Statistical Machine Learning: Part 2

Dino Sejdinovic

February 2021

Supervised Learning

Supervised learning:

- In addition to the observations of X, we have access to their response variables / labels $Y \in \mathcal{Y}$: we observe $\{(x_i, y_i)\}_{i=1}^n$.
- Types of supervised learning:
  - Classification: discrete responses, e.g. $\mathcal{Y} = \{+1, -1\}$ or $\{1, \ldots, K\}$.
  - Regression: a numerical value is observed and $\mathcal{Y} = \mathbb{R}$.

The goal is to accurately predict the response Y on new observations of X, i.e., to **learn a function** $f : \mathbb{R}^p \to \mathcal{Y}$, such that $f(X)$ will be close to the true response Y.

Loss function

- Suppose we made a prediction $\hat{Y} = f(X) \in \mathcal{Y}$ based on observation of X.
- How good is the prediction? We can use a **loss function** $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ to formalize the quality of the prediction.
- Typical loss functions:
  - **Misclassification loss** (or **0-1 loss**) for classification

$$L(y, f(x)) = \begin{cases} 0 & f(x) = y \\ 1 & f(x) \neq y \end{cases} .$$

  - **Squared loss** for regression

$$L(y, f(x)) = (f(x) - y)^2 .$$

- Many other choices are possible.

Loss functions for binary classification

Classes are denoted -1 and $+1$. Class prediction is $\text{sign}(f(x))$, whereas the magnitude of $f(x)$ represents the “confidence".

- $0/1$ loss $L(y, f(x)) = \mathbb{1}\{yf(x) \leq 0\}$,
  (also called misclassification loss, optimal solution is called the **Bayes classifier** and is given by $f(x) = \text{argmax}_{k \in \{0,1\}} \mathbb{P}(Y = k | X = x)$),
- hinge loss $L(y, f(x)) = (1 - yf(x))_+$
  (used in **support vector machines** - leads to sparse solutions),
- exponential loss $L(y, f(x)) = e^{-yf(x)}$
  (used in **boosting** algorithms - Adaboost),
- logistic loss $L(y, f(x)) = \log\left(1 + e^{-yf(x)}\right)$
  (used in **logistic regression**, associated with a probabilistic model).

The loss can penalize misclassification (wrong sign) as well as the overconfident misclassification (wrong sign and large magnitude) and even underconfident correct classification (correct sign but small magnitude).

