $N \times D$ matrix. Note that the alphabet is binary.

a. Observe the dataset by using the command `imagesc(X); colorbar.` What is the main feature of the data?

b. Use PCA to identify the principal component of the data. Note that this requires to set $M = 1$ in PCA. You should use the function `eig` in MATLAB. Since the resulting vector may be complex due to numerical rounding errors, plot the real part. Compare the principal component with your observation of the data at point a.

c. We now wish to use a Bernoulli-Bernoulli RBM to learn the $M = 2$ main features of the data. To this end, perform 100 iterations of the approximate EM algorithm described in class. To initialize, use `W=rand(D,M); a=rand(M,1); b=rand(D,1).` At each iteration, choose randomly and uniformly one sample $\mathbf{x}_n$ from the data set and evaluate the gradient of the log-likelihood with respect to the parameters $\mathbf{W}, \mathbf{a}, \mathbf{b}$ for that point. Apply the computed gradient with learning rate $\eta = 0.1$. To compute the gradient, apply the Contrastive Divergence method explained in the notes: Given the data point $\mathbf{x}_n$, generate $\tilde{\mathbf{z}}_n$, then again $\mathbf{x}'_n$ and finally $\mathbf{z}'_n$. Plot one of columns of $\mathbf{W}$ and compare with the principal component. To facilitate comparison, normalize the column so that the norm of the column is 1 as for the principal component.

4. (Adapted from [4]) Consider a generative undirected and unstructured model $p(\mathbf{x}|\theta)$, where $\mathbf{x} = [x_1, x_2] \in \{0, 1\}^2$. The parameters θ are defined as $\theta_{ij} = \Pr[x_1 = i \text{ and } x_2 = j]$ with $i, j \in \{0, 1\}$. Note that we hence have four parameters and that $\sum_{i,j} \theta_{ij} = 1$.

a. Assume that we have one observation $\mathcal{D} = \{[1, ?]\}$, where "?" denotes the fact that the corresponding variable x_2 is hidden. Write the log-likelihood for the parameters θ given observation $\mathcal{D}$ as a function of $\theta_{00}, \theta_{01}, \theta_{10}, \theta_{11}$.

b. Assume that we now have three independent observations

$$\mathcal{D} = \{[1, 1], [1, ?], [?, 0]\},$$

where "?" denotes the fact that the corresponding variable is hidden. Note that the first observation has no hidden variables, while the second and third have different latent variables. Write the log-likelihood for the parameters θ given observation $\mathcal{D}$ as a function of $\theta_{00}, \theta_{01}, \theta_{10}, \theta_{11}$.

c. Derive the EM algorithm for the maximization of the likelihood obtained at point b. Give the explicit update rule that computes θ^{new} as a function of θ^{old} (Hint: Compute first $p(x_2 = 1|x_1 = 1, \theta)$ and $p(x_1 = 1|x_2 = 0, \theta)$).

References

[1] C. Bishop, *Pattern recognition and Machine Learning*, Springer, 2006.

[2] K. P. Murphy, *Machine learning: a probabilistic perspective*, MIT press, 2012.

[3] Y. Bengio, J. Goodfellow, and Courville. *Deep learning*, MIT Press, available at http://www.iro.umontreal.ca/∼bengioy/dlbook (2015).

[4] D. Barber, *Bayesian reasoning and machine learning*, Cambridge University Press, 2012.

[5] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, M. I. Jordan, “How to Escape Saddle Points Efficiently,” arXiv:1703.00887.

[6] A. Krause, P. Perona, and R. G. Gomes, "Discriminative clustering by regularized information maximization," *Advances in neural information processing systems*, 2010.

[7] R. Salakhutdinov, A. Mnih, and G. Hinton. "Restricted Boltzmann machines for collaborative filtering," in *Proc. ACM International Conference on Machine learning*, 2007.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, Springer, 2001.

6. Probabilistic Graphical Models

1 Introduction

- We have so far dealt with unstructured probabilistic models.
- Specifically for generative directed models, we have assumed a model that can be represented by the Bayesian network, in Fig. 1, where α parametrizes $p(z)$ and β parametrizes $p(x|z)$.

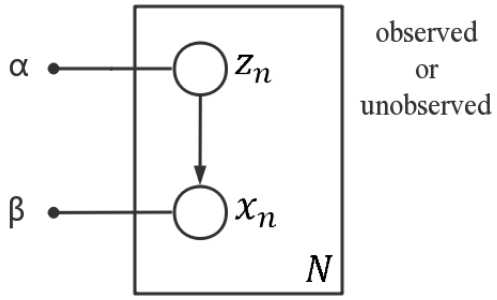


Figure 1: An unstructured generative directed model.

- The Bayesian network in Fig. 1, encodes the factorization of the joint distribution $p(\mathbf{x}, \mathbf{z}|\theta)$ as

$$p(\mathbf{x}, \mathbf{z}|\theta) = \prod_{n=1}^{N} p(z_n|\alpha)p(x_n|z_n, \beta), \tag{1}$$

  where $\theta = [\alpha, \beta]$.
- We have also talked about generative undirected model, which can be represented by the Markov random field in Fig. 2,

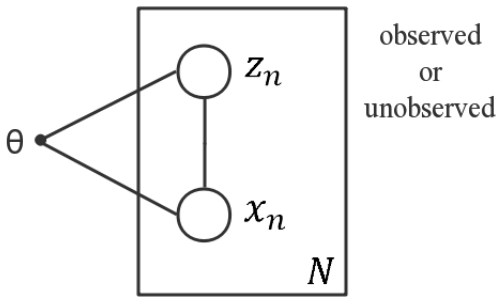


Figure 2: An unstructured generative undirected model.

which encodes the factorization of the joint distribution

$$p(\mathbf{x}, \mathbf{z}|\theta) = \prod_{n=1}^{N} p(x_n, z_n|\theta). \tag{2}$$

- An example of undirected models are energy-based models, such as RBM studied in Chapter 5.
- Bayesian variants of directed and undirected models treat the parameters θ as random variables.
- Unlike the simple scenarios studied so far, in many problems of practical interest, the joint distribution of observed and unobserved (if any) variables has a more complex structure that can be leveraged to define more specific models.
- **Example:** Image denoising via supervised learning.
  - We wish to learn the joint distribution $p(\mathbf{x}, \mathbf{t}|\boldsymbol{\theta})$, where $\boldsymbol{x}$ is the noisy image and $\boldsymbol{t}$ the noiseless image. See Fig. 3 for an illustration.
  - Instead of using general unstructured models as in Fig. 1 or Fig. 2, we can adopt a more structured model that accounts for the following assumptions:
    - neighboring pixels are correlated, while pixels further apart are not directly related;
    - noise acts independently on each pixel.
  - These assumptions are encoded by the Markov random field in Fig. 4.

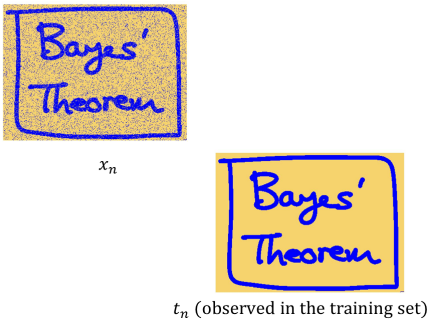


Figure 3: Example of noisy and noiseless images (from [1]).

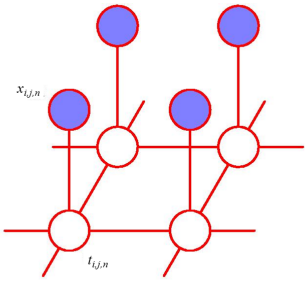


Figure 4: Markov random field representing a structured model for the image denoising problem (from [1]).

- **Example:** Text classification via supervised or unsupervised learning.
  - $t_n \in \{1, ..., T\}$ = topic of a document n (e.g., $t_n \in$ {sport, politics, entertainment}).
  - $\mathbf{x}_n = \begin{pmatrix} x_{1n} \\ \vdots \\ x_{Wn} \end{pmatrix}$ "bag-of-words" model, where $x_{wn} = 1$ if word $w \in \{1, ..., W\}$ is in document n.
  - Instead of using the general model, which contains $(T-1)+T(2^w-1)$ parameters, we can adopt a more structured model that encodes the assumption:
    - once the topic is fixed, the presence of a word is independent on the presence of other words.
  - The resulting model is known as Bernoulli naive Bayes model:

- $t_n \sim \text{Cat}(\boldsymbol{\alpha})$
- $\mathbf{x}_n | t_n \sim \prod_{w=1}^{W} p(x_{wn}|t_n) = \prod_{w=1}^{W} \text{Bern}(x_{wn}|\beta_{wt_n})$

– The Bernoulli naive Bayes model is hence characterized by the parameters

$$\boldsymbol{\theta} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_T \end{bmatrix} \text{ where } \alpha_i = \Pr[t = i\text{th topic}] \text{ with } \sum_{i=1}^{T} \alpha_i = 1$$

– Note that we have $(T-1) + TW \ll (T-1) + T(2^w - 1)$ parameters in the structured model.

– The Bayesian network representing the Bernoulli naive Bayes model is shown in Fig. 5.

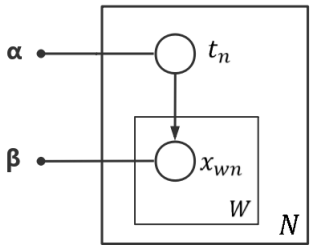


Figure 5: Bayesian network representing the Bernoulli naive Bayes model.

