Technical University Berlin

Lecture Notes

Machine Learning I

read by Prof. Dr. Klaus-Robert Müller in the winter term 2019/2020

Contents

Lecture Notes by Viktor Glombik.

Last edited on October 8, 2020.

List of Figures

1 Bayes Decision Theory

14.10.2019

Probabilities and Measurements

- State of nature ω_i.
- prior probability $P(\omega_i)$
- measurements $x \in \mathbb{R}^d$
- likelihood function $p(x \mid \omega_j)$.

Bayes theorem

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Classification error

$$P(\text{error} \mid x) = \begin{cases} P(\omega_1 \mid x), & \text{if we decide } \omega_2, \\ P(\omega_2 \mid x), & \text{if we decide } \omega_1, \end{cases}$$

Optimal classifiers Bayes decision rule: decide ω_i, if $P(\omega_i \mid x) > P(\omega_j \mid x)\ \forall j \neq i$. Bayes error rate $R_B := \int_{\mathbb{R}^d} \min_i \big(1 - P(\omega_i \mid x)\big) \cdot p(x)\,\mathrm{d}x$.

Discriminant functions Bayes decision rule: decide ω_i if $g_i(x) > g_j(x)$ for all $j \neq i$, where $g_i(x) := f(P(\omega_i, \mid x))$ and f is a monotonically increasing function, i.e. decide ω_1 over ω_2 if $P(x \mid \omega_1)P(\omega_1) > P(x \mid \omega_2)P(\omega_2)$, which is equivalent to the one stated in "optimal classifiers".

This is somehow an generalization, which seems to be irrelevant?? **Actions and Cost** Let $(\alpha_k)_{k=1}^{\ell}$ be actions. Then $\lambda_{i,j} := \lambda(\alpha_i \mid \omega_j)$ describes the loss cause by taking action α_i given state of nature ω_j. The expected cost of action and the risk of the optimal decider are given by

$$R(\alpha_i \mid x) := \sum_{j=1}^{c} \lambda_{i,j} P(\omega_j \mid x) \quad \text{and} \quad R_\alpha := \int_{\mathbb{R}^d} \min_i R(\alpha_i \mid x) \cdot p(x)\,\mathrm{d}x,$$

respectively.

Example 1.1 Common and easy-to-handle likelihood functions include the multivariate normal and Bernoulli densities

$$\mathbb{R}^d \to \mathbb{R},\ x \mapsto \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{(x-\mu)^\top \Sigma^{-1}(x-\mu)}{2}\right) \sim \mathcal{N}_d(\mu, \Sigma)$$

$$\{0,1\}^d \to \mathbb{R},\ x \mapsto \prod_{i=1}^{d} \Pr[x_i = 0] \cdot \mathbb{1}_{x_i=0} + \Pr[x_i = 1] \cdot \mathbb{1}_{x_i=1} = \prod_{i=1}^{d} p_i^{x_i}(1-p_i)^{1-x_i}.$$

2 Maximum likelihood and Bayesian estimation

21.10.2019

What if $p(w_j)$, $P(x|w_j)$ or $p(w_j|x)$ are not known? Whilst $p(w_j)$ are relatively easy to estimate $p(w_j|x)$ is not. Maximum likelihood estimation assumes $p(x|w_j) \sim \mathcal{N}(\mu_j, \Sigma_j)$ for fixed but unknown parameters $\theta := (\mu_j, \Sigma_j)$ and estimates θ by maximising the probability of generating samples x that are given.

Let $(H_k)_{k=1}^n$ be examples, where the data in H_j is generated independently according to $p(x|w_j)$, e.g, $p(x|w_j, \theta) \sim \mathcal{N}(\mu_j, \Sigma_j)$. We can now write $H = (\tilde{x}_k)_{k=1}^n$, which we require to be i.i.d. Then

$$p(H|\theta) = \prod_{k=1}^n p(x_k|\theta)$$

WHY NOT X TILDE?? holds, where $p(H|\theta)$ can be seen as a function of θ and is called likelihood function with respect to generating the H. For convenience we can switch to maximising the log-likelihood

likelihood function

$$\ell(\theta) := \log(p(H|\theta)) = \sum_{k=1}^n \log(p(x_k|\theta)).$$

Differentiation yields

$$\nabla_\theta l(\theta) = \sum_{k=1}^n \nabla_\theta \log(p(x_k|\theta)).$$

Example 2.1 (Σ known, μ unknown) We have

$$\ell(\mu) = \sum_{k=1}^n \log(p(x_k|\mu)) = \sum_{k=1}^n -\frac{1}{2}\left(\log((2\pi)^d|\Sigma|) + (x_k - \mu)^\top \Sigma^{-1}(x_k - \mu)\right).$$

We want $\nabla_\mu \ell(\mu) = \sum_{k=1}^n \Sigma^{-1}(x_k - \mu) \overset{!}{=} 0$, implying $\sum_{k=1}^n (x_k - \hat{\mu}) = 0$ for the minimizer (estimator) $\hat{\mu}$ and therefore $\hat{\mu} = \frac{1}{n}\sum_{k=1}^n x_k$.

Example 2.2 (μ and σ^2 unknown) Let $\theta := (\mu, \sigma^2)$. We have

$$\ell(\theta) := \sum_{k=1}^n \log(p(x_k|\theta)) = \sum_{k=1}^n -\log(\sqrt{2\pi\theta_2}) - \frac{-1}{2\theta_2}(x_k - \theta_1)^2.$$

Differentiation yields

$$\nabla_\theta \ell(\theta) = \sum_{k=1}^n \left(\frac{x_k - \theta_1}{\theta_2}, \frac{(x_k - \theta_1)^2}{2\theta_2} - \frac{1}{\theta_2^2}\right)^\top.$$

Setting this equal to zero yields

$$\sum_{k=1}^n \frac{1}{\theta_2}(x - \hat{\theta}_1) = 0 \implies \hat{\theta}_1 = \hat{\mu} = \frac{1}{n}\sum_{k=1}^n x_k$$

and

$$\sum_{k=1}^n \hat{\theta}_2 = \sum_{k=1}^n (x_k - \hat{\theta}_1)^2 \implies \hat{\theta}_2 = \hat{\sigma}^2 = \frac{1}{n}\sum_{k=1}^n (x_k - \hat{\mu})^2.$$

We can see that the variance estimation is biased since it should have a $\frac{1}{n-1}$ factor instead of the $\frac{1}{n}$.

On the other hand, Bayes estimation proposes that in the light of data the prior can estimate the posterior:

$$p(w_i|H,x) = \frac{p(x|w_i,H)p(w_i|H)}{\text{normalisation}}.$$

We want to find $p(x|w_i,H)$, where $p(x)$ is fixed but unknown. Thus we want to estimate $p(x|H)$ and $p(x|\theta)$. The Bayesian approach suggests that θ is a random variable $p(\theta)$. Then we see data $p(\theta|H)$.

$$p(x|H) = \int p(x,\theta|H)\,\mathrm{d}\theta = \int p(x|\theta,H)p(\theta|H)\,\mathrm{d}\theta \overset{\text{ind.}}{=} \int p(x|\theta)p(\theta|H)\,\mathrm{d}\theta.$$

Now assume $p(\mu) \sim \mathcal{N}(\mu_0,\sigma_0^2)$. We obtain, assuming that $p(x_k|\mu)$ and $p(\mu)$ are Gaussians,

$$\begin{aligned}
p(\mu|H) &\propto p(H|\mu)p(\mu) \overset{\text{ind.}}{=} \prod_{k=1}^{n} p(x_k|\mu)\cdot p(\mu) \\
&= \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x_k-\mu)^2}{\sigma^2}\right) \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{1}{2}\frac{(\mu-\mu_0)^2}{\sigma_0^2}\right) \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{(\mu-\mu_0)^2}{\sigma_0^2} + \sum_{k=1}^{n}\frac{(x_k-\mu)^2}{\sigma^2}\right)\right) \\
&= \exp\left(\frac{-\sigma^2\mu^2 - 2\sigma^2\mu\mu_0 + \sigma^2\mu_0^2}{2\sigma^2\sigma_0^2} + \sum_{k=1}^{n}\frac{-\sigma_0^2x_k^2 + 2\sigma_0^2\mu x_k - \sigma_0^2\mu^2}{2\sigma^2\sigma_0^2}\right) \\
&= \exp\left(\frac{-\mu^2(\sigma^2+n\sigma_0^2) + 2\mu(\mu\sigma^2 + \sigma_0^2\sum_{k=1}^{n}x_i) - \left(\mu_0\sigma^2 + \sigma_0^2\sum_{k=1}^{n}x_k^2\right)}{2\sigma_0\sigma^2}\right) \\
&\propto \exp\left(\frac{-\mu^2 + 2\mu\left(\frac{\mu_0\sigma^2+\sigma_0^2\sum_{k=1}^{n}x_k}{\sigma^2+n\sigma_0^2}\right) - \left(\frac{\mu_0\sigma^2+\sigma_0^2\sum_{k=1}^{x_k}x_k}{\sigma^2+n\sigma_0^2}\right)^2}{\frac{2\sigma_0^2\sigma^2}{\sigma^2+n\sigma_0^2}}\right) \\
&\propto \exp\left(-\frac{\left(\mu - \frac{\mu_0\sigma^2+\sigma_0^2\sum_{k=1}^{n}x_k}{\sigma^2+n\sigma_0^2}\right)^2}{2\frac{\sigma_0^2\sigma^2}{\sigma^2+n\sigma_0^2}}\right) \sim \mathcal{N}(\mu_n,\sigma_n^2)
\end{aligned}$$

[2] where

$$\frac{1}{\sigma_n^2} := \frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}, \quad \frac{\mu_n}{\sigma_n^2} := \frac{n}{\sigma^2}\hat{\mu}_n + \frac{\mu_0}{\sigma_0^2}, \quad \text{where } \hat{\mu}_n := \frac{1}{n}\sum_{k=1}^{n} x_k.$$

In multiple dimensions this is similar: Consider data generated from the distribution $p(x \mid \mu) \sim \mathcal{N}(\mu,\Sigma)$, where the mean parameter has the prior distribution $p(\mu) \sim \mathcal{N}(\mu_0,\Sigma_0)$ and the covariance parameter Σ is known. We seek to estimate the parameter μ after observing some data set $\mathcal{D}$.