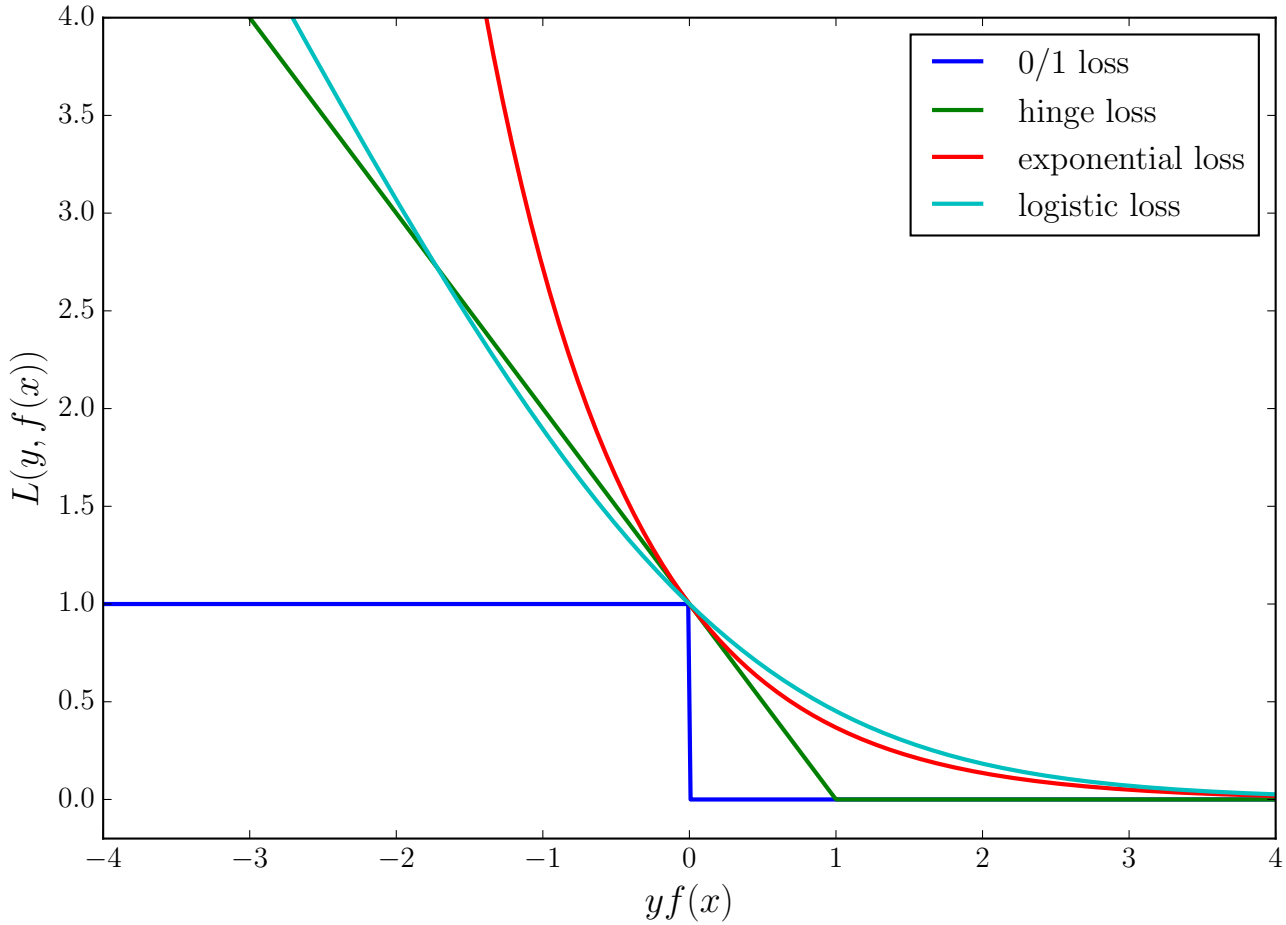


Figure: Loss functions for binary classification

Loss functions for regression

- squared loss: $L(y, f(x)) = (y - f(x))^2$
  (least squares regression: optimal f is the conditional mean $\mathbb{E}[Y|X = x]$),
- absolute loss: $L(y, f(x)) = |y - f(x)|$
  (less sensitive to outliers: optimal f is the conditional median $\text{med}[Y|X = x]$),
- τ-pinball loss: $L(y, f(x)) = 2\max\{\tau(y - f(x)), (\tau - 1)(y - f(x))\}$ for $\tau \in (0, 1)$
  (quantile regression: optimal f is the τ-quantile of $p(y|X = x)$),
- ϵ-insensitive (Vapnik) loss: $L(y, f(x)) = \begin{cases} 0, \text{ if } |y - f(x)| \leq \epsilon, \\ |y - f(x)| - \epsilon, \text{ otherwise.} \end{cases}$
  (**support vector regression** - leads to sparse solutions).