- It should be always kept in mind that the degree of specialization of a model determine a trade-off between bias (measured by $L_p(h^*)$) and approximation error (measured by $L_p(h_\theta) - L_p(h^*)$ and caused by overfitting). In fact, more specialized, or structured, have less parameters to learn. We refer to Chapter 4 for details.
- It is also useful to keep in mind that a model is not necessarily meant to provide an accurate depiction of reality, but to yield a satisfactory performance for the learning task (e.g., classification).
- As a final note, this chapter, we will use a flexible notation for the model variables that is tailored to the different problems.
- Structure in probabilistic model is conveniently represented by means of graphs.
- As assumed structure simplifies, or even makes possible at all, learning.

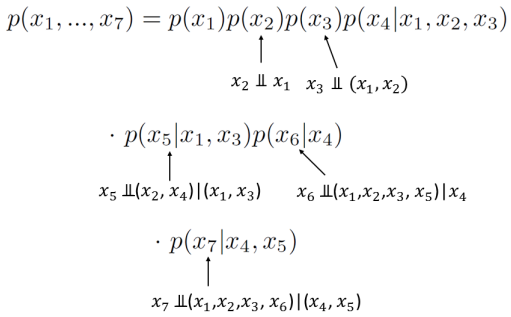


Figure 6: A structured probabilistic model.

- Graphical models encode the structure of a probability model, particularly about conditional independence properties.
- Three main types:
  a. Directed graphical models: Bayesian networks.
  b. Undirected graphical models: Markov random fields.
  c. Factor graphs.

2 Bayesian Networks

2.1 Definitions and Examples

- Bayesian networks (or Bayes nets) encode a probability factorization or, equivalently, a set of conditional independence relationships.
- The starting point is the chain rule for probabilities:

$$p(x_1, ..., x_k) = p(x_1)p(x_2|x_1)...p(x_k|x_1, ..., x_{k-1}), \tag{3}$$

  where the order is arbitrary in (3).
- Factorization (3) applies for general unstructured model. However, as discussed in many important applications, it is convenient to encode more structures in the model.
- For example, we may have the independence relationships encoded by the factorization in Fig. 6, where the notation $A \perp\!\!\!\perp B|C$ indicates that A is independent of B given C.
- The factorization at the previous equation is represented by the Bayesian network in Fig. 7.

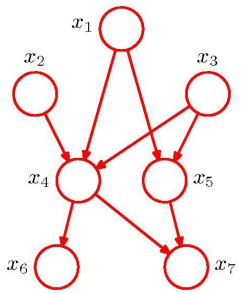


Figure 7: Bayesian network representing the factorization in Fig. 6 (from [1]).

- **Definition:** A Bayesian network is a directed acyclic graph (DAG)[1], in which vertices represent random variables (or vectors), with an associated joint distribution that factorizes as

$$p(\mathbf{x}) = \prod_k p(x_k|x_{\mathcal{P}(k)}) \tag{4}$$

  where $\mathcal{P}(k)$ denotes the set of parents of node k in the graph.

- In a Bayesian network parameters are represented by dots and observed variables are represented by full circles.

- Plates are used to represent replications of a part of the graph.

- **Examples:**

  a. Basic generative model for supervised learning: The factorization is

$$p(\mathbf{x}, \mathbf{t}|\alpha, \beta) = \prod_{n=1}^{N} p(t_n|\alpha)p(x_n|t_n, \beta) \tag{5}$$

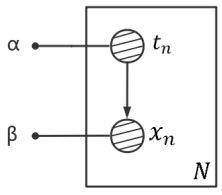


Figure 8: Bayesian network corresponding to (5).

[1] There are no directed cycles, that is, no closed paths following the direction of the arrows.

and the Bayesian network is shown in Fig. 8. The Bayesian version assuming independence of priors corresponds to the following factorization

$$p(\alpha, \beta, \mathbf{x}, \mathbf{t}) = p(\alpha)p(\beta)\left(\prod_{n=1}^{N} p(t_n|\alpha)p(x_n|t_n, \beta)\right) p(t|\alpha)p(x|t, \beta) \tag{6}$$

and Bayesian network is shown in Fig. 9.

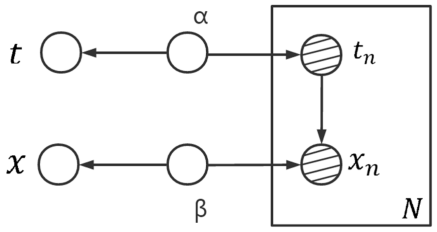


Figure 9: Bayesian network corresponding to (6).

b. Bernoulli naive Bayesian model for text classification: The probability distribution factorizes as

$$p(\mathbf{x}, \mathbf{t}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{n=1}^{N} \mathrm{Cat}(t_n|\boldsymbol{\alpha}) \cdot \left(\prod_{w=1}^{W} \mathrm{Bern}(x_{wn}|\beta_{wt_n})\right) \tag{7}$$

and the corresponding Bayesian network is shown in Fig. 10.

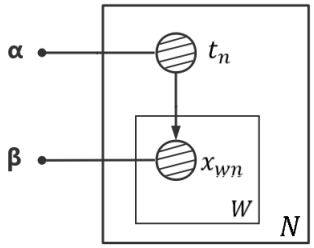


Figure 10: Bayesian network corresponding to (7).

For a Bayesian formulation, one typically makes the global independence assumption:

$$p(\boldsymbol{\alpha}, \boldsymbol{\beta}) = p(\boldsymbol{\alpha})\left(\prod_{t=1}^{T}\prod_{w=1}^{W} p(\beta_{wt})\right), \tag{8}$$

where typical choices are Dirichlet for $p(\boldsymbol{\alpha})$ and Beta for $p(\beta_{wt})$. The corresponding Bayesian network is illustrated in Fig. 11.

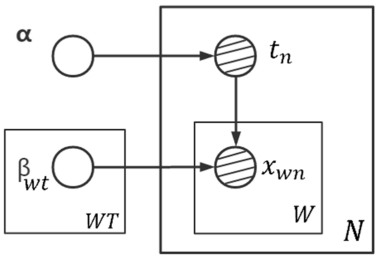


Figure 11: Bayesian network corresponding to (8).

c. Latent Dirichlet model for unsupervised text classification:
$\boldsymbol{\alpha}_n$ = distribution of topic for document $n \sim \text{Dir}(\boldsymbol{\theta}_\alpha)$.
t_{in} = topic for the word in the ith position of document $n \sim \text{Cat}(\boldsymbol{\alpha}_n)$.
β_t = distribution of topics for topic $t \sim \text{Dir}(\theta_\beta)$.
x_{in} = ith word in the document $n \sim \text{Cat}(\beta_{t_{in}})$,

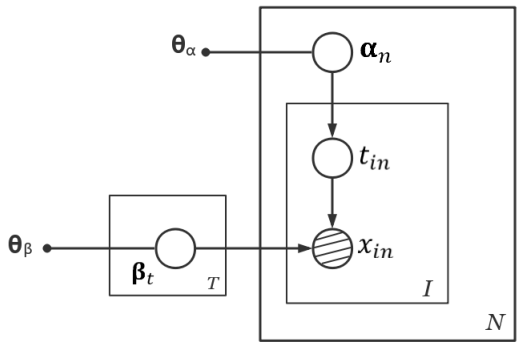


Figure 12: Latent Dirichlet model for unsupervised text classification.

d. QMR-DT: Fig. 13.

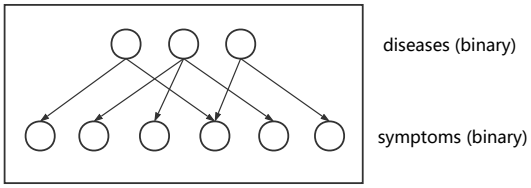


Figure 13: QMR-DT.

e. Hidden Markov models for speech recognition: Fig. 14.

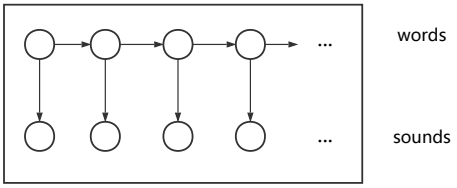


Figure 14: HMM for speech recognition.

f. Monitoring patients in an Intensive Care Unit: Fig. 15.

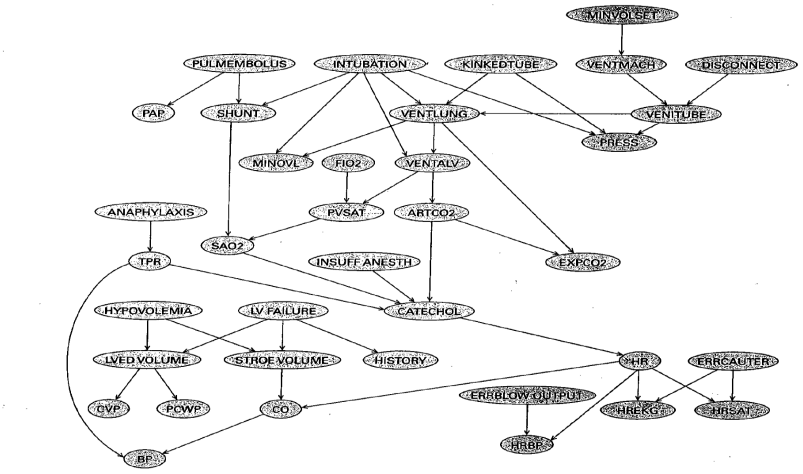


Figure 17.C.1 — The ICU-Alarm Bayesian network.