As above the MLE is given by

$$\hat{\mu} = \arg\max_{\mu} \log(p(\mathcal{D} \mid \mu)) = \frac{1}{n}\sum_{k=1}^{n} x_k.$$

The Bayesian estimator is given by

$$p(\mu \mid \mathcal{D}) \sim \mathcal{N}(\mu_n,\Sigma_n),\; p(x \mid \mathcal{D}) \sim \mathcal{N}(\mu_n,\Sigma+\Sigma_n),$$

where

$$\Sigma_n^{-1} = n\Sigma^{-1} + \Sigma_0^{-1} \quad \text{and} \quad \Sigma_n^{-1}\mu_n = n\Sigma^{-1}\hat{\mu} + \Sigma_0^{-1}\mu_0.$$

Lemma 2.3 (Convergence of Bayes Parameter Estimation)

① *The variance of the posterior is contained both by the uncertainty of the data mean and of the prior:* $\sigma_n^2 \leqslant \min\left(\frac{\sigma^2}{n}, \sigma_0^2\right)$.

② *The mean of the posterior distribution lies somewhere on the segment between the mean of the prior distribution and the sample mean:* $\min(\mu_0, \hat{\mu}_n) \leqslant \mu \leqslant \max(\mu_0, \hat{\mu}_n)$.

Proof. ① We have $a = \frac{1}{\frac{1}{b}+\frac{1}{c}}$ for $a = \sigma_n^2$, $b := \frac{\sigma^2}{n}$, $c := \sigma_0^2$. Without loss of generality let $b \geqslant c$. Then

$$\min\left(\frac{\sigma^2}{n}, \sigma_0^2\right) = \min(b, c) = c = \frac{1}{\frac{1}{c}} \geqslant \frac{1}{\frac{1}{b}+\frac{1}{c}}.$$

② We have

$$\sigma_n^2\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right) = 1 \tag{1}$$

from the first equality. Without loss of generality assume $\hat{\mu}_n \leqslant \mu_0$ $(\star)$. Thus

$$\begin{aligned}\hat{\mu}_n &\overset{(1)}{=} \hat{\mu}_n \sigma_n^2\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right) = \sigma_n^2\left(\frac{n}{\sigma^2}\cdot\hat{\mu}_n + \frac{\hat{\mu}_n}{\sigma_0^2}\right) \overset{(\star)}{\leqslant} \sigma_n^2\left(\frac{n}{\sigma^2}\cdot\hat{\mu}_n + \frac{\mu_0}{\sigma_0^2}\right)\\ &= \mu_n = \sigma_n^2\left(\frac{n\hat{\mu}_n}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}\right) \overset{(\star)}{\leqslant} \sigma_n^2\mu_0\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right) \overset{(1)}{=} \mu_0.\end{aligned}$$

□

3 Dimensionality Reduction and PCA

In many machine learning problems, the data such as images are high-dimensional. Let $(x_k)_{k=1}^N \subset \mathbb{R}^M$ be centered data points and $X := (x_1 \ \dots \ x_N)^T \in \mathbb{R}^{N\times M}$ be their associated matrix and $\Sigma := \frac{1}{N-1}X^\top X \in \mathbb{R}^{M\times M}$ the associated covariance matrix and μ_i are means of the two classes. Standard classification techniques are ill-defined for $M \gg N$ and ill-conditioned or numerically unstable even for $M < N$, where M is the dimension of feature space and N is number of data points. The usual solution is $w = \Sigma^{-1}(\mu_1 - \mu_2)$, but $\Sigma \in \mathbb{R}^{M\times M}$ is singular and ill-conditioned.

Since

$$\operatorname{cov}(X, X) = \mathbb{E}[X^\top X] - \mu_X \mu_X^\top,$$

a good approximation for centered data is $\frac{1}{N-1}X^\top X$.

We have to deal with the curse of dimensionality: When the dimensionality increases, the volume of the space increases so fast, that the available data becomes sparse. Furthermore, the amount of data needed for a reliable result often grows exponentially with the dimensionality.

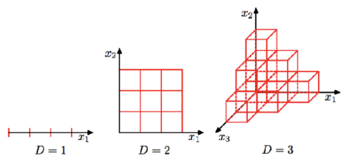


Figure 1: Curse of dimensionality. [1]

Regularisation

Therefore, we want to impose constraints on the parameters in order to stabilise solution. One way to do this is to introduce a prior probability.

In a linear regression task $y = X\beta + \varepsilon \in \mathbb{R}^n$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ the best linear unbiased estimator (the minimum variance unbiased estimator that is linear in y) is the LS-estimator $\hat\beta := (X^\top X)^{-1}X^\top y$ (Gauss-Markov-theorem).

Under the simple prior probability $\beta \sim \mathcal{N}(0, \sigma_\beta^2 I)$ and the maximum a-posteriori (MAP) approach

$$\hat\beta = \arg\max_\beta p(\beta|\,\mathcal{D})$$

we get

$$\hat\beta = \left(X^\top X + \frac{\sigma^2}{\sigma_\beta^2} I\right)^{-1} X^\top y,$$

as using Bayes rule (B) and that $p(\mathcal{D})$ doesn't depend on β it holds that

$$\begin{aligned} p(\beta|\,\mathcal{D}) &\overset{(B)}{=} p(\mathcal{D}\,|\beta)p(\beta) \propto \exp\left(-\frac{1}{2\sigma^2}\|y - X\beta\|^2\right)\cdot \exp\left(-\frac{1}{2\sigma_\beta^2}\|y\|^2\right) \\ &= \exp\left(-\frac{1}{2\sigma^2}\left(\|y - X\beta\|^2 + \frac{\sigma^2}{\sigma_\beta^2}\|y\|^2\right)\right) \end{aligned}$$

As $x \mapsto e^x$ is strictly monotonically increasing and continuous we have

$$\hat\beta = \arg\max_\beta p(\beta|\,\mathcal{D}) = \arg\max_\beta \|y - X\beta\|^2 + \frac{\sigma^2}{\sigma_\beta^2}\|\beta\|^2$$

Differentiating the last term with respect to β, setting $\tilde\sigma := \frac{\sigma^2}{\sigma_\beta^2}$ and equating to zero yields

$$\begin{aligned} -2X^\top(y - X\beta) + 2\tilde\sigma\beta \overset{!}{=} 0 &\implies X^\top(y - X\beta) = \tilde\sigma\beta \\ &\implies \beta = (X^\top X + \tilde\sigma I)^{-1}X^\top y. \end{aligned}$$

This is good, since $\Sigma := X^\top X + \tilde\sigma I$ is regular as it is symmetric and positive definite: For $v \neq 0$ and $\tilde\sigma > 0$ it holds that

$$v^\top(X^\top X + \tilde\sigma I)v = (Xv)^\top(Xv) + \tilde\sigma v^\top v = \|Xv\|^2 + \tilde\sigma\|v\|^2 > 0$$

Also, Σ is better conditioned: as it is symmetric, for $\tilde{\sigma} := \frac{\sigma^2}{\sigma_\beta^2}$ we have

$$\kappa(\Sigma) = \frac{\lambda_{\max}(\Sigma)}{\lambda_{\min}(\Sigma)} = \frac{\lambda_{\max}(X^\top X) + \tilde{\sigma}}{\lambda_{\min}(X^\top X) + \tilde{\sigma}}.$$

For example if $\lambda_{\max}(X^\top X) = 10^n = (\lambda_{\min}(X^\top X))^{-1}$, we have $\kappa(X^T X) = 10^{2n}$ but for $\tilde{\sigma} = 1$ we have $\kappa(\Sigma) = 10^n$

Dimensionality reduction

As we still have to invert a large matrix, which is very expensive, we want to to reduce the data to its most relevant features, which can be achieved by finding the relevant directions/subspaces in correlated data.

This makes visualisation of and thereby insights into the data easier, reduces the chance of overfitting (as data is lower dimensional), enables faster algorithms (algorithms suffer the curse of dimensionality) and lesser storage requirements.

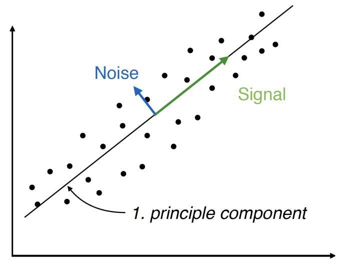


Figure 2: Which line fits data best? The line w that minimises the noise and maximises the signal [Pearson 1901].

Given centered data $(x_k)_{k=1}^n \subset \mathbb{R}^d$, we want to find a k-dimensional ($k < n$) subspace so that the projected data

1. is a close to the original data (minimum noise)
2. has maximum variance (maximum signal).