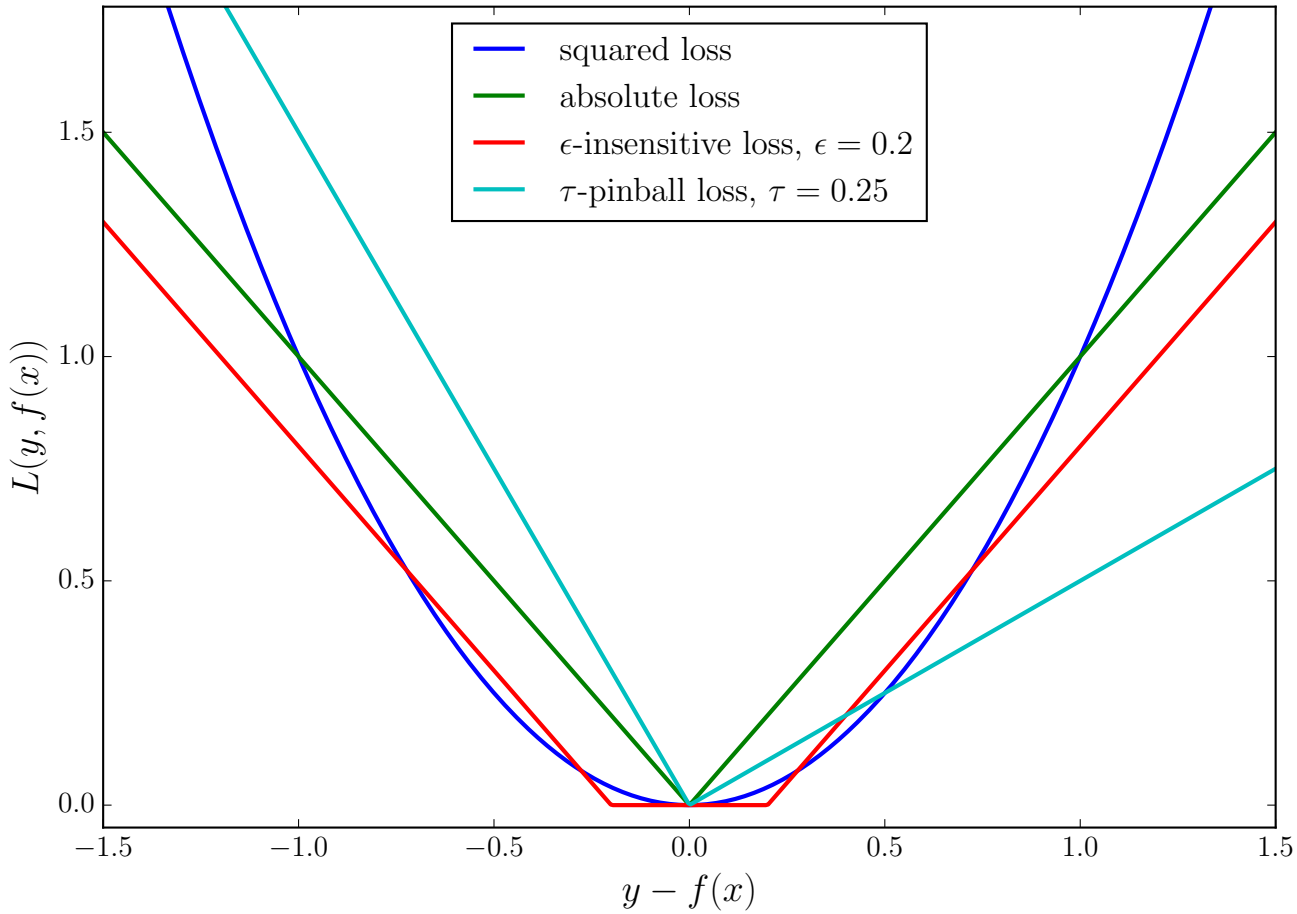


Figure: Loss functions for regression

Quantile regression example

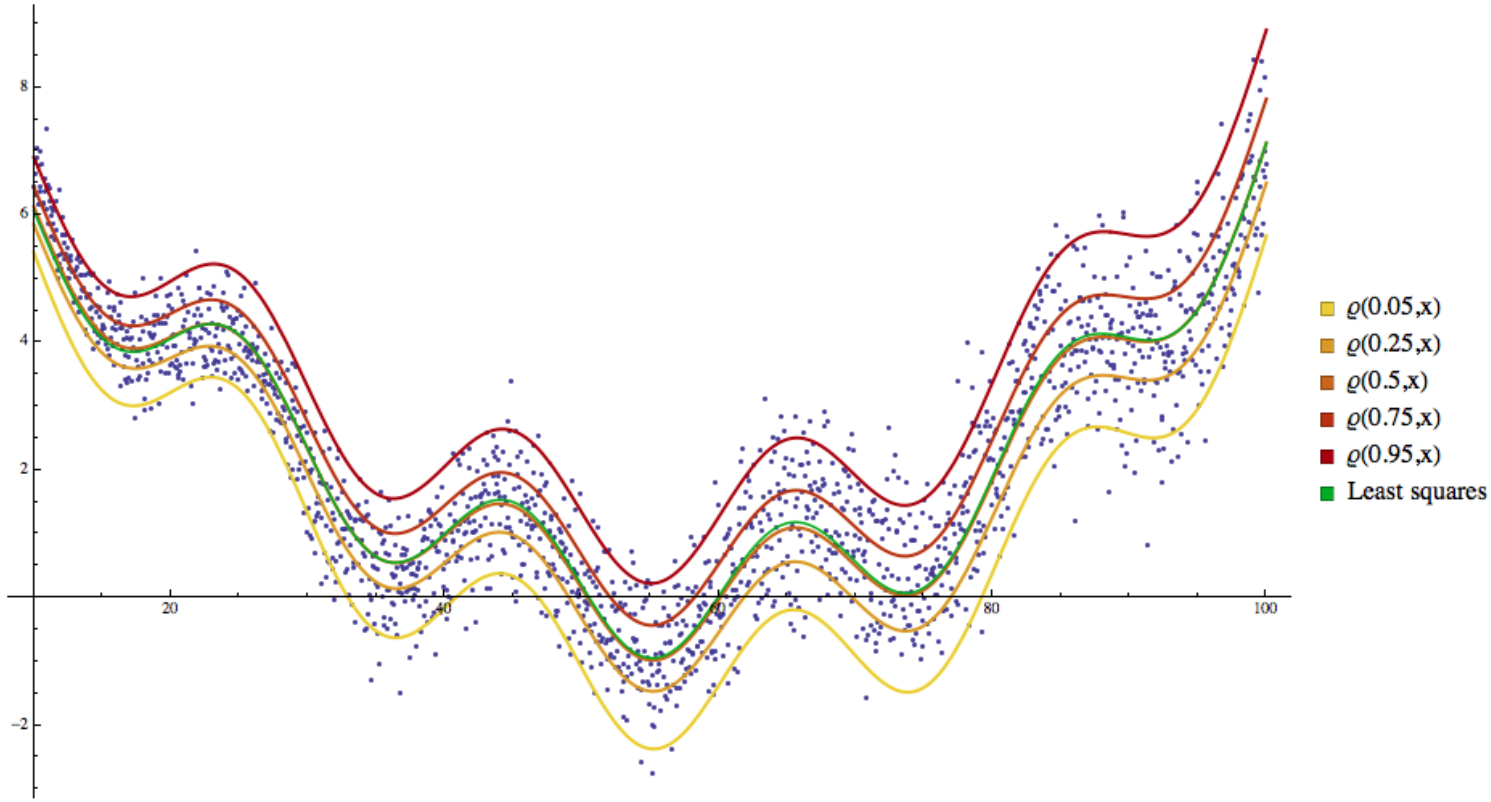


Figure: Quantiles

Risk

- paired observations $\{(x_i, y_i)\}_{i=1}^n$ viewed as i.i.d. realizations of a random variable (X, Y) on $\mathcal{X} \times \mathcal{Y}$ with joint distribution P_{XY}

Risk

For a given loss function L, the **risk** R of a learned function f is given by the expected loss

$$R(f) = \mathbb{E}_{P_{XY}} \left[L(Y, f(X)) \right],$$

where the expectation is with respect to the true (unknown) joint distribution of (X, Y).