Figure 15: The ICU-Alarm Bayesian network (from [2]).

2.2 Assessing conditioned independence

- Bayesian network enable the efficient assessment of global conditional dependence properties, that is, they allow to address, but only in the positive, the question: Is a certain subset of variables independent of another conditioned on a third set of variables?
- Given a Bayesian network, we wish to check if the independence condition $A \perp\!\!\!\perp B|C$ holds for arbitrary subset of variables A, B and C.
- d-separation algorithm:
  a. Build a subgraph $\mathcal{G}'$ consisting of all vertices A, B and C and all edges and vertices encountered by moving backwards one or more edges from the vertices in A, B and C.
  b. Build a subgraph $\mathcal{G}''$ from $\mathcal{G}'$ by deleting all edges coming out of the vertices in C.

  – If there is no path, neglecting the directionality of the edges, between A and B, then $A \perp\!\!\!\perp B|C$ holds.
  – If there is, then there is at least one joint distribution that factorizes as for the given Bayesian network for which $A \perp\!\!\!\perp B|C$ does not hold.
- **Examples:**
  a. V-structured: Based on d-separation, as seen in Fig. 16, $A \perp\!\!\!\perp B|C$ not true in general. Specific examples are given in Fig. 17.

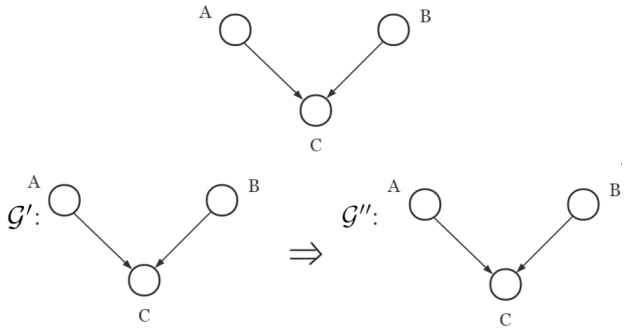


Figure 16:

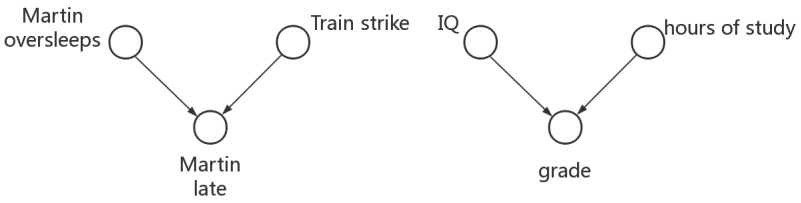


Figure 17: Examples of V-structures.

b. Bayesian prediction: Using d-separation, we can seen from Fig. 18 that $\mathbf{t} \perp\!\!\!\perp t | \boldsymbol{w}$ holds, and hence we have $p(t|\mathbf{t}) = \int p(t, \mathbf{w}|\mathbf{t}) d\mathbf{w} = \int p(\mathbf{w}|\mathbf{t}) p(t|\mathbf{w}) d\mathbf{w}$, that is, we can first compute the posterior $p(\mathbf{w}|\mathbf{t})$ and then perform prediction.

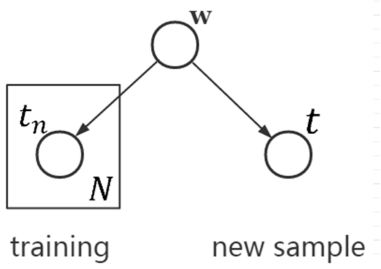


Figure 18: Bayesian prediction.

2.3 Learning Bayesian networks

- Assume that the graph is given (structure learning is also an important problem that we won't consider).
- A Bayesian network can be generally parametrized as

$$p(\mathbf{x}) = \prod_k p(x_k | x_{\mathcal{P}(k)}, \mu_{k|x_{\mathcal{P}(k)}}), \tag{9}$$

  where $\mu_{k|x_{\mathcal{P}(k)}}$ are the parameters defining the conditional distribution $p(x_k|x_{\mathcal{P}(x)})$.
- It is emphasized that in this section, $\mathbf{x}$ represents one data point, possibly including hidden variables.
- In most cases of interest, $p(x_k|x_{\mathcal{P}(k)}, \mu_{k|x_{\mathcal{P}(k)}})$ is in the exponential family (for every $x_{\mathcal{P}(k)}$) or is a GLM.

- Modeling the parameters:
  - Separate parameters: The parameters $\mu_{k|x_{\mathcal{P}(k)}}$ are different for each k and each value of $x_{\mathcal{P}(k)}$.

    ex:
    * Mixture of Gaussian
    * QDA
  - Shared parameters: Some of the parameters $\mu_{k|x_{\mathcal{P}(k)}}$ are tied to be equal across different values of $x_{\mathcal{P}(k)}$ and/or across k. The value of $\mu_{k|x_{\mathcal{P}(k)}}$ may even be independent of $x_{\mathcal{P}(k)}$, e.g., for GLMs.

    ex:
    * Latent Dirichlet model: The distributions β_t of different topics t are characterized by the same parameter $\boldsymbol{\theta}_\beta$.
    * Linear regression or any other GLM: The weight vector is independent of $x_{\mathcal{P}(k)}$.
- Modeling the data:
  - Fully observed: The variables $(x_{1n}, ..., x_{Dn})$ in each data point $\mathbf{x}_n$ are fully observed.

    ex:
    * supervised learning (Chapters 1, 3, 4).
  - Missing data: Some of the variables $(x_{1n}, ..., x_{Dn})$ are missing at least in some data points $\mathbf{x}_n$.

    ex:
    * Unsupervised learning (Chapter 5).

2.3.1 ML learning for the fully observed case with separate parameters

- Given a fully observed data set $\mathcal{D} = \{\mathbf{x}_n\}$, with $\boldsymbol{x}_n = [x_{kn}]$ (discrete), the log-likelihood is given as:

$$\ln p(\mathcal{D}|\boldsymbol{\mu}) = \sum_{n=1}^{N} \sum_{k} \ln p(x_{kn}|x_{\mathcal{P}(k)n}, \mu_{k|x_{\mathcal{P}(k)n}}) \tag{10}$$

$$= \sum_{k} \sum_{n=1}^{N} \ln p(x_{kn}|x_{\mathcal{P}(k)n}, \mu_{k|x_{\mathcal{P}(k)n}}) \tag{11}$$

$$= \sum_{k} \sum_{x_{\mathcal{P}(k)}} \underbrace{\sum_{\substack{n: \\ x_{\mathcal{P}(k)n} = x_{\mathcal{P}(k)}}} \ln p(x_{\mathcal{P}(k)n}, \mu_{k|x_{\mathcal{P}(k)n}})}_{\text{depends only on } \mu_{k|x_{\mathcal{P}(k)}}}. \tag{12}$$

- It follows that the ML estimate decomposes on each variable for each configuration of the parents.

- Problem of data fragmentation: each parameter is estimated based only on a fraction of the data.
- As a concrete example for binary variables, we have

$$\hat{\mu}_{k|x_{\mathcal{P}(k)},ML} = \frac{\sum_{n:x_{\mathcal{P}(k)n}=x_{\mathcal{P}(k)}} x_{nk}}{\left\{n : x_{\mathcal{P}(k)n} = x_{\mathcal{P}(k)}\right\}} \tag{13}$$

- **Example:** Consider the Bayesian shown in Fig. 12 with the observed data

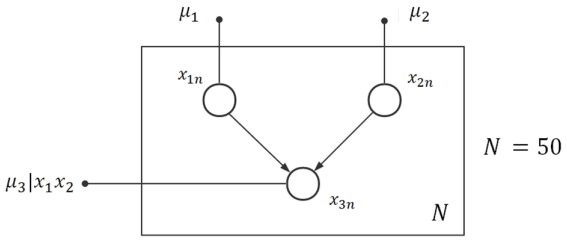


Figure 19: Example of ML learning for the fully observed case with separate parameters.

$$\underset{\#\text{ observations}}{\mathcal{D}} = \left\{ \underbrace{\begin{pmatrix}1\\0\\1\end{pmatrix}}_{10}, \underbrace{\begin{pmatrix}0\\1\\1\end{pmatrix}}_{14}, \underbrace{\begin{pmatrix}1\\1\\0\end{pmatrix}}_{8}, \underbrace{\begin{pmatrix}0\\0\\0\end{pmatrix}}_{12}, \underbrace{\begin{pmatrix}1\\0\\0\end{pmatrix}}_{1}, \underbrace{\begin{pmatrix}0\\1\\0\end{pmatrix}}_{2}, \underbrace{\begin{pmatrix}1\\1\\1\end{pmatrix}}_{1}, \underbrace{\begin{pmatrix}0\\0\\1\end{pmatrix}}_{2} \right\}$$

The maximum likelihood estimates are:

$\hat{\mu}_{1,ML} = \frac{10+8+1+1}{50} = \frac{20}{50} = \frac{2}{5}$

$\hat{\mu}_{2,ML} = \frac{14+8+2+1}{50} = \frac{1}{2}$

$\hat{\mu}_{3|00} = \frac{2}{12+2} = \frac{1}{7}$

$\hat{\mu}_{3|11} = \frac{1}{8+1} = \frac{1}{9}$

$\hat{\mu}_{3|01} = \frac{14}{14+2} = \frac{7}{8}$

$\hat{\mu}_{3|10} = \frac{10}{10+1} = \frac{10}{11}$

2.4 Extensions

- MAP and Bayesian approaches: MAP decompose in a similar way under independence assumptions on the parameters, and the same holds for Bayesian approach.
- Shared parameters:
  - With shared parameters, ML requires aggregate statistics across all variables that share the same parameters.
  - Bayesian approach is more complex in the presence of shared variables.
  - An alternative approach with "soft sharing" is the hierarchical Bayes model (see Fig. 17.11 in [2]).
- Missing data:
  - For the missing data case, learning typically involves the EM algorithm.