The orthogonal projection of a vector a onto a line b is given by $\frac{b^\top ab}{\|b\|^2}$. To minimise the distance (1) we calculate

$$\begin{aligned}\min_{\|w\|=1} \left\| \frac{w^\top x_i w}{\|w\|^2} - x_i \right\|^2 &= \min_{\|w\|=1} w^\top x_i x_i^\top w \underbrace{w^\top w}_{=1} - 2w^\top x_i x_i^\top w + x_i x_i^\top \\ &= \min_{\|w\|=1} -w^\top x_i x_i^\top w = \max_{\|w\|=1} w^\top x_i x_i^\top w.\end{aligned}$$

The restriction $\|w\| = 1$ expresses that we are only interested in the direction of w, not is length. Therefore

$$\min_{\|w\|=1} \sum_{i=1}^n \|w^\top x_i w - x_i\|^2 = \max_{\|w\|=1} w^\top X X^\top w.$$

To maximise variance (2) we calculate (recall that X is centered)

$$\max_{\|w\|=1} \operatorname{Var}(w^\top X) = \max_w \mathbb{E}[(w^\top X)^2] = \max_w \frac{1}{n} \sum_{i=1}^n w^\top x_i x_i^\top w = \max_w w^\top X X^\top w,$$

which is the same as above! How much of the signal are we going to loose?

$$\sum_{i=k+1}^d \operatorname{Var}(w^T x_i) = \frac{1}{n} \sum_{k+1}^d \lambda_i,$$

where λ_i are the eigenvalues of $XX^\top$.

Defining the scatter matrix $S := XX^\top$ leads to the constrained optimisation problem

scatter matrix

$$\max_w w^\top S w \qquad \text{subject to } \|w\| = 1$$

Defining Lagrangian $\mathcal{L}(w, \lambda) := w^\top S w + \lambda(1 - w^\top w)$, differentiating and setting to zero we obtain

$$\frac{\partial L(w, \lambda)}{\partial w} = 2Sw - 2\lambda w \overset{!}{=} 0 \implies Sw = \lambda w,$$

which is an eigenvalue problem.

The k-th eigenvalue represents the variance in the direction of the k-th eigenvector:

$$\operatorname{Var}(w_k^\top x) = \frac{1}{n} w_k^\top S w_k = \frac{1}{n}\lambda_k w_k^\top w_k = \frac{\lambda_k}{n},$$

as w_k is an eigenvector of S with unit norm. Furthermore, the k-th principal component is given by the k-th column of XS, where $\frac{1}{n-1}X^TX = SDS^T$ is the diagonalization of the covariance matrix. Simply put we have: "The direction of largest variance corresponds to the largest eigenvector, i.e. the eigenvector with largest eigenvalue."

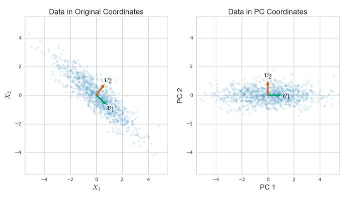


Figure 3: PCA rotates data into a new coordinate system with the directions of the largest variance being the new coordinate axes.

Instead of calculating the eigendecomposition of $S = XX^T$ we compute the SVD of X, as it is computationally more stable: for $\varepsilon > 0$ consider the LÄUCHLI matrix $X := \left(\begin{smallmatrix} 1 & \varepsilon & 0 & 0 \\ 1 & 0 & \varepsilon & 0 \\ 1 & 0 & 0 & \varepsilon \end{smallmatrix}\right)$. Then $XX^T \approx \left(\begin{smallmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{smallmatrix}\right)$, which is singular, whereas the singular values are tiny but non-zero.

The SVD has computational complexity $\mathcal{O}(\min(n^2d, d^2n))$. As we only need a few largest principle components and not all of them, we can perform power iteration: Start with a random normed vector x_0. Then iterate $x_{k+1} := \frac{Ax_k}{\|Ax_k\|}$. The solution is the largest eigenvector of A.

power iteration

Proof. Assume A is orthogonally diagonisable with $A = U\Lambda U^{-1}$ (we write $U := (u_1 \ \dots \ u_d)$) and let d be the geometric and algebraic multiplicity of λ_1. Then we have $\lambda_1 = \ldots = \lambda_d$ and assume further that $|\lambda_d| > |\lambda_{d+1}| \geqslant \ldots \geqslant |\lambda_n|$. Let $x = \sum_{i=1}^n \beta_i u_i$ with $\sum_{i=1}^d \beta_i u_i \neq 0$. Then we have $u_i = Ue_i$ and thus

The proof is analogous in the non-diagonisable setting, where one uses the Jordan canonical form.

$$\begin{aligned} A^k x &= U\Lambda^k \left(\sum_{i=1}^n \beta_i e_i\right) = U\left(\lambda_1^k \sum_{i=1}^d \beta_i e_i + \sum_{i=d+1}^n \lambda_i^k \beta_i e_i\right) \\ &= \lambda_1^k \left(\sum_{i=1}^d \beta_i u_i + \underbrace{\sum_{i=d+1}^n \left(\frac{\lambda_i}{\lambda_1}\right)^k \beta_i u_i}_{\xrightarrow{k\to\infty} 0 \text{ as } \frac{\lambda_i}{\lambda_1} \leqslant 1}\right). \end{aligned}$$

We have $\lim_{k\to\infty} \lambda_1^k = 0$ if $|\lambda_1| < 1$ and ∞ else. Because of the condition on x, x has a non-zero share in λ_1-direction, i.e. $\beta_1 \neq 0$. Therefore, $\frac{A^k x}{\|A^k x\|} \xrightarrow{k\to\infty} u_1$.□

Remark 3.1 The convergence is geometric, with ratio $\left|\frac{\lambda_2}{\lambda_1}\right|$. Thus, the method converges slowly if there is an eigenvalue close in magnitude to the dominant eigenvalue.

To find the second eigenvector we use that $S = \sum_{i=1}^d \lambda_i w_i w_i^\top$, so we can use power iteration on $S - \lambda_1 w_1 w_1^\top$.

Applications

We have seen dimensional reduction as an application of PCA. There are more:

Eigenfaces. We use the idea that faces look very similar compared to random images. The principle components are directions in our faces space. Thus each principle component is a face representation, too.

Denoising. First reduce the dimension to filter out "noise" directions $(w_i^\top x)_{i=1}^k$. Then project back into the original space via $\sum_{i=1}^k w_i^\top x w_i + \sum_{i=1}^n x_i$ and undo the centering.

EEG. In electroencephalographic recordings, eye blink artifacts can be stronger than the neuronal activity. Therefore it is reasonable to remove the first few principal components.

How robust is PCA to outliers? Not very robust.

4

bla

5 Clustering and Expectation Maximisation

5.1 K-means clustering

We partition ($X = \bigcup_{k=1}^K S_k$) the unlabelled data points $X := \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ in to K disjoint sets S_k based on similarity. We can define the clusters S_k by the minimum euclidean distance to the cluster mean. We want to find $\theta = \{S_1, \ldots, S_k\}$ which minimises the objective

$$J(\theta) = \sum_{k=1}^{K} \sum_{x_n \in S_k} \|x_n - \mu_k\|, \quad \text{where } \mu_k := \frac{1}{|S_k|}$$

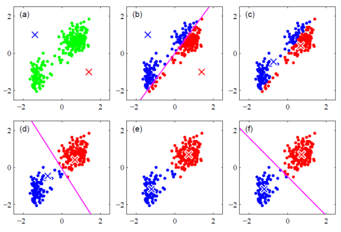


Figure 4: TODO

Algorithm 1: Expectation Maximisation

```
1 Choose $K$ random points as initial cluster centres $\mu_1^{(0)}, \ldots, \mu_K^{(0)}$.
2 **Assignment (E):**
    $S_k^{(t)} := \Big\{x_n : \|x_n - \mu_k^{(t)}\|^2 < \|x_n - \mu_\ell^{(t)}\|^2 \ \forall \ell \in \{1, \ldots, K\}\Big\}$.
3 **Update (M):** $\mu_k^{(t+1)} := \frac{1}{|S_k^{(t)}|} \sum_{x_n \in S_k^{(t)}} x_n$.
4 Iterate 2. and 3. until convergence to a local minimum.
```

Density Estimation with GMMs

A multivariate Gaussian density does not always model data (e.g. class-conditional densities) well. A solution is the Gaussian mixture model

$$p(x|\theta) := \sum_{k=1}^{K} \tau_k p_k(x|\mu_k, \Sigma_k), \quad \text{where } p_k(x|\mu_k, \Sigma_k) \sim \mathcal{N}(\mu_k, \Sigma_k),$$

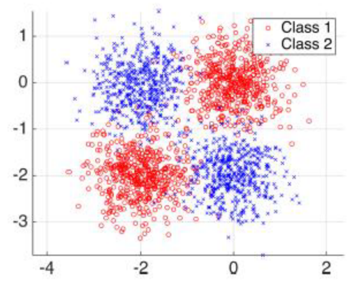


Figure 5: An example where a Gaussian model will not preform well.

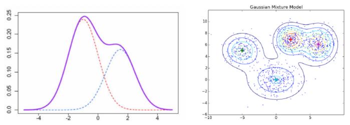


Figure 6: GMM's are universal density approximations

where τ_k are scaling factors or "priors" of $p_k(x|\mu_k, \Sigma_k)$ with $\sum_{k=1}^K \tau_k = 1$.

With the parameter vector $\theta := \{\tau_1, \ldots, \tau_k, \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K\}$, we can fit a GMM using maximum likelihood: the log-likelihood is

$$\begin{aligned} L(\theta) = \log(p(X|\theta)) &= \log\left(\prod_{n=1}^{N} \sum_{k=1}^{K} \tau_k p_k(x_n|\mu_k, \Sigma_k)\right) \\ &= \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \tau_k p_k(x_n|\mu_k, \Sigma_k)\right) \\ &= \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \underbrace{\tau_k \frac{(2\pi)^{-\frac{d}{2}}}{|\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x_n - \mu_k)^\top \Sigma_k^{-1}(x_n - \mu_k)\right)}_{=:f(\theta_k)}\right), \end{aligned}$$

which is difficult to optimise because of the red factor **(WHYYY)**, which is due to the logarithm: $\frac{\partial L(\theta)}{\partial \theta_i} = \sum_{n=1}^{N} \frac{1}{\sum_k f(\theta_k)} \frac{\partial f(\theta_i)}{\theta_i}$ and thus admits no analytic solution **(WHYYY)**.