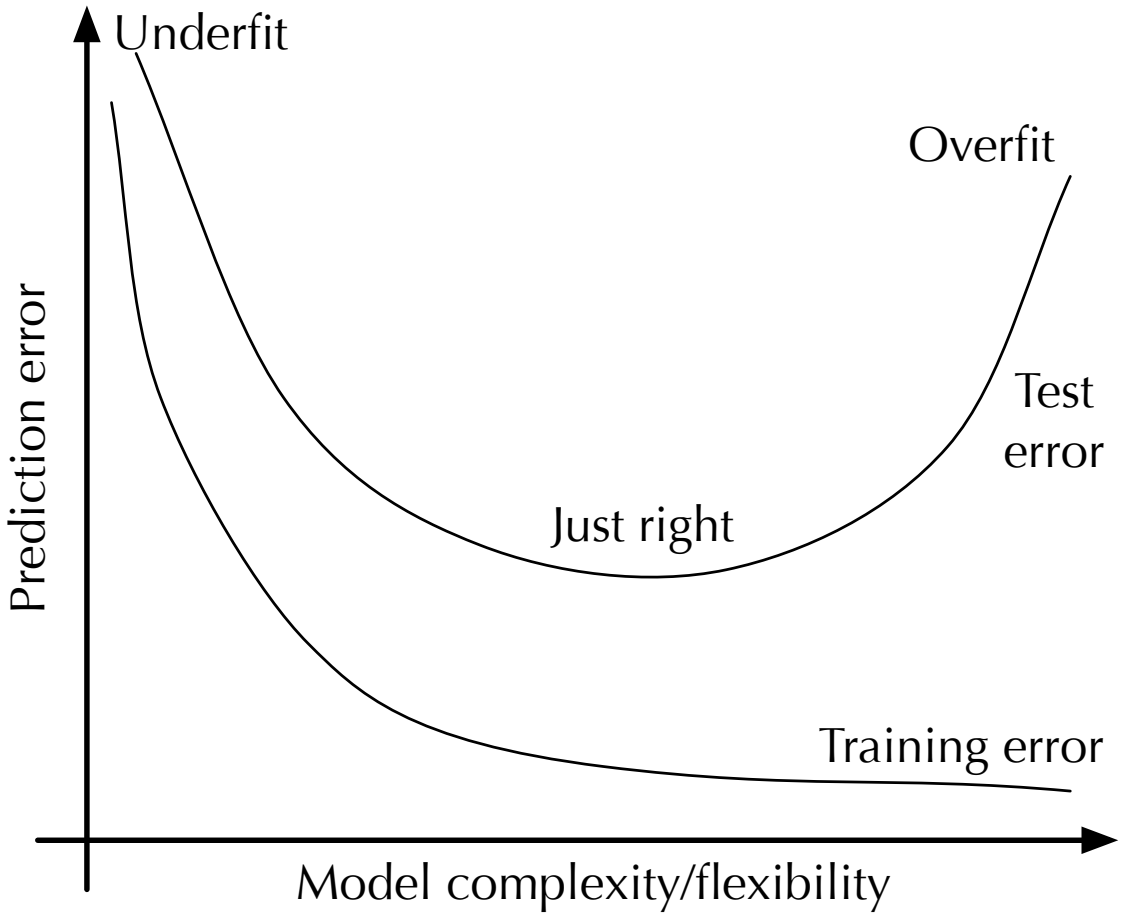
- The risk is unknown, but we can compute the **empirical risk**:

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)).$$

Generalization

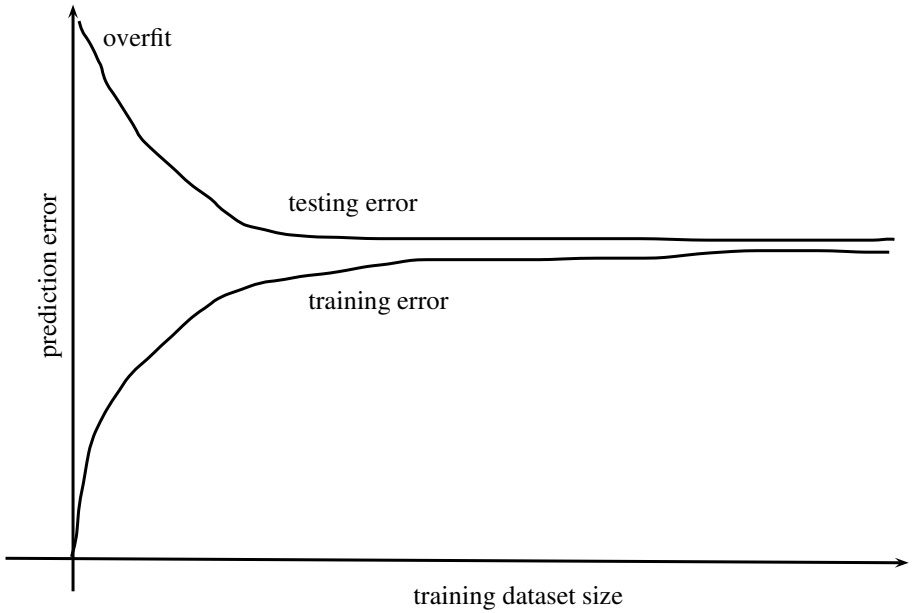
- Generalization ability: what is the out-of-sample error of learner f?
- training error $\neq$ testing error.
- We learn f by minimizing some variant of empirical risk $R^{\text{emp}}(f)$- what can we say about the true risk $R(f)$?
- Two important factors determining generalization ability:
  - Model complexity
  - Training data size
- To control overfitting, we need to reduce the model complexity to an appropriate level, i.e. **regularize learning**.