3 Markov Random Fields

3.1 Definitions and examples

- **Definition:** A Markov random field is an undirected graph in which vertices represent random variables (or vectors) with associated joint distribution that factorizes as

$$p(\mathbf{x}) = \frac{1}{Z}\prod_c \psi_c(\mathbf{x}_c) \tag{14}$$

  where:
  - c is the index of cliques[2] in the graph.
  - $\psi_c(\mathbf{x}_c) \geq 0$ is the factor or potential associated with clique c (not a probability).
  - $Z = \sum_x \prod_c \psi_c(\mathbf{x}_c)$ is the partition function.
- Each factor encodes the compatibility of the values $\mathbf{x}_c$ in each clique, rather than cause-effect relationships as in Bayesian networks.

[2] A clique is a fully connected subgraph

- **Examples:**

  a. For the undirected graph in Fig. 20, the distribution factorizes as

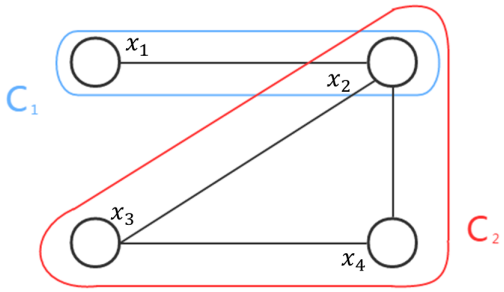


Figure 20: Example of a Markov random field.

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{c_1}(x_1, x_2)\psi_{c_2}(x_3, x_4, x_5). \tag{15}$$

  b. Image denoising: The Markov random field is shown in Fig. 21 and the joint distribution factorizes as

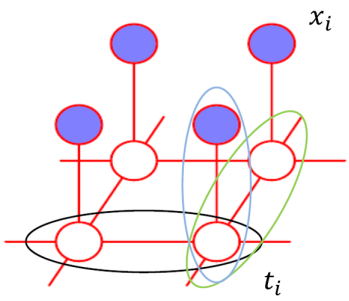


Figure 21: Markov random field of image denoising.

$$p(\mathbf{x}, \mathbf{t}) \propto \prod_{\{i,j\}} \psi_{i,j}(t_i, t_j) \cdot \prod_i \psi_i(t_i, x_i) \tag{16}$$

where $\{i, j\}$ is an edge of the Markov random field. A specific example is the Ising model, where $t_i, x_i \in \{-1, +1\}$, and the potentials are

defined as

$$\psi_{ij}(t_i, t_j) = \exp(-E(t_i, t_j|\beta)) \tag{17}$$

$$\psi_i(t_i, x_i) = \exp(-E(t_i, x_i|\eta)) \tag{18}$$

with

$$E(t_i, t_j|\beta) = -\beta t_i t_j \tag{19}$$

and

$$E(t_i, x_i|\eta) = -\eta t_i x_i \tag{20}$$

From (19), a large $\beta > 0$ yields a large probability where t_i and t_j are equal, while, from (20), a large $\eta > 0$ corresponds to a low noise assumption.

c. Restricted Boltzmann Machine (RBM): The Markov random field is shown in Fig. 22 and the distribution factorizes as

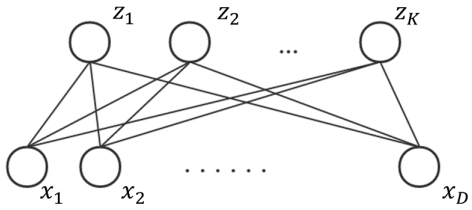


Figure 22: Restricted Boltzmann Machine.

$$p(\mathbf{x}, \mathbf{z}) \propto \prod_{i=1}^{K} \prod_{j=1}^{D} \exp(-E(x_j, z_i|w_{ji}) - E(z_i|a_i) - E(x_j|b_j)), \tag{21}$$

with potentials

$$E(x_j, z_i|w_{ji}) = -w_{ji} x_j z_i \tag{22}$$

$$E(x_j|b_j) = -b_j x_j \tag{23}$$

$$E(z_i|a_i) = -a_i z_i. \tag{24}$$

- The partition function Z is typically hard to compute due to the need to integrate over a large domain.
- It is often convenient to work with the unnormalized distribution and perform normalization at the end.

- As seen, we typically have energy-based models

$$\psi_c(\mathbf{x}_c) = \exp(-\underbrace{E_c(\mathbf{x}_c)}_{\text{energy function}}) \tag{25}$$

and hence

$$-\ln p(\mathbf{x}) = \sum_c E_c(\mathbf{x}_c) + \ln Z \tag{26}$$

$$= \sum_c E_c(\mathbf{x}_c) + A \tag{27}$$

where A is the log-partition function.

- In energy based-models, maximum probability states are those with minimal energy.

3.2 Assessing conditional independence

- Markov random fields, in a manner similar to Bayesian networks, enable the assessment of conditional independence properties.
- In fact, this is easier with Markov random fields thanks to the Hammersley–Clifford theorem:

  If $\psi_c(\mathbf{x}_c) > 0$ for all cliques c, conditional independence $A \perp\!\!\!\perp B|C$ can be tested as follows:

  - Eliminate all variables in C and connected edges;
  - If there is no path between A and C, then $A \perp\!\!\!\perp B|C$ holds;
  - If there is, then there is at least one joint distribution that factorizes as for the Markov field for which $A \perp\!\!\!\perp B|C$ does not hold.

- **Example:** Fig. 23

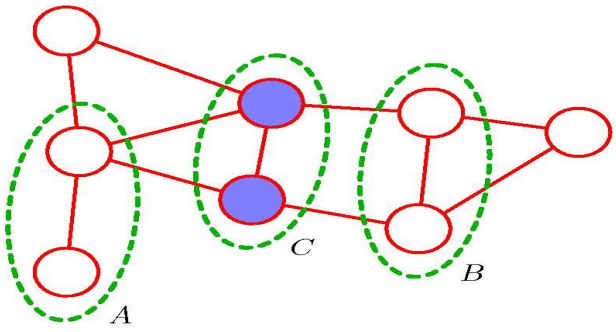


Figure 23: In this example we have $A \perp\!\!\!\perp B|C$ (from [1]).

- We will assume the standard log-linear model

$$E_c(\mathbf{x}_c|\boldsymbol{\eta}_c) = -\boldsymbol{\eta}_c^T \mathbf{u}_c(\mathbf{x}_c) \tag{28}$$

where $\boldsymbol{\eta}_c$ are the (natural) parameters associated with cluster c and $\mathbf{u}_c(\mathbf{x}_c)$ are the corresponding sufficient statistics.

- **Example:** RBMs are log-linear models with

$$E(x_j, z_i|w_{ji}) = -w_{ji}x_j z_i \tag{29}$$

$$E(x_j|b_j) = -b_j x_j \tag{30}$$

$$E(z_i|a_i) = -a_i z_i \tag{31}$$

and hence the natural parameters are $\{w_{ji}, b_j, a_i\}$, while the sufficient statistics are $\{x_j, z_i, x_j z_i\}$.

- Energy-based models with log-linear factor belong to the exponential family.

3.3 Learning Markov random fields

- Learning Markov random fields is made complicated by the partition function. In fact, the log-partition function is

$$A(\boldsymbol{\eta}) = \ln \left(\sum_{\mathbf{x}} \exp(-\sum_c E_c(\mathbf{x}_c|\boldsymbol{\eta}_c)) \right) \tag{32}$$

which depends on all parameters $\boldsymbol{\eta}$. Therefore, it is not possible to carry out separately the estimate of the parameters $\{\boldsymbol{\eta}_c\}$, even in the fully observed case with separate parameters.

- Nevertheless, in the fully observed case the ML problem

$$\underset{\boldsymbol{\eta}}{\text{minimize}} - \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\boldsymbol{\eta}) = \sum_{n=1}^{N} \sum_c E_c(\mathbf{x}_{cn}|\boldsymbol{\eta}_c) + A(\boldsymbol{\eta}) \tag{33}$$

is convex and can be tackled by means of gradient algorithm as we have seen in Chapter 2.

- In fact, following the same calculation in Chapter 2, for the case of separate parameters, we can write

$$\nabla_{\boldsymbol{\eta}_c} \ln p(\mathbf{x}_n|\boldsymbol{\eta}) = \mathbf{u}_c(\mathbf{x}_{cn}) - E[\mathbf{u}_c(\mathbf{x}_c)]. \tag{34}$$

- The main issue is the computation of $E[\mathbf{u}_c(\mathbf{x}_c)]$, which typically requires Monte Carlo approximations as seen for RBM.