We can instead fit a GMM using EM by introducing auxiliary variables indicating the membership of each sample to a Gaussian: $z_1, \ldots, z_N \in \{0, 1\}^K \sim \text{Categorial}(\tau)$ with $p(z_{nk} = 1) = \tau_k$ and for each n there exists exactly one k

This is also how to sample from a GMM: 1 sample $z \sim \text{Categorial}(\tau)$, 2 sample $x \sim \mathcal{N}(\mu_k, \Sigma_k)$, where $z_k = 1$.

such that $z_{nk} = 1$ and $z_{nj} = 0$ for $j \neq k$.

Algorithm 2: Fitting a GMM using EM.

```
1 Initialise $t = 0$, $\theta^{(0)} = \{\tau_1, \dots, \tau_K, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_k\}$
2 **Expectation:** compute membership probabilities given $\theta^{(t)}$
      $$q^{(t)}(z_{nk}) := p(z_{nk} = 1 | x_n, \theta^{(t)}) \overset{(B)}{=} \frac{p(x_n | z_{nk}, \theta^{(t)}) p(z_{nk}, \theta^{(t)})}{p(x_n | \theta^{(t)})} = \frac{\tau_k^{(t)} p_k(x_n | \mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_{\ell=1}^K \tau_\ell^{(t)} p_\ell(x_n | \mu_\ell^{(t)}, \Sigma_\ell^{(t)})}.$$
   **WHY ISN'T IT $z_{nk} = 1$ IN THE SECOND TERM??**
3 **Maximisation:** update $\theta$ given (soft) cluster assignments:
      $$\tau_k^{(t+1)} := \frac{1}{N} \sum_{n=1}^N q^{(t)}(z_{nk}), \quad \mu_k^{(t+1)} := \frac{1}{N\tau_k^{(t+1)}} \sum_{n=1}^N q^{(t)}(z_{nk}) x_n,$$
      $$\Sigma_k^{(t+1)} = \frac{1}{N\tau_k^{(t+1)}} \sum_{n=1}^N q^{(t)}(z_{nk})(x_n - \mu_k^{(t+1)}(x_n - \mu_k^{(t+1)})^\top.$$
```

e.g. $\tau_k^{(0)} = \frac{1}{K}$, $\Sigma_k^{(0)} = I$, $\mu_k^{(0)}$ random.

It is also possible to use hard cluster assignments, where one instead uses

$$q^{(t)}(z_{nk}) := p(z_{nk} = 1 | x_n, \theta^{(t)}), \quad z_{nk}^{(t)} = \mathbb{1}\left(q^{(t)}(z_{nk}) = \max_\ell q^{(t)}(z_{n\ell})\right).$$

$$\tau_k^{(t+1)} := \frac{1}{N} \sum_{n=1}^N z_{nk}, \quad \mu_k^{(t+1)} := \frac{1}{N\tau_k^{(t+1)}} \sum_{n=1}^N z_{nk} x_n,$$

$$\Sigma_k^{(t+1)} = \frac{1}{N\tau_k^{(t+1)}} \sum_{n=1}^N z_{nk} \left(x_n - \mu_k^{(t+1)}\right) \left(x_n - \mu_k^{(t+1)}\right)^\top.$$

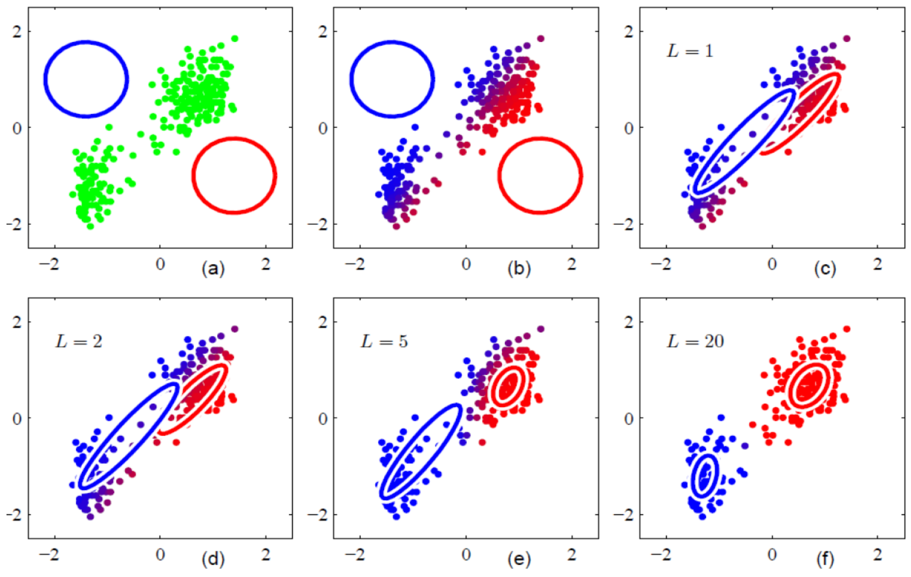


Figure 7: Fitting a GMM using EM. [Bishop 2006, Figure 9.8]

In contrast to K-means, GMM also for

- unequal cluster variances and cluster probabilities
- non-spherical clusters

- soft cluster assignment

Im maximum likelihood, our goal is to optimise $p(X|\theta)$. Do the update equations optimise $p(X|\theta)$? Let's look at the EM algorithm in general.

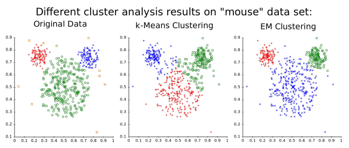


Figure 8: Fitting a GMM: comparison using EM [Wikipedia]

Given unobserved (latent) variables z, the observed data X and the model parameters θ, we want to maximise the likelihood of the observed data (= incomplete-data likelihood) $L(\theta|X) := p(X|\theta)$ and find

$$\hat{\theta} = \arg\max_{\theta} \log(p(X|\theta)) = \arg\max_{\theta} \log\left(\sum_{z\in Z} p(X,z|\theta)\right).$$

Minimising this directly is difficult because of the $\log\left(\sum\ldots\right)$ structure. On the other hand, it is often easy to optimise the complete-data likelihood $L(\theta|X,z) = \log(p(X,z|\theta))$

Example 5.2 (GMM)
In GMM, the incomplete-data log-likelihood is

$$\log(p(X|\theta)) := \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \tau_k p_k(x_n|\mu_k,\Sigma_k)\right),$$

while the complete-data log-likelihood is

$$\log(p(X,z|\theta)) := \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \delta_{z_{nk}=1}\tau_k p_k(x_n|\mu_k,\Sigma_k)\right),$$

$\rightarrow$ Analytic maximum likelihood estimate for each $\theta_k = (\tau_k,\mu_k,\Sigma_k)$.

But we don't know z, so we have to estimate it jointly with θ. The expectation maximisation algorithm iterates between updates of the hidden variables and parameters. In theory, the updates are defined such that $p(X|\theta)$ increases in each step, so we always find a local maximum of $p(X|\theta)$ (can be hard to find global maximum if $p(X|\theta)$ is non-concave). Technically, we optimise a lower bound on $p(X|\theta)$ and subsequently improve that bound.

Let $q(z)$ be a probability mass function (and thus $\sum_z q(z) = 1$) of choice of z. By JENSEN's inequality, applicable due to the concavity of the logarithm, we have

$$\log(p(X|\theta)) = \log\left(\sum_z p(X,z|\theta)\right) = \log\left(\sum_z q(z)\frac{p(X,z|\theta)}{q(z)}\right)$$
$$\overset{(J)}{\geqslant} \sum_z q(z)\log\left(\frac{p(X,z|\theta)}{q(z)}\right) =: F(q(z),\theta).$$

This bound (with structure $\sum_z \log(\ldots)$) is much easier to optimise than the $\log(\sum\ldots)$ structure.

Instead of the true objective (maximising the data log likelihood with respect to θ) $\hat{\theta} = \arg\max_\theta \log(p(X|\theta))$, which is difficult, we consider the EM objective (maximise, with respect to q and θ, the lower bound)

$$\hat{q},\hat{\theta} = \arg\max_{q,\theta} F(q(z),\theta).$$

The are two way to improve the lower bound

① Expectation: improve $q(z)$ for given θ.

② Maximisation: improve θ for given $q(z)$.

Let us first tackle step ① The difference between the data log-likelihood and the lower bound is

$$\begin{aligned}\log(p(X|\theta)) - \sum_z q(z)\log\left(\frac{p(X,z|\theta)}{q(z)}\right) &\overset{(B)}{=} \log(p(X|\theta)) - \sum_z q(z)\log\left(\frac{p(X|\theta)p(z|X,\theta)}{q(z)}\right) \\ &= \underbrace{\log(p(X|\theta)) - \sum_z q(z)\log\left(p(X|\theta)\right)}_{=0 \quad \text{as } \sum_z q(z)=1} - \sum_z q(z)\log\left(\frac{p(z|X,\theta)}{q(z)}\right) \\ &= \sum_z q(z)\log\left(\frac{q(z)}{p(z|X,\theta)}\right) = \mathrm{KL}(q(z)\|p(z|X,\theta)),\end{aligned}$$

where

$$KL(P\|Q) = \sum_x P(x)\log\left(\frac{Q(x)}{P(x)}\right) = \mathbb{E}_x\left[\log\left(\frac{Q(x)}{P(x)}\right)\right] \geqslant 0$$

is the KULLBACK-LEIBLER divergence, which is a measure of the distance between two distributions P and Q but not a metric (not symmetric, triangle inequality not fulfilled).