- As for Bayesian networks, learning with missing data typically requires the use of the EM algorithm.

3.4 Converting a Bayesian network to a Markov random field

- Given $p(\mathbf{x}) = \prod_k p(x_k|x_{\mathcal{P}(k)})$ (Bayesian network), we can define potential functions

$$\psi_k(x_k, x_{\mathcal{P}(k)}) = p(x_k|x_{\mathcal{P}(k)}) \tag{35}$$

  to obtain the factorization

$$p(\mathbf{x}) = \prod_k \psi_k(x_k, x_{\mathcal{P}(k)}) \quad (Z = 1) \tag{36}$$

- Each clique contains x_k and its parents $x_{\mathcal{P}}(k)$.
- It follows that the Markov random fields is obtained by following the procedure:

  1. Connect all pairs of parents by one undirected edge ("moralization").

  2. Make all edges undirected.

- **Example:** Fig. 24.

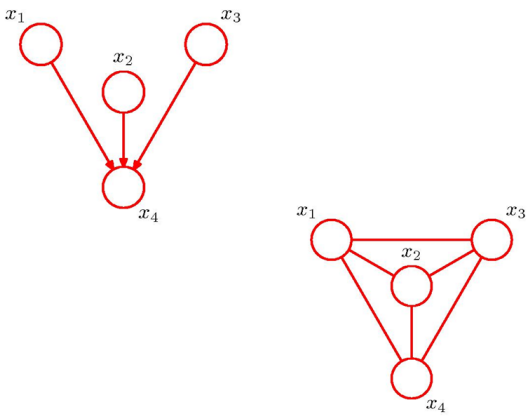


Figure 24: Example of conversion of a Bayesian network into a Markov random field (from [1]).

3.4.1 Directed vs. undirected graphs

- Some conditional independence properties can be expressed by BN or MRF but not by both
- **Examples:**

a. Fig. 25.

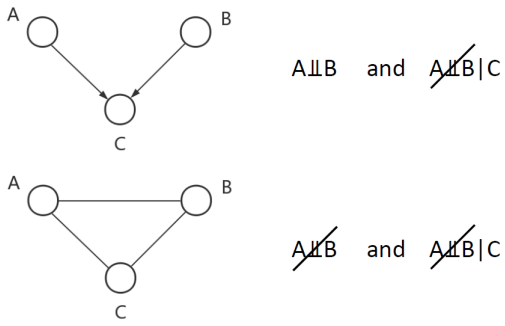


Figure 25: Example of conditional independence properties.

b. No directed graphs exists that satisfies the conditional independence properties for the Markov random fields in Fig. 26.

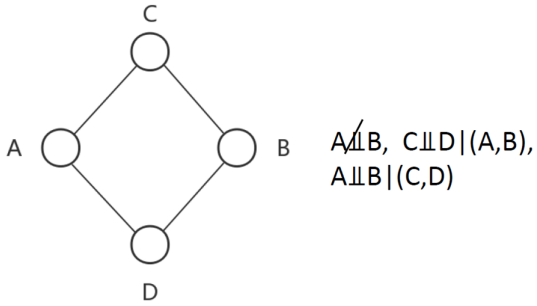


Figure 26: Example of conditional independence properties.

4 Bayesian Inference in Probabilistic Graphic Models

- Bayesian inference amounts to the computation of

$$p(\text{unobserved}|\text{observed}).$$

- We have encountered this operation in a number of problems:
  - Learning step with hidden variables:
    - In the E-step of EM, we need to compute

$$p(z_n|x_n,\theta) = \frac{p(x_n,z_n|\theta)}{p(x_n|\theta)} = \frac{p(x_n,z_n|\theta)}{\sum_{z_n} p(x_n,z_n|\theta)} \tag{37}$$

  - Prediction step in generative models for supervised learning:
    - Given a new sample x, we need to compute

$$p(t|x,\hat{\theta}) = \frac{p(x,t|\hat{\theta})}{p(x|\hat{\theta})} = \frac{p(x,t|\hat{\theta})}{\sum_{t} p(x,t|\hat{\theta})} \tag{38}$$

      for frequentist approaches, and

$$p(\theta|\mathcal{D}) = \frac{p(|\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(|\mathcal{D}|\theta)p(\theta)}{\sum_{\theta} p(|\mathcal{D}|\theta)p(\theta)} \tag{39}$$

      for the Bayesian approach.

- Bayesian inference hence requires to marginalize over (all or) some of the unobserved variables.
- The complexity of this step is exponential in the number of unobserved variables and hence it can be prohibitive.
- The structure encoded in probabilistic graphic models can help reduce this complexity.
- **Example:** Hidden Markov Model (HMM): The Bayesian network of an HMM is shown in Fig. 27. Inferring the hidden variables requires to compute

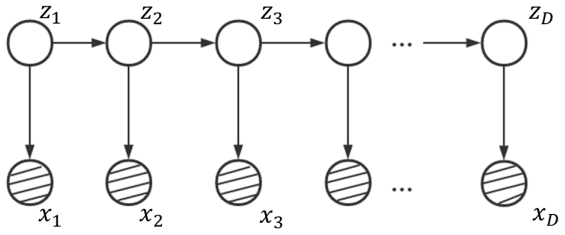


Figure 27: Hidden Markov Model (HMM).

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{x})} \quad \text{with} \quad p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x},\mathbf{z}). \tag{40}$$

The complexity of computing $p(\mathbf{x})$ is $\mid \mathcal{Z} \mid^D$. For the same HMM, we may instead want to predict an individual hidden variable z_k by computing

$$p(z_k|\mathbf{x}) = \sum_{\mathbf{z} \setminus z_k} p(\mathbf{z}|\mathbf{x}) = \frac{\sum\limits_{\mathbf{z} \setminus z_k} p(\mathbf{z},\mathbf{x})}{p(\mathbf{x})}, \tag{41}$$

where $p(\mathbf{x}) = \sum\limits_{z_k} \sum\limits_{\mathbf{z} \setminus z_k} p(\mathbf{x},\mathbf{z})$. The complexity is still $\mid \mathcal{Z} \mid^D$.

- For joint distributions defined by a probabilistic graphic model

$$p(\mathbf{x}) = \frac{1}{Z} \prod_c \psi_c(\mathbf{x}_c), \tag{42}$$

we can state the Bayesian inference problem in general form as the so-called sum-product inference task

$$\sum_{\mathbf{z}} p(\mathbf{x}) = \sum_{\mathbf{z}} \prod_c \psi_c(\mathbf{x}_c), \tag{43}$$

where $\mathbf{z}$ contains a subset of the variables in $\mathbf{x}$ representing a subset or all of the unobserved variables.

- When the undirected graph describing the joint distribution is a tree[3], the complexity of sum-product inference becomes exponential only in the maximum number of variables in each factor, also known as treewidth of the graph.

- In this case, the sum-product inference problem can be exactly solved via message passing, or belief propagation, over an associated factor graph.

- **Example:** For the HMM in Fig. 27, we have the associated undirected graph in Fig. 28 that is a tree, and the corresponding factorization can be written as

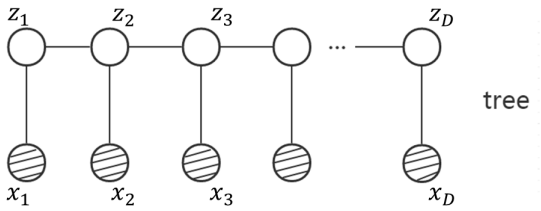


Figure 28: HMM as a Markov random field.

[3]There is only one path between any pairs of nodes (no loops).

$$p(\mathbf{x}, \mathbf{z}) \propto \prod_i \psi(z_i, z_{i+1}) \psi(x_i). \tag{44}$$

Bayesian inference has complexity of the order $\mid \mathcal{Z} \mid^2$.

- See Fig. 29 for an illustration of the concept of message passing.

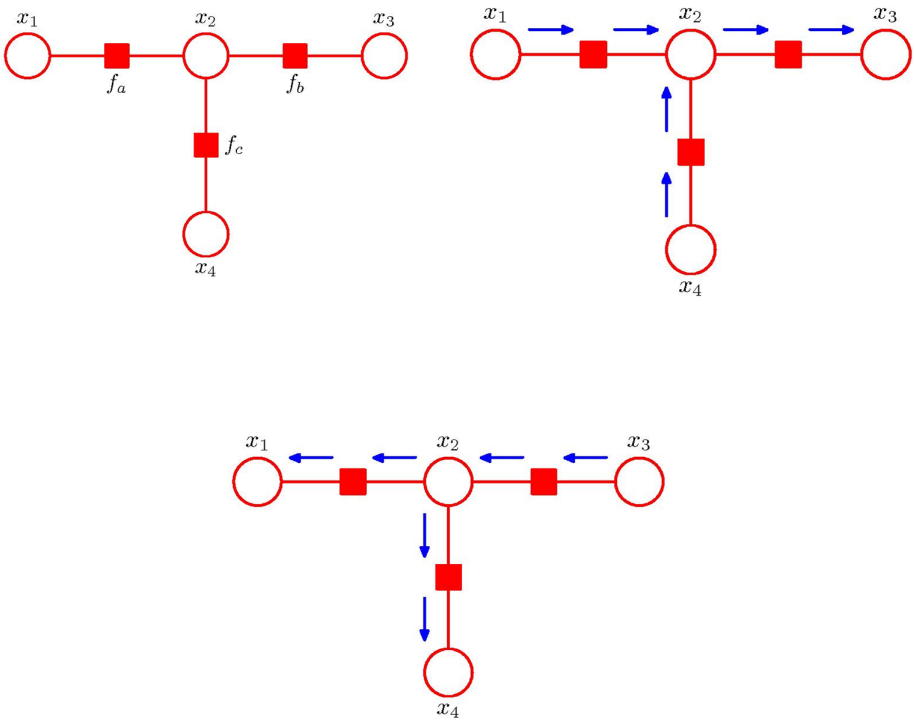


Figure 29: Illustration of message passing (from [1]).

- When the undirected graph is not a tree, one can use the junction tree algorithm for exact Bayesian inference, The idea is to group variables together so that the resulting graph is a tree. The complexity depends on the treewidth of the resulting graph.
- It is also possible to use message passing when the graph has loops. This is known as loopy believe propagation. This will be further discussed in the next Chapter.

5 Problems

1. Consider a data set in which each record n concerns a student taking some class. Specifically, each data point n includes the following variables:

- $D_n \in \{0, 1\}$ = Difficulty of the class (easy, 0, or difficult, 1)
- $I_n \in \{0, 1\}$ = Intelligence of the student (average, 0, or high, 1)
- $G_n \in \{0, 1\}$ = Grade of the student in the class (low, 0, or high, 1)
- $S_n \in \{0, 1\}$ = SAT score of the student (low, 0, or high, 1)
- $L_n \in \{0, 1\}$ = Letter of recommendation obtained by the student from the professor teaching the class (negative, 0, or positive, 1).

a. How many parameters define the joint distribution $p(D, I, G, S, L)$ if we don't impose any structure on it?

b. Propose conditional independence assumptions to simplify the model and specify the corresponding Bayesian network and the corresponding factorization of the joint distribution.

c. How many parameters does your proposed model have?

d. Read the section in the notes about assessing conditional independence properties in Bayesian networks (not covered in class). Study the d-separation algorithm. This algorithm allows to determine if any given conditional independence property holds in the model. More precisely, the output of this algorithm is one of the following: 1) Yes, the given conditional independence relationship holds; 2) No, the conditional independence does not hold in general, although it may hold for some specific choice of the conditional probabilities. Using d-separation, check if the following conditional independence assumptions hold in your proposed model:

- D_n independent of I_n?
- D_n independent of I_n conditioned on G_n?
- L_n independent of S_n?
- L_n independent of S_n given G_n?

For each relationship, explain if you model yields interpretable results. In other words, do the outcomes of d-separation comply with your intuition? If you are unsure about point b, you can use the following factorization $p(D, I, G, S, L) = p(D)p(I)p(G|D, I)p(S|I)p(L|G)$.

2. Consider a naive Bayes model

$$p(t, \mathbf{x}|\theta) = \text{Bern}(t|p) \prod_{i=1}^{D} \text{Bern}(x_i|\mu_t),$$

where the parameters are $\theta = (p, \mu_1, \mu_0)$.

a. Draw the directed acyclic graph that describes the corresponding Bayesian network assuming N i.i.d. data points $(t_n, \mathbf{x}_n)$.

b. Write the equations needed to compute ML estimate of θ for fully observed data.

c. For a given estimate $\hat{\theta}$, evaluate the optimum prediction t given a new data point x based on the learnt distribution $p(t, x|\hat{\theta})$.

d. Use the function [x,t,xtest,ttest]=datagenhw11(N); to generate a training set of N data points (x,t) and a N test points (xtest,ttest). Note that $D = 10$. Fix $N = 1000$. Evaluate the ML estimate in MATLAB using the training set.

e. Evaluate the training and test errors for the optimal prediction derived at step c. given the ML estimate at step c.

f. Repeat for different values of N and comment.

3. Consider the model

$$\begin{aligned} z &\sim \text{Bern}(0.5) \\ x|z &\sim \text{Bern}(0.8z) \\ y|z &\sim \text{Bern}(1 - 0.8z) \end{aligned}$$

and compute $p(y = 1|x = 0)$.

4. Consider the Hidden Markov Model

$$\begin{array}{ccccc} X_1 & \rightarrow & X_2 & \rightarrow & X_3 \\ \downarrow & & \downarrow & & \downarrow \\ Y_1 & & Y_2 & & Y_3 \end{array}$$

Using d-separation, determine if the following independence relationships hold: (a) Y_1 independent of Y_2; (b) Y_1 independent of Y_2 conditioned on X_2; (c) Y_1 independent of Y_2 conditioned on X_3.

References

[1] Bishop, *Pattern recognition and Machine Learning*. Springer, 2006.

[2] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

7. Approximate Inference

- We have seen that Bayesian inference, that is, the calculation of the distribution $p(z|x)$ of unobserved variable z given the observation x plays a key role in learning (EM, Bayesian approach) and in prediction.
- We have also seen that computing $p(z|x)$ requires the solution of a sum-product inference task

$$p(x) = \sum_z p(x, z), \tag{1}$$

  where $p(x, z)$ is a factorized joint probability distribution.
- The complexity is exponential in the treewidth of the corresponding Markov field model.
- What to do when this complexity is excessive? We can resort to approximate inference via
  - Monte Carlo methods
  - Variational inference.

1 Monte Carlo methods: Importance Sampling

- The general idea behind Monte Carlo methods is replacing the ensemble average in (1) with an empirical average.
- We have:

$$p(x) = \sum_z p(z)p(x|z) = E_{z\sim p(z)}[p(x|z)]. \tag{2}$$

- If it is easy to generate $z_m \underset{i.i.d.}{\sim} p(z), m = 1, \ldots, M$, then we can compute

$$p(x) \approx \frac{1}{M} \sum_{m=1}^{M} p(x|z_m) \xrightarrow[M\to\infty]{} p(x). \tag{3}$$

- However, this is generally not easy. One may instead resort to a simplified distribution $q(z)$, typically having convenient factorization properties.
- We have:

$$\begin{aligned} p(x) &= \sum_z p(z)p(x|z)\frac{q(z)}{q(z)} \\ &= \sum_z q(z)\frac{p(z)}{q(z)}p(x|z) \\ &= E_{z\sim q(z)}\Big[\frac{p(z)}{q(z)}p(x|z)\Big]. \end{aligned} \tag{4}$$

- Hence, we can use the Importance Sampling estimate

$$\begin{aligned} &z_m \underset{i.i.d.}{\sim} q(z),\ m = 1, \ldots, M, \\ &p(x) = \frac{1}{M} \sum_{m=1}^{M} \frac{p(z_m)}{q(z_m)} p(x|z_m) \xrightarrow[M\to\infty]{} p(x). \end{aligned} \tag{5}$$

- The variance of this estimate depends on how well $q(z)$ approximates $p(z)$.
- We can also select $q(z)$ as a function of x and hence we can write $q(z|x)$.
- The approach requires to be able to compute $p(z)$, which may not be easy.
- Importance Sampling is more often used in combination with variational inference, to be discussed next.

2 Variational Inference

- The general idea behind Variational inference is replacing the ensemble average in (1) with an optimization.
- From Chapter 5, we know that

$$\ln p(x) = \ln \sum_z p(x,z) \geq \mathcal{L}(q), \tag{6}$$

for any $q(z)$, possibly dependent on x, where $\mathcal{L}(q)$ is the ELBO. Note that we have suppressed the dependence on the parameters Q that are fixed.

- Recall that

$$\begin{aligned} \mathcal{L}(q) &= E_{z\sim q(z)}[\ln p(x,z)] + H(q) \\ &= -\mathrm{KL}(q(z)||p(x,z)), \end{aligned} \tag{7}$$

and that (6) holds with equality if $q(z) = p(z|x)$.

- It follows that we have the equality

$$\ln p(x) = \max_{q(z)} \mathcal{L}(q), \tag{8}$$

which yields as a solution $q(z) = p(x|z)$.

- Solving this problem directly is generally prohibitive.
- The key idea of variational inference is to choose a parametric form for $q(z)$ that enables the solution of problem (8):

$$\begin{aligned} \ln p(x) &= \max_{q(z)\in\mathcal{Q}} \mathcal{L}(q) \\ &= \min_{q(z)\in\mathcal{Q}} \mathrm{KL}(q(z)||p(x,z)), \end{aligned} \tag{9}$$

where $\mathcal{Q}$ is a specific subset of distribution.

- The optimization above is known as I-projection.
- The result of the optimization depends on x and is known as variational posterior or inference network.
- The variational posterior directly provides an estimate of $p(z|x)$.
- Example (I-projection vs. M-projection):
  - Fix $p(\mathbf{z})$ to be bi-modal Gaussian (blue lines in Fig. 1).
  - $q(\mathbf{z})$ is a conventional Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ to be designed.
  - I-projection:
    - $\underset{\boldsymbol{\mu},\boldsymbol{\Sigma}}{\text{minimize}}\ \mathrm{KL}(q||p)$;
    - Tends to be mode-seeking and exclusive, since $q(\mathbf{z})$ determines the support in which $p(\mathbf{z})$ and $q(\mathbf{z})$ are compared (zero-forcing);
    - Tends to be more accurate where $q(\mathbf{x})$ is larger;
    - See Fig. 1 and Fig. 2.