We have $\mathrm{KL}(q(z)\|p(z|X,\theta)) = 0$ if and only if $q(z) = p(z|X,\theta)$, so the lower bound is strict if $q(z) = p(z|X,\theta)$ **(???)**. In the expectation step we thus set $q^{(t)}(z) = p(z|X,\theta^{(t)})$.

Now let us turn to step ②, where we maximise $F(q(z),\theta)$ with respect to θ. We have

$$\begin{aligned}\theta^* &= \arg\max_\theta \sum_z q(z)\log\left(\frac{p(X,z|\theta)}{q(z)}\right) \\ &= \arg\max_\theta \sum_z q(z)\log\left(p(X,z|\theta)\right) - \underbrace{\sum_z q(z)\log(q(z))}_{=:H(q)\ (\text{entropy})} \\ &= \arg\max_\theta \sum_z q(z)\log\left(p(X,z|\theta)\right).\end{aligned}$$

To find the maximum, we set the gradient to zero. Typically, there is a analytic solution, due the the favourable $\sum\log$ structure. In the maximisation step, we thus set $\theta^{(t+1)} = \arg\max_\theta F(q^{(t)}(z),\theta)$.

We thus have

$$\begin{aligned}\log(p(X|\theta^{(t)}) &\overset{\text{E-step}}{=} F\left((q^{(t+1)}(z),\theta^{(t)}\right) \\ &\overset{\text{M-step}}{\leqslant} F\left((q^{(t+1)}(z),\theta^{(t+1)}\right) \overset{(J)}{\leqslant} \log(p(X|\theta^{(t+1)}),\end{aligned}$$

so $L(\theta|X) = \log(p(X|\theta))$ converges to a local maximum.

Note that update of $q(z) = p(z|X,\theta) \to$ update of z **(???)**. We summarise our findings in the following algorithm.

```
Data: latent $z$, observed data $X$, model parameters $\theta$.
1 Initialise $\theta^{(0)}$ randomly.;
2 Expectation: $q^{(t)}(z) = p(z|X,\theta^{(t)})$.;
3 Maximisation: $\theta^{(t+1)} = \arg\max_\theta \sum_z q^{(t)}(z)\log(p(X,z|\theta))$.;
4 Iterate until convergence.
```

Why is the first step called "expectation"? Remember that the maximisation step is

$$\begin{aligned}\theta^{(t+1)} &= \arg\max_\theta F(q^{(t)}(z),\theta) = \arg\max_\theta \sum_z q^{(t)}(z)\log(X,z|\theta) + H(q)\\ &= \arg\max_\theta \sum_z q^{(t)}(z)\log(X,z|\theta)\\ &= \arg\max_\theta \mathbb{E}_{z|X,\theta^{(t)}}\left[\log(X,z|\theta)\right] =: \arg\max_\theta Q(\theta|\theta^{(t)}).\end{aligned}$$

The original "expectation step" (Dempster et al., 1977) is to compute

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{z|X,\theta^{(t)}}\left[\log(p(X,z|\theta))\right],$$

which boils down estimating $q^{(t)}(z) = p(z|X,\theta^{(t)})$.

For a fitting a GMM using EM, the expectation step

$$q^{(t)}(z_{nk}) := p(z_{nk}=1|x_n,\theta^{(t)}) = \frac{\tau_k^{(t)} p_k(x_n|\mu_k^{(t)},\Sigma_k^{(t)})}{\sum_{\ell=1}^K \tau_\ell^{(t)} p_\ell(x_n|\mu_\ell^{(t)},\Sigma_\ell^{(t)})}.$$

We have

$$\begin{aligned}Q(\theta,\theta^{(t)}) &= \sum_z q^{(t)}(Z)\log(p(X,z|\theta)) = \mathbb{E}_{Z|X,\theta^{(t)}}\left[\log(p(X,Z|\theta))\right]\\ &= \mathbb{E}_{Z|X,\theta^{(t)}}\left[\log\left(\prod_{n=1}^N p(x_n,z_n|\theta)\right)\right] = \sum_{n=1}^N \mathbb{E}_{Z|X,\theta^{(t)}}\left[\log(p(x_n,z_n|\theta))\right]\\ &= \sum_{n=1}^N\sum_{k=1}^K p(z_{nk}=1|x_n,\theta^{(t)})\log\left(\tau_k p_k(x_n|\mu_k\Sigma_k)\right)\\ &= \sum_{n=1}^N\sum_{k=1}^K q^{(t)}(z_{nk})\left(\log(\tau_k) - \frac{1}{2}\log(|\Sigma_k|) - \frac{1}{2}(x_n-\mu_k)^\top\Sigma_k^{-1}(x_n-\mu_k) - \frac{d}{2}\log(2\pi)\right).\end{aligned}$$

We get

$$\tau^{(t+1)} = \arg\max_\tau Q(\theta,\theta^{(t)} \quad \text{subject to} \quad \sum_{k=1}^K \tau_k = 1.$$

The Lagrangian is

$$L(\tau,\lambda) := \sum_{k=1}^K\sum_{k=1}^K \log(\tau_k)\sum_{n=1}^N q^{(t)}(z_{nk}) + \lambda\left(1-\sum_{k=1}^K\tau_k\right)$$

and its derivative with respect to $\tau_k^{(t+1)}$

$$\frac{1}{\tau_k^{(t+1)}}\sum_{n=1}^N q^{(t)}(z_{nk}) - \lambda \overset{!}{=} 0,$$

yielding $\tau_k^{(t+1)} = \frac{1}{\lambda}\sum_{n=1}^N q^{(t)}(z_{nk})$. From $\sum_{k=1}^K \tau_k^{(t+1)} = 1$ we get $\lambda = N$ **(as $\sum_{k=1}^K\sum_{n=1}^N q^{(t)}(z_{nk}) = N$??!)** and thus

$$\tau_k^{(t+1)} = \frac{1}{N}\sum_{n=1}^N q^{(t)}(z_{nk}).$$

Furthermore,

$$\begin{aligned}(\mu_k^{(t+1)},\Sigma_k^{(t+1)}) &= \arg\max_{\mu_k,\Sigma_k} Q(\theta|\theta^{(t)})\\ &= \arg\max_{\mu_k,\Sigma_k} -\frac{1}{2}\sum_{n=1}^N q^{(t)}(z_{nk})\Big(\log(|\Sigma_k|) + (x_n-\mu_k)^\top\Sigma_k^{-1}(x_n-\mu_k)\Big).\end{aligned}$$

Setting the derivative to zero yields

$$\mu_k^{(t+1)} = \frac{1}{N\tau_k^{(t+1}} \sum_{n=1}^{N} q(t)(z_{nk}) x_n,$$

$$\Sigma_k^{(t+1)} = \frac{1}{N\tau_k^{(t+1)}} \sum_{n=1}^{N} q(t)(z_{nk}) \left(x_n - \mu_k^{(t+1)}\right) \left(x_n - \mu_k^{(t+1)}\right)^\top,$$

which is just a weighted version of the maximum likelihood estimate for a single GAUSSIAN.

In summary, EM

- is a "meta-algorithm" for obtaining local maximum likelihood estimates
- is also applicable to maximum a-posteriori (MAP) estimation
- is particularly useful in models with latent variables z, where optimising the incomplete likelihood directly is hard, but optimising the complete-data likelihood $p(X, z|\theta)$ is easy. One alternates between estimating z and θ.
- can be applied to a GMM, but EM is not equal to a GMM.

Other applications of EM are Hidden MARKOV models, missing / incomplete data, or if only summary data is observed.

Pro	Con
no step size / learning rate	"only" local minima found
each iteration improves likelihood	solution dependent on initialisation
	can be slow

Sometimes it is possible to use generic solvers (e.g. NEWTON). But: complicated gradients and update rules and no improvement guarantees (e.g. JENSEN requires densities) **(???!)**.

6 Fisher Discriminant & Classification

Recall that if $p(x|\omega_1) \sim \mathcal{N}(\mu_1, \Sigma)$ and $p(x|\omega_1) \sim \mathcal{N}(\mu_2, \Sigma)$, the BAYES decision rule from section one becomes a linear function of x: we decide ω_1 if

$$x^\top \Sigma \mu_1 - \frac{1}{2}\|\mu_1\|^2 + \log(P(\omega_1)) \geqslant x^\top \Sigma \mu_2 - \frac{1}{2}\|\mu_2\|^2 + \log(P(\omega_2))$$

Assume now that for each class ω_i we only know the mean $\boldsymbol{\mu}_i$ and the covariance Σ_i of our data generating distribution. The goal of the Fisher Discriminant is to find a linear projection $y = w^\top x$ of the data such that separability as measured by the ratio of between- and within-class variability is maximised.

Fisher Discriminant

For each class, we first express the mean in projected space:

$$\mu_i = \int w^\top x \, \mathrm{d}p(x|\omega_i) = w^\top \boldsymbol{\mu}_i.$$

The within-class variability is

$$\sigma_i^2 = \int (w^\top x - \mu_i)^2 \, \mathrm{d}p(x|\omega_i) = w^\top \Sigma_i w.$$

The between-class variability is the squared difference of the class means:

$$\sigma_B^2 := (\mu_1 - \mu_2)^2 = w^\top S w, \quad \text{where } S_B := (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top.$$

We thus maximise

$$J(w) = \frac{\sigma_B^2}{\sigma_1^2 + \sigma_2^2} = \frac{w^\top S_B w}{w^\top S_W w}, \quad \text{where } S_W := \Sigma_1 + \Sigma_2,$$

which leads to the generalised eigenvalue problem

$$S_B w = \lambda S_W w.$$

As $S_B w$ and $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ are collinear (they differ by a scalar factor of $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top w$), the solution is

$$w = S_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2).$$

It is equivalent to an optimal classifier between Gaussian densities of same covariances (cf. above).

Unlike PCA, Fisher vectors are not sensitive to large variance components. This can be useful to build classifiers that are robust to large variations of contrast.

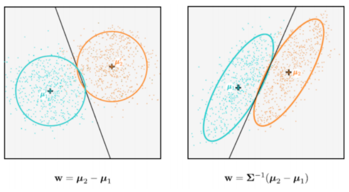


Figure 9: Fisher faces adapts to the covariance structure of the data and consequently achieves better separation.

6.1 Direct error minimisation

We observe that for other types of distributions, e.g. non-Gaussian, computing the Fisher discriminant may result in a suboptimal classifier. Instead, we can consider the overall error of the classifier (minimise it by gradient descent)

$$\int \sum_{i=1}^{c} P(\omega_i|x)\,\mathbb{1}(o(x) \neq \omega_i)p(x)\,\mathrm{d}x,$$

where $o(x)$ is the decision function.

Remark 6.2 (Finite sample approximation)
In practice, we don't know the probability distribution $p(x)$ and only have access to a limited data set $\mathcal{D} := ((x_n, t_n))_{n=1}^{N}$ with $t_n \in \{\omega_1, \ldots, \omega_c\}$. Replacing $p(x)$ and $p(\omega_i|x)$ by the empirical distribution formed by our data set yields the simplified objective

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{c} \mathbb{1}(o(x) \neq \omega_i)\,\mathbb{1}(t_n = \omega_i)$$

Remark 6.3 (Differentiable error functions)
A further difficulty with the original error function is that the decision function $o(x)$ and the way it appears in the error function is not differentiable. Therefore, we need to define a surrogate objective that is differentiable and that approximates our true objective. One possibility is to use the mean square error

$$E_D(f) = \sum_{n=1}^{N} \sum_{i=1}^{c} \left(\mathbb{1}(t_n = \omega_i) - f_i(x_n)\right)^2,$$

where $f\colon \mathbb{R}^d \to \mathbb{R}^c$ is the prediction function, for example, the collection of linear models $f_i(x) = w_i^\top x + b_i$. The objective $E_D(f)$ is differentiable and can be optimised by gradient descent, i.e. we compute $w \leftarrow w - \gamma \nabla_w E_D(f)$ and similarly for b. After learning f, we can build the decision function: decide ω_i if $f_i(x) > f_j(x)$ for all $j \neq i$.

One could also use the cross-entropy as a surrogate:

$$E_D(f) = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{c} \mathbb{1}(t_n = \omega_i)\left(-\log(f_i(x_n))\right),$$

where g_i is the modelled probability for class i, and subject to the constraints $g_i \geqslant 0$ and $\sum_{j=1}^{c} g_j(x) = 1$. One class of functions that outputs a probability distribution subject to these constraints is the generalised linear model

$$g_i(x) = \frac{\exp(w_i^\top x + b_i)}{\sum_{j=1}^{c} \exp(w_j^\top x + b_j)}.$$

Later this semester: regularisation, large-margin classifiers (hinge loss) and nonlinear classifiers (decision trees, kernel machines, neural networks).

7 Model selection

Given data, a linear model might seem appropriate, even though the data is quadratic (cf. right). If we take a polynomial model with order equal to the number of data points, overfitting occurs. To balance those out, we turn to Occam's razor, the law of parsimony: the simplest model/most intuitive method is preferable. But what does "simple" or "intuitive" mean? One approach is that a good model "must accurately describe a large class of observations on the basis of a model that contains only a few arbitrary elements, and it must make definite predictions about the results of future observations" (Stephen Hawking). We thus seek a model with the lowest prediction error.

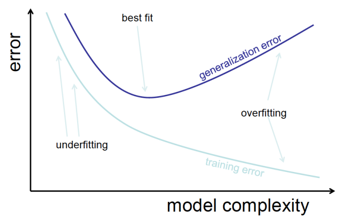


Figure 10: Occam's razor (semi-quantitative version).

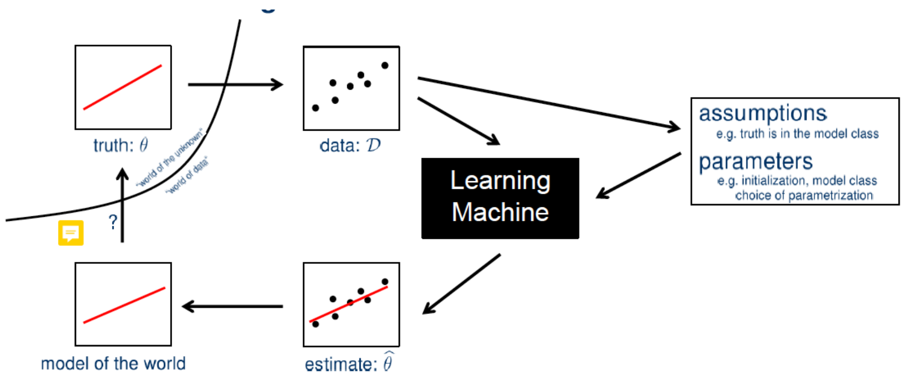


Figure 11: Learning machine = function $\mathcal{D} \mapsto \hat{\theta}$

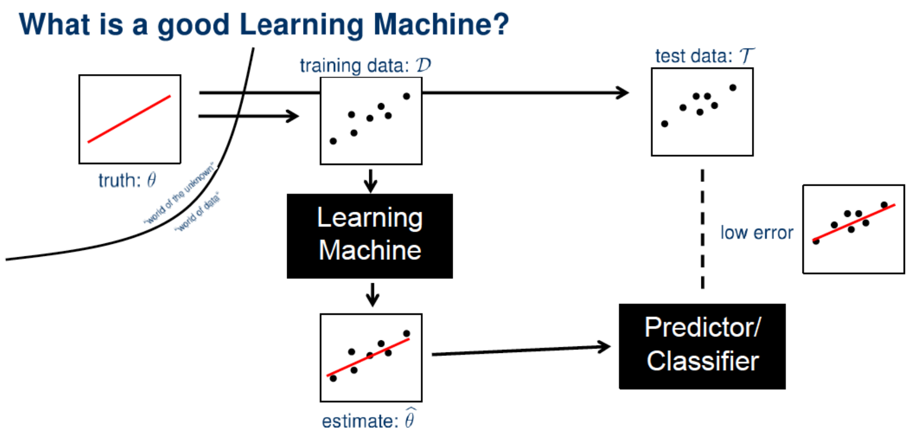


Figure 12: Caption

7.1 Bias and variance

In parametric estimation, $\theta \in \mathcal{C}^n$ is a value (e.g. $\theta = (\mu, \Sigma)$ for Gaussians) and $\hat{\theta}$ is a function in the data $\mathcal{D} = \{X_1, \ldots, X_n\}$, where X_i are random variables giving back data points. The bias of $\hat{\theta}$, $\mathbb{E}[\hat{\theta} - \theta]$, measures the expected deviation of the mean. The variance of $\hat{\theta}$, $\mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2]$, measures the scatter around the estimator mean. The MSE of $\hat{\theta}$, $\mathbb{E}[(\hat{\theta} - \theta)^2]$, measures the prediction error.

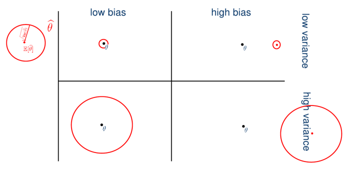


Figure 13: Bias and variance.

Lemma 7.2 (Bias-variance-tradeoff / decomposition) *We have MSE*$(\hat{\theta})$ = *Bias*$(\hat{\theta})^2 +$ *Var*$(\hat{\theta})$.

Proof. We have

$$\begin{aligned}
\mathbb{E}[(\hat{\theta} - \theta)^2] &= \mathbb{E}[\hat{\theta}^2] - 2\theta\,\mathbb{E}[\hat{\theta}] + \theta^2 \\
&= \mathbb{E}[\hat{\theta}^2] - 2\,\mathbb{E}[\hat{\theta}]^2 + \mathbb{E}[\hat{\theta}]^2 + \mathbb{E}[\hat{\theta}]^2 - 2\theta\,\mathbb{E}[\hat{\theta}] + \theta^2 \\
&= \mathbb{E}[\hat{\theta}^2] - 2\,\mathbb{E}[\hat{\theta} \cdot \mathbb{E}[\hat{\theta}]] + \mathbb{E}[\hat{\theta}]^2 + (\mathbb{E}[\hat{\theta}] - \theta)^2 \\
&= \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2] + (\mathbb{E}[\hat{\theta} - \theta])^2 \qquad \square
\end{aligned}$$

Example 7.3 (Estimation of the mean) Let $X_1, \ldots, X_N$ be i.i.d. $\sim \mathcal{N}(\mu, \sigma)$. The "natural" estimator is $\hat{\mu} := \frac{1}{N}\sum_{i=1}^N X_i$. We have

$$\text{Bias}(\hat{\mu}) = \mathbb{E}[\hat{\mu} - \mu] = \frac{1}{N}\sum_{i=1}^N \mathbb{E}[X_i] - \mu = 0$$

and MSE($(\hat{\mu})$ =Var$(\hat{\mu}) = \frac{\sigma^2}{N}$, as

$$\text{Var}(\hat{\mu}) = \mathbb{E}[(\hat{\mu} - \mu)^2] = \mathbb{E}[(\hat{\mu} - \mathbb{E}[\hat{\mu}])^2] = \frac{1}{N^2} N\sigma^2 = \frac{\sigma^2}{N},$$

as the X_i are independent.