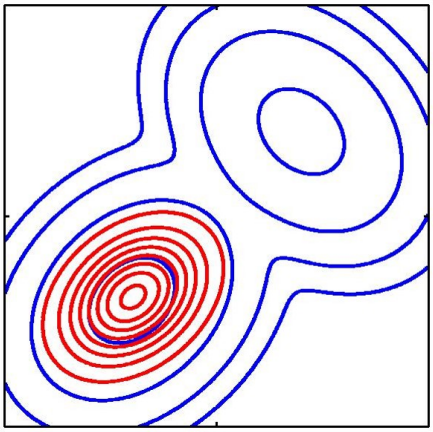


Figure 1: Blue contour lines define a mixture of Gaussian distribution p and red contour lines correspond to distribution obtained q via an I-projection of p onto the space of bivariate Gaussian distributions (from [3]).

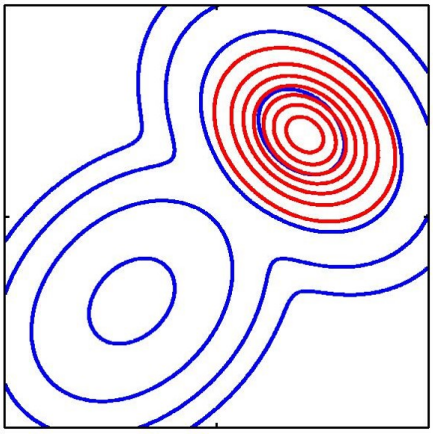


Figure 2: Blue contour lines define a mixture of Gaussian distribution p and red contour lines correspond to distribution obtained q via an I-projection of p onto the space of bivariate Gaussian distributions (from [3]).

- M-projection:
  - minimize $\underset{\boldsymbol{\mu},\boldsymbol{\Sigma}}{\text{minimize}}\ \mathrm{KL}(p||q)$;
  - Tends to avoid having $q(\mathbf{z}) = 0$ when $p(\mathbf{z}) \neq 0$, and hence it tends to span the support of $p(\mathbf{z})$ (inclusive or zero avoiding);
  - The output is shown in Fig. 3;
  - It can be shown that the solution is $\boldsymbol{\mu} = E[\mathbf{x}]$ and $\boldsymbol{\Sigma} = E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T]$ (moment matching);
  - In the Appendix, it is shown that the M-projection with $q(\mathbf{z})$ from the exponential family $q(\mathbf{z}) = \frac{1}{Z}\exp(\boldsymbol{\eta}^T\mathbf{u}(\mathbf{z}))$ yields natural parameters $\boldsymbol{\eta}^*$ such that

$$E_p[\mathbf{u}(\mathbf{z})] = E_{q_{\boldsymbol{\eta}^*}}[\mathbf{u}(\mathbf{z})]. \tag{10}$$

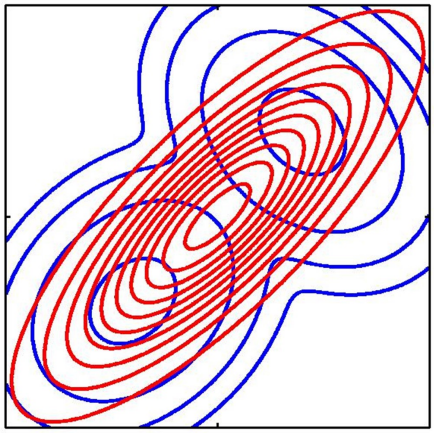


Figure 3: Blue contour lines define a mixture of Gaussian distribution p and red contour lines correspond to distribution obtained q via an M-projection of p onto the space of bivariate Gaussian distributions (from [3]).

- Remark: I-projection and M-projection can be both generalized by using the notion of α-divergence as discussed in the Appendix (see [4]).

2.1 Mean field variational inference

- An important example of variational inference methods is mean field, which assumes the factorization $q(\mathbf{z}) = \prod_j q(z_j)$.

  Example: See Fig. 4 and Fig. 5.

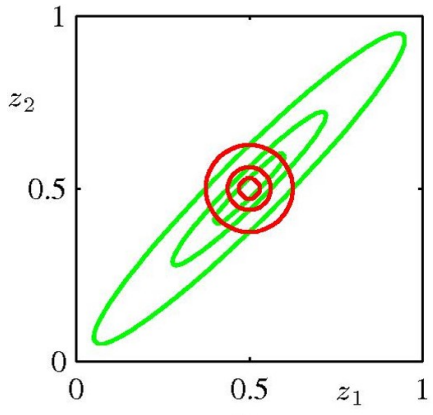


Figure 4: Green contour lines correspond to a correlated Gaussian distribution $p(z_1, z_2)$, and red contour lines represent the mean field approximation where $q(z_i) = \mathcal{N}(z_i|\mu_i, \sigma_i^2),\ i = 1, 2$ (from [3]).

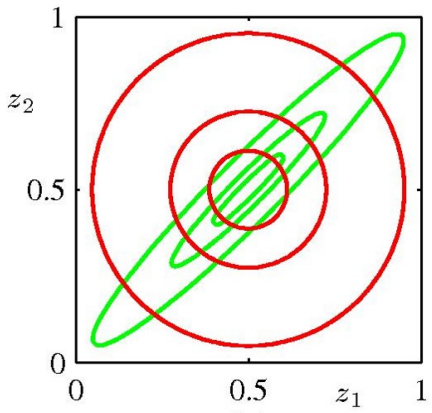


Figure 5: Green contour lines correspond to a correlated Gaussian distribution $p(z_1, z_2)$, and red contour lines represent M-projection of $p(z_1, z_2)$ into the space $\mathcal{Q} = \{p(z_1, z_2) = \prod_i \mathcal{N}(z_i|\mu_i, \sigma_i^2)\}$ (from [3]).

- The mean field method performs the I-projection for one z_j at a time, fixing the factors $\{q(z_j)\}$ for $i \neq j$. This corresponds to tackling the I-projection problem using coordinate descent.
- This yields the problem

$$\begin{aligned}
\underset{q_j}{\text{minimize}}\ & \mathrm{KL}\big(\prod_i q_i(z_i)||p(\mathbf{x},\mathbf{z})\big) \\
&= -E_{\mathbf{z}\sim q(\mathbf{z})}[\ln p(\mathbf{x},\mathbf{z})] - H(q) \\
&= -E_{z_j\sim q(z_j)}[\underbrace{E_{z_{i\neq j}\sim\prod_{i\neq j} p(z_j)}[\ln p(\mathbf{x},\mathbf{z})]}_{\triangleq E_{i\neq j}[\ln p(\mathbf{x},\mathbf{z})]}] - \sum_i H(q(z_i)) \\
&= -E_{z_j\sim q(z_j)}[\ln\exp(E_{i\neq j}[\ln p(\mathbf{x},\mathbf{z})])] - \sum_i H(q(z_i)).
\end{aligned} \tag{11}$$

- Neglecting constants independent of $q(z_j)$, this problem becomes:

$$\begin{aligned}
\underset{q_j}{\text{minimize}}\ & -E_{z_j\sim q(z_j)}[\ln\exp(E_{i\neq j}[\ln p(\mathbf{x},\mathbf{z})])] - H(q(z_j)) \\
&= \mathrm{KL}(q(z_j)||\exp(E_{i\neq j}[\ln p(\mathbf{x},\mathbf{z})])),
\end{aligned} \tag{12}$$

  whose solution is

$$q(z_j) = \frac{\exp(E_{i\neq j}[\ln p(\mathbf{x},\mathbf{z})])}{\sum_{z_j}\exp(E_{i\neq j}[\ln p(\mathbf{x},\mathbf{z})])}. \tag{13}$$

- The equation (13) can be solved by cycling through the factors $q(z_j)$ or choosing them randomly.
- Convergence to a stationary point is guaranteed.
- If $p(\mathbf{x},\mathbf{z}) = \frac{1}{Z}\prod_c \psi_c(\mathbf{x}_c,\mathbf{z}_c)$ according to a given probabilistic graphical model, the mean field equation can be written as

$$\begin{aligned}
q(z_j) &\propto \exp(E_{i\neq j}[\sum_c \ln\psi_c(\mathbf{x}_c,\mathbf{z}_c)]) \\
&\propto \exp(\sum_{c:z_j\in\mathbf{z}_c} E_{i\neq j}[\ln\psi_c(\mathbf{x}_c,\mathbf{z}_c)]).
\end{aligned} \tag{14}$$

- To compute $q(z_j)$ we hence need only the $q(z_j)$ factors for the variables z_i that belong to the same cliques as z_j. This enables implementations by means of local message passing.
- Example (Ising model):

– $p(\mathbf{x}, \mathbf{z}) = \frac{1}{Z}\exp(-E(\mathbf{x}, \mathbf{z}))$
– $E(\mathbf{x}, \mathbf{z}) = -\sum_{\{i,j\}} z_i z_j - \sigma \sum_i x_i z_i$

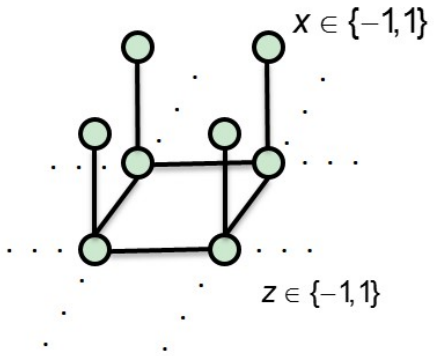


Figure 6: Markov random field for the Ising model.

– Mean field assumes $q(\mathbf{z}) = \prod_i q(z_i)$ and obtain

$$\begin{aligned} q(z_j) \propto & \exp\Big(\sum_{\{i,j\}} E_i[z_i z_j] + \sigma x_j z_j\Big) \\ = & \exp\Big(z_j\Big(\sum_{\{i,j\}} \mu_i + \sigma x_j\Big)\Big), \end{aligned} \tag{15}$$

and hence

$$q(z_j = 1) = E_q[z_j] = \frac{1}{1 + \exp(-2(\sum_{\{i,j\}} \mu_i + \sigma x_j))}, \tag{16}$$

where $\mu_i = q(z_i = 1) - q(z_i = -1) = 2q(z_i = 1) - 1$.

– Final estimate:

$$q(z_j = 1) \underset{z_j=-1}{\overset{z_j=1}{\gtrless}} \frac{1}{2}. \tag{17}$$

– See Fig. 7.

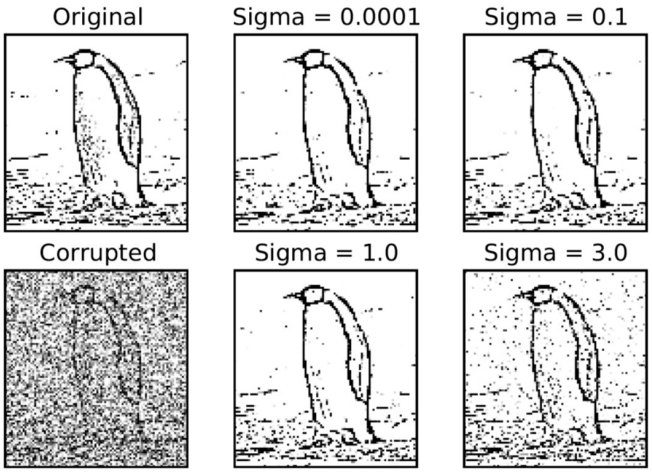


Figure 7: Denoising via the mean field method with different values of σ (@Salimbeni 2015).

- Mean field assumes a fully factorized distribution $q(z_j)$. It is possible to develop methods that are based on more general forms for $q(\mathbf{z})$ that captures some of the real independencies in $p(\mathbf{x}|\mathbf{z})$. This can be done by choosing a probabilistic graphical model for $q(\mathbf{z})$ and using the corresponding factorization

$$q(\mathbf{z}) = \frac{1}{Z}\prod_{c'}\phi_{c'}(z_{c'}). \tag{18}$$

- There is also a class of methods based on "locally consistent" $q(\mathbf{z})$ that yield methods such as loopy belief propagation.
- We refer to [2] for further discussion.

2.2 Stochastic gradient-based variational inference

- The mean field method made specific assumptions about the factorization of the variational distribution $q(z)$ and no assumption about the parametric form of $q(z)$. Alternatively, it is possible to develop methods that learn a generic distribution $q(z|\theta)$ parametrized by vector θ under the following basic assumptions:
  - It is easy to draw samples $z \sim q(z|\theta)$
  - We can compute the gradient $\nabla_\theta \ln q(z|\theta)$
- The methods can scale to large data sets and are based on stochastic gradient.
- The key observation is that the gradient of the KL divergence can be written as

$$\nabla_\theta \mathrm{KL}(q(z|\theta)||p(x,z)) = \mathrm{E}_{z\sim q(z|\theta)}\left[\nabla_\theta \ln p(z|\theta) g(x,z|\theta) + \nabla_\theta g(x,z|\theta)\right], \tag{19}$$

  where

$$g(x,z|\theta) = \ln q(z|\theta) - \ln p(x,z) \tag{20}$$

  and we used the equality $\nabla_\theta \ln q(z|\theta) = \nabla_\theta q(z|\theta)/q(z|\theta)$ (see [1] for details).
- The expectation above can be computed using Monte Carlo methods by drawing one or more samples $z \sim q(z|\theta)$. The resulting gradients are also known as likelihood ratio or REINFORCE gradients. In practice, these gradients have high variance and control variates need to be introduced [1].
- Alternatively, the reparametrization trick method can be used to compute the gradient above. This approaches requires $\ln p(x,z)$ with respect to z [1].
- Amortized variational inference: While the approach discussed so far fits a different $q(z|\theta)$ for each data point, it is faster to learn a single parameter θ for all x. This can be done by adopting a model $q(z|x,\theta)$ with a single θ for all x. Stochastic gradient methods can be devised in a manner similar to the discussion above [1].
- It is possible to choose parametric models such as (18) that factorize in specific ways that encode additional information.

2.3 Variational EM

- When variational inference via M-projection is used for the E step of the EM algorithm, the resulting scheme is known as variational EM.
- An example is the variational autoencoder, which is a variational EM method based on amortized variational inference.
- When the M-projection is used instead, we obtain the wake-sleep algorithm.

3 Problems

1. Load this dataset in MATLAB. It contains a single noisy binary image x.

a. Show this image using imagesc.

b. Implement the mean field scheme based on the Ising model. Please note that you need to first convert the image using the alphabet {-1,1}. You can initialize the μ values to equal the image x itself (after conversion to {-1,1} alphabet) and you can set $\sigma = 0.1$. Plot the estimated image after each iteration. To estimate the image, select 1 if the corresponding probability $q(z_j)$ is larger than $1/2$ and 0 otherwise.

c. What happens if you increase σ? Justify your observation.

d. What happens if you make σ negative? Justify your observation.

References

[1] Shakir Mohamed David Blei, Rajesh Ranganath. Variational inference: Foundations and modern methods. Technical report.

[2] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principle and Technoloty.* The MIT Press, Cambridge, Massachusetts, 2009.

[3] Christopoher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, Cambridge, United Kingdom, 2006.

[4] Tom Minka et al. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

Appendix: M-projection with exponential family

- Consider the problem minimize $\mathrm{KL}(p||q_{\boldsymbol{\eta}})$, where $q(\mathbf{x}|\boldsymbol{\eta}) = \frac{1}{Z(\boldsymbol{\eta})}\exp(\boldsymbol{\eta}^T\mathbf{u}(\mathbf{x}))$.
- If there exists a value of $\boldsymbol{\eta}^*$ such that
$$E_{p(\mathbf{x})}[\mathbf{u}(\mathbf{x})] = E_{q(\mathbf{x}|\boldsymbol{\eta}^*)}[\mathbf{u}(\mathbf{x})], \tag{21}$$
then $q(\mathbf{x}|\boldsymbol{\eta}^*)$ is the M-projection.
- Proof: We have
$$\begin{aligned}\mathrm{KL}(p||q_{\boldsymbol{\eta}}) &= -H(p) - \int p(\mathbf{x})(-\ln Z(\boldsymbol{\eta}) + \boldsymbol{\eta}^T\mathbf{u}(\mathbf{x}))d\mathbf{x} \\ &= -H(p) + \ln Z(\boldsymbol{\eta}) - \boldsymbol{\eta}^T E_p[\mathbf{u}(\mathbf{x})].\end{aligned} \tag{22}$$
Hence,
$$\begin{aligned}&\mathrm{KL}(p||q_{\boldsymbol{\eta}}) - \mathrm{KL}(p||q_{\boldsymbol{\eta}^*}) \\ &= \ln\frac{Z(\boldsymbol{\eta})}{Z(\boldsymbol{\eta}^*)} - (\boldsymbol{\eta} - \boldsymbol{\eta}^*)^T E_p[\mathbf{u}(\mathbf{x})] \\ &= \ln\frac{Z(\boldsymbol{\eta})}{Z(\boldsymbol{\eta}^*)} - (\boldsymbol{\eta} - \boldsymbol{\eta}^*)^T E_{q_{\boldsymbol{\eta}^*}}[\mathbf{u}(\mathbf{x})] \\ &= E_{q_{\boldsymbol{\eta}^*}}\left[\ln\frac{q_{\boldsymbol{\eta}^*}(\mathbf{x})}{q_{\boldsymbol{\eta}}(\mathbf{x})}\right] \\ &= \mathrm{KL}(q_{\boldsymbol{\eta}^*}||q_{\boldsymbol{\eta}}) \geq 0.\end{aligned} \tag{23}$$

4 Appendix: Generalized Projections using α-Divergence

- The α-divergence is defined as [4]
$$D_\alpha(p||q) = \frac{\sum_x \alpha p(x) + (1-\alpha)q(x) - p(x)^\alpha q(x)^{1-\alpha}}{\alpha(1-\alpha)}, \tag{24}$$
where p and q need not be normalized.
- It can be proved that, as $\alpha \to 0$, we obtain $D_\alpha(p||q) = \mathrm{KL}(q||p)$, and, when $\alpha \to 1$, we have $D_\alpha(p||q) = \mathrm{KL}(p||q)$.
- Performing projections $\min_{q\in\mathcal{Q}} D_\alpha(p||q)$ with $\alpha \leq 0$ and decreasing values of α yields an increasingly mode-seeking, or exclusive, solution; while increasing values of $\alpha \geq 1$ yield increasingly inclusive and zero avoiding solutions. See Fig. 1 in [4] for an example.
- For all $\alpha \neq 0$, it can be proved that the stationary point of the projection $\min_{q\in\mathcal{Q}} D_\alpha(p||q)$ coincide with those of the projection $\min_{q\in\mathcal{Q}}\mathrm{KL}(p||p^\alpha q^{1-\alpha})$.