If instead $X_1, \ldots, X_N \in \mathbb{R}^n$ for $n > 2$ are i.i.d. with $X_i \sim \mathcal{N}(\mu, \sigma \cdot I$, the natural estimator from above has the same variance, MSe and bias, but the (biased) James-Stein-estimator

$$\hat{\mu}_{\text{JS}} := \hat{\mu} - \frac{(n-2)\sigma^2}{\hat{\mu}^\top \hat{\mu}}\hat{\mu}$$

has Bias$(\hat{\mu}_{\text{JS}}) > 0$ and MSE$(\hat{\mu}_{\text{JS}})$ <MSE$(\hat{\mu})$.

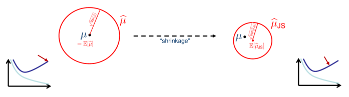


Figure 14: The James-Stein paradox

In supervised learning, the training data $\mathcal{D}$ is comprised of $X_1, \ldots, X_N$ with labels $Y_1, \ldots, Y_N$. The parameter θ "is" a generative function $f = f_\theta$: $Y_i = f(X_i) + \varepsilon_i$, where ε_i is an error with $\mathbb{E}[\varepsilon_i] = 0$. The learning machine learns an approximation $\hat{f} = f_{\hat{\theta}}$ such that $Y_i \approx \hat{f}(X_i)$.

In e.g. linear regression we have $f(x) := b^\top x + a$ and $\theta := (a, b)$.

The bias of $\hat{f}$ at X_i is Bias$(\hat{f}|X_i) := \mathbb{E}_Y[\hat{f}(X_i) - f(X_i)]$, the variance of $\hat{f}$ at X_i is Var$(\hat{f}|X_i) := \mathbb{E}_Y[(\hat{f}(X_i) - \mathbb{E}_Y[\hat{f}(X_i)])^2]$ and the MSE of $\hat{f}$ at X_i is MSE$(\hat{f}|X_i) := \mathbb{E}_Y[(\hat{f}(X_i) - Y_i)^2]$.

training data: $\mathcal{D}$
Regression estimator

Lemma 7.4
We have $MSE(\hat{f}|X_i) = Var(\varepsilon_i) + Bias(\hat{f}|X_i)^2 + Var(\hat{f}|X_i)$.

The following theorem is a version of the "no free lunch" theorem in the discrete case.

Theorem 7.4.1: No free lunch

Let $P(\hat{f}(X)|\mathcal{D}, A)$ be the probability that a algorithm A says that x is $\hat{f}(x)$ when training on the data $\mathcal{D}$ and

$$\mathrm{Err}(f, A) := \sum_{\substack{x \notin \mathcal{D} \\ f(x) \neq \hat{f}(x)}} P(\hat{f}(X)|\mathcal{D}, A)$$

the expected off-training set classification error. For arbitrary algorithms, it holds that $\mathbb{E}_f[\mathrm{Err}(f, A_1)] = \mathbb{E}_f[\mathrm{Err}(f, A_2)]$, where $\mathbb{E}_f$ denotes uniform averaging over all possible f.

This result is independent of $\mathcal{D}$, N, A_1, A_2 and the structure of f and A_i!

Remark 7.5 While Occam's razor states that the simplest model or most intuitive method is preferable, the no free lunch theorem states that for each scenario on which a method works better, there are scenarios, for which it is worse. So the choice of the model/method depends on the scenario. Thus, without knowledge on the scenario, it is impossible to predict the prediction error.

So trial and error is the best and only philosophically founded method?

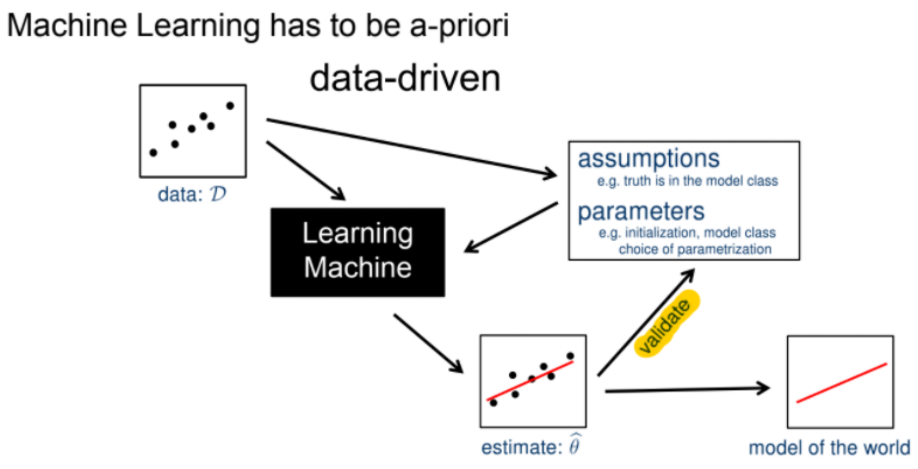

Machine Learning has to be a-priori
data-driven
data: $\mathcal{D}$
assumptions
e.g. truth is in the model class
parameters
e.g. initialization, model class
choice of parametrization
Learning Machine
validate
estimate: $\hat{\theta}$
model of the world

Figure 15: There are many methods assessing features of real world data.

7.6 Methods for model selection

From Occam's razor we have regularisation, i.e. loss = error + complexity and entropy maximisation, i.e. entropy = − information = goodness of fit − complexity (e.g. maximum likelihood, MDL).

An utilitaristic approach is cross-validation:

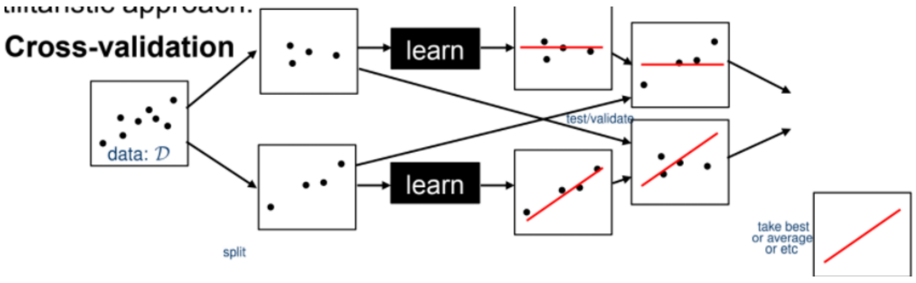
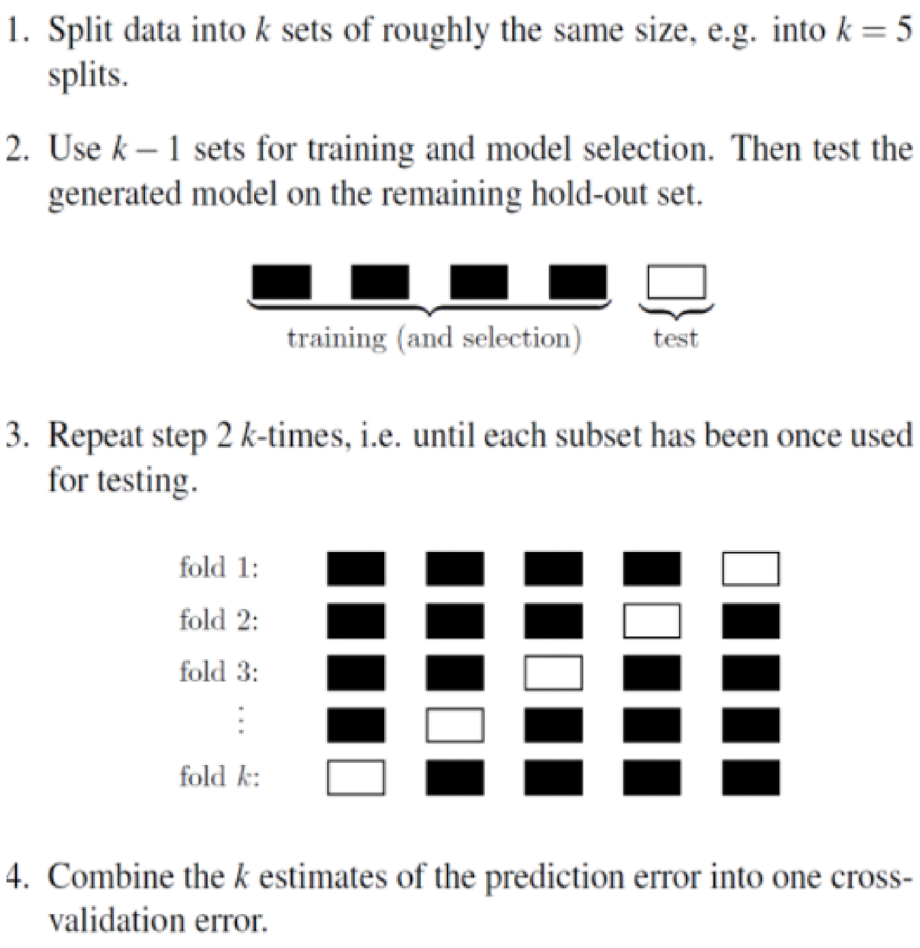


Figure 16: Cross-validation.



1. Split data into k sets of roughly the same size, e.g. into $k = 5$ splits.
2. Use $k-1$ sets for training and model selection. Then test the generated model on the remaining hold-out set.
3. Repeat step 2 k-times, i.e. until each subset has been once used for testing.
4. Combine the k estimates of the prediction error into one cross-validation error.

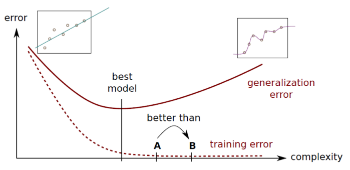
training (and selection)
test
fold 1:
fold 2:
fold 3:
⋮
fold k:

Figure 17: k-fold cross validation

Never use the test set!

8 Learning theory and kernel methods

Recall the Occam's razor principle states that "given two models with the same training set error, the simple one should be preferred because it is likely to have a lower generalisation error". In learning theory, there are three scenarios: regression, classification and density estimation. One learns a function f from examples $((x_k, y_k))_{k=1}^N \in \mathbb{R}^n \times \mathbb{R}^m$ or $\in \mathbb{R}^n \times \{\pm 1\}^m$ generated by $P(x,y)$ such that the expected numbers of errors on the test set (drawn from $P(x,y)$)



$$R[f] = \int \frac{1}{2} |f(x) - y|^2 \, \mathrm{d}P(x,y)$$

is minimal.

But P is unknown, so we need an induction principle.

In empirical risk minimisation (ERM) we replace the average over $P(x,y)$ by an average over the training, i.e. minimising the training error:

$$R_{\text{emp}}[f] = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} |f(x_i) - y_i|^2.$$

By the law of large numbers we have $R_{\text{emp}}[f] \to R[f]$ as $N \to \infty$. But ERM is not consistent, for $N \to \infty$, ERM need not lead to the same results as RM.

Vapnik showed that uniform convergence was needed, leading to the development of VC theory. He show that with a probability of at least $1 - \eta$,

$$R[f] \leqslant R_{\text{emp}} + \sqrt{\frac{d + d \log\left(\frac{2N}{d}\right) - \log\left(\frac{\eta}{4}\right)}{N}}.$$

Structural risk minimisation (SRM) introduces a structure on the set of functions $(f_a)_a$ and minimises the RHS to get a low risk. Here, d is the VC-dimension, which measures the "complexity" of a function class.

An intuitive definition is that the VC-dimension is the maximum number of data points that the function class can always shatter (i.e. classify in any way possible) The VC-dimension of $(\operatorname{sgn}(a + bx + cy))_{a,b,c \in \mathbb{R}}$ is 3, as we can shatter three non-collinear points, but we can never shatter four points.

A counterexample for "simplicity = few parameters" is the two parameter model $x \mapsto a \sin(wx)$, which can fit almost any dataset in $\mathbb{R}$. It is a simple model but does not lead to a low generalisation error. The VC dimension of $(x \mapsto \operatorname{sgn}(\sin(ax))$ is ∞.

HIER FEHLT VIEL

Theorem 8.0.1: Mercer

If k is a continuous kernel of a positive integral operator on $L^2(D)$ and D is compact, i.e. $\int f(x) k(x,y) f(y) \, \mathrm{d}x \, \mathrm{d}y \geqslant 0$ for $f \neq 0$, it can be expanded as

$$k(x,y) = \sum_{i=1}^{N_F} \lambda_i \psi_i(x) \psi_i(y)$$

with $\lambda_i > 0$, $N_F \in \mathbb{N} \cup \{\infty\}$ and we the map $\Phi(x) := ((\sqrt{\lambda_i}\psi_i(x))_{i=1}^{N_F})^\top$ satisfies $\Phi(x) \cdot \Phi(y) = k(x,y)$.

Example 8.1 (Common kernels) Polynomial: $k(x,y) := (xy+c)^d$, RBF (radial basis function): $k(x,y) := \exp\left(\frac{-\|x-y\|^2}{2\sigma^2}\right)$ and inverse multiquadric: $k(x,y) := (\|x-y\|^2+c^2)^{-\frac{1}{2}}$.

Consider

$$\min_w \|w\|^2 \quad \text{subject to} \quad y_i(w\cdot\Phi(x_i)+b) \geqslant 1 \quad \forall i \in \{1,\dots N\},$$

which requires that the training data is separated correctly, otherwise we introduce slack variables:

$$\min_w \|w\|^2 + C\sum_{i=1}^N \xi_i^p \quad \text{subject to} \quad y_i(w\cdot\Phi(x_i)+b) \geqslant 1-\xi,\ \xi_i \geqslant 0 \quad \forall i \in \{1,\dots N\},$$

The LAGRANGIAN is

$$L(w,b,a) := \frac{1}{2}\|w\|^2 - \sum_{i=1}^N a_i(y_i(w\cdot\Phi(x_i)+b)-1).$$

We have

$$\frac{\partial L(w,b,a)}{\partial b} = -\sum_{i=1}^N a_i y_i \overset{!}{=} 0$$

and

$$\frac{\partial L(w,b,a)}{\partial w} = w - \sum_{i=1}^N a_i y_i \Phi(x_i) \overset{!}{=} 0.$$

Thus the dual objective function is

$$\frac{1}{2}\left\|\sum_{i=1}^N a_i y_i \Phi(x_i)\right\|^2 - \sum_{i=1}^N a_i y_i\left(\left\langle \sum_{j=1}^N a_j y_j \Phi(x_j), \Phi(x_i)\right\rangle + b\right) + \sum_{i=1}^N a_i,$$

which simplifies to $\sum_{i=1}^N a_i - \sum_{i,j=1}^N a_i a_j y_i y_j k(x_i,x_j)$, so the dual problem is

$$\max_{a\geqslant 0} \sum_{i=1}^N a_i - \sum_{i,j=1}^N a_i a_j y_i y_j k(x_i,x_j)$$

$$\text{subject to} \quad 0 \leqslant a_i \leqslant C \quad \forall i \in \{1,\dots,N\} \text{ and } \sum_{i=1}^N a_i y_i = 0.$$

If $y_i(w\cdot\Phi(x_i)+b) > 1$, then $a_i = 0$ and x_i is irrelevant. But if $y_i(w\cdot\Phi(x_i)+b) = 1$, i.e. on / in margin, then x_i is a support vector.

Implementation issues: working set methods

We introduce the following matrix notation: let $a = (a_1,\dots,a_M)^\top$, y analogously and $H := (y_i y_j k(x_i,x_j))_{i,j}$ and $\mathbb{1}$ be a vector of length M consisting of ones.

The dual problem then becomes

$$\max_a \mathbb{1}^\top a - \frac{1}{2}a^\top H a \quad \text{subject to} \quad y^\top a = 0,\ a - C\mathbb{1} \leqslant 0,\ a \geqslant 0$$

hier kommen jetzt noch viele details.

One-class SVMs try to fit a hypersphere around the data

9 Scalable machine learning

9.1 Introduction

Scalable machine learning algorithms scale efficiently with given problem properties. In software systems, scalable is typically defined with respect to resources; a scalable system is able to solve a problem faster with more resources. In machine learning, it is more complex: approximated solutions are allowed. Here, "“scalable” machine learning is almost always based on finding more efficient algorithms, and most often, approximations to the original algorithm which can be computed much more efficiently."

One needs scalable machine learning if the default algorithm choice can't be used due to resource (e.g. time, memory) constraints. Instead one uses a scalable algorithm; one that is more complex algorithm and that fits these constraints.

Examples for constraints are complex models (Chess), large data sets (translation) or complex learning.

The typical resource constraints in ML are time, computational power and memory. The temporal and spatial needs of algorithms depend on the number of samples N, the number of input dimensions / features D, the number of output dimensions C and the prediction quality (e.g. loss, accuracy) epochs. E.g. Full batch logistic regression takes $O(N \cdot D \cdot C \cdot \text{epochs})$.

Training is the bottleneck: a lot of optimisation problems are not decomposable ad thus hard to parallelise.

Example 9.2 (Faster learning algorithms) Nonlinear kernel learning involves the solution of a quadratic problem. Sequential minimal optimisation (SMO) chooses two samples (x_1, y_1) and (d_2, y_2) and then solves analytically. Smart ways to choose samples result in a faster algorithm.

Example 9.3 (Approximated features) Kernel machines scale quadratically with the number of samples; they are in $O(N^2)$. The kernel function, which measures the similarity between samples can be approximated: (Fourier) random features approximate the kernel function with a Fourier transform. Kernel machines the scale with $O(ND)$ and can be used in primal space.

So, in a nutshell, scalable machine learning is not parallelised standard methods but instead specialised algorithms such as faster learning algorithms (SMO) approximated optimisation(SGD) (and parallelisation)

Remark 9.4 (Why are approximations good enough?)
Machine learning is complicated due to many unreliable components: noisy data, noisy optimisation processes, limit machine precision and overfitting.

pro	con
scales when needed	Amdahl's law, overhead, parallel slowdown
	Algorithm complexity
	technical challenges

Table 1: Pros and cons of parallel/distributed solutions

9.5 Algorithms

Example 9.6 ()
Gradient based optimisation] Gradient based optimisation drives machine learning, as there often are no closed form solutions (e.g. linear logistic regression but not linear least squares) and one can use the chain rule and backpropagation to inspect the process and it scales. Given an error function $E(f) := \int \ell(f(x), y)\,\mathrm{d}P(z)$ or the discrete version $E_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$ for a measure P a loss function ℓ and data x with labels y, we can iteratively improve with derivatives:

$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla w Q(z_i, w_t), \qquad \text{where } Q(z, w) := \ell(f_w(x), y).$$

This converges given the right learning rate $\gamma > 0$ and other assumptions.

Stochastic gradient descent is $w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t)$, i.e. gradient of a single sample. It is stochastic because it is a noisy approximation of the true gradient. In this case convergence assumptions are e.g. a decaying learning rate.

In practice, Mini-batches reduce noise and improve performance and there are more evolved update schemes like (Nesterov)-momentum.

Example 9.7 (Parallelising and distributing gradient based optimisation)
Convergence speed has biggest impact on the training time: No parallelisation strategy will solve a badly conditioned optimisation problem fast. There a lot of "tricks of the trade" like shuffling the training set or experimenting with the learning rate on smaller sample of training set, or normalising the inputs, right activation function, individual learning rates.

slide 22.

References

[1] Bishop 2007

[2] this

Index