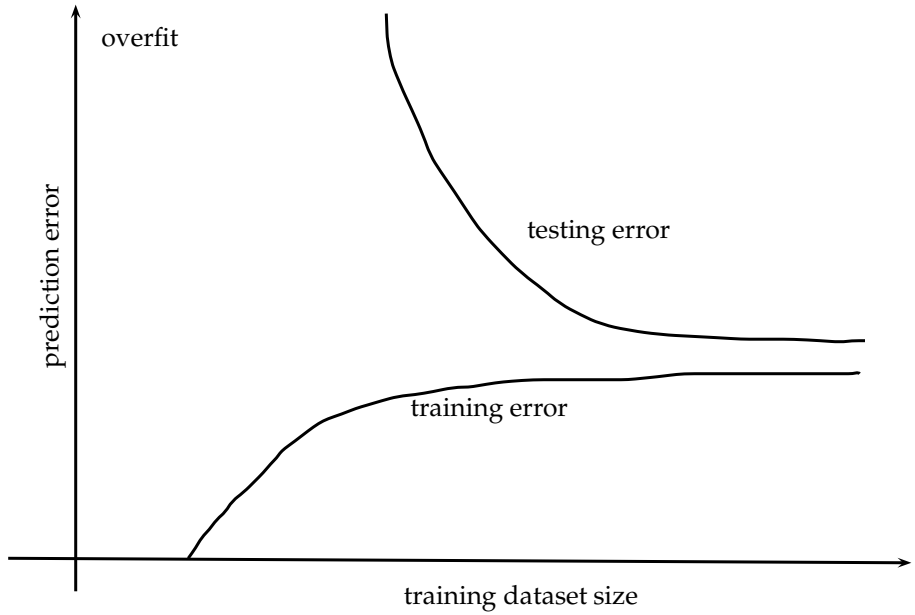
Learning Curves



The relationship between training and test error as a function of model complexity.

Learning Curves





Fixed model complexity, varying dataset size.
Two models: one simple, one complex. Which is which?

Bias-Variance Tradeoff

- Where does the prediction error come from?
- Example: Squared loss in regression: $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \mathbb{R}$,

$$L(Y, f(X)) = (Y - f(X))^2$$

- Optimal f is the conditional mean:

$$f_*(x) = \mathbb{E}[Y|X = x]$$

- Follows from $R(f) = \mathbb{E}_X \mathbb{E}\left[(Y - f(X))^2 \middle| X\right]$ and

$$\begin{aligned} & \mathbb{E}\left[(Y - f(X))^2 \middle| X = x\right] \\ = \; & \mathbb{E}\left[Y^2 \middle| X = x\right] - 2f(x)\,\mathbb{E}[Y|X = x] + f(x)^2 \\ = \; & \mathsf{Var}[Y|X = x] + (\mathbb{E}[Y|X = x] - f(x))^2 . \end{aligned}$$

Bias-Variance Tradeoff

- Optimal risk is the intrinsic conditional variability of Y (noise):

$$R(f_*) = \mathbb{E}_X \left[\mathsf{Var} \left[Y | X \right] \right]$$

- **Excess risk**:

$$R(f) - R(f_*) = \mathbb{E}_X \left[\left(f(X) - f_*(X) \right)^2 \right]$$

- How does the excess risk behave **on average**?
- Consider a randomly selected dataset $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^n$ and $f^{(\mathcal{D})}$ trained on $\mathcal{D}$ using a model (hypothesis class) $\mathcal{H}$.

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} \left[R(f^{(\mathcal{D})}) - R(f_*) \right] &= \mathbb{E}_{\mathcal{D}} \mathbb{E}_X \left[\left(f^{(\mathcal{D})}(X) - f_*(X) \right)^2 \right] \\ &= \mathbb{E}_X \mathbb{E}_{\mathcal{D}} \left[\left(f^{(\mathcal{D})}(X) - f_*(X) \right)^2 \right]. \end{aligned}$$

Bias-Variance Tradeoff

- Denote $\bar{f}(x) = \mathbb{E}_{\mathcal{D}} f^{(\mathcal{D})}(x)$ (average decision function over all possible datasets)

$$\mathbb{E}_{\mathcal{D}}\left[\left(f^{(\mathcal{D})}(X) - f_*(X)\right)^2\right] = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(f^{(\mathcal{D})}(X) - \bar{f}(X)\right)^2\right]}_{\mathrm{Var}_X(\mathcal{H},n)} + \underbrace{\left(\bar{f}(X) - f_*(X)\right)^2}_{\mathrm{Bias}^2_X(\mathcal{H},n)}$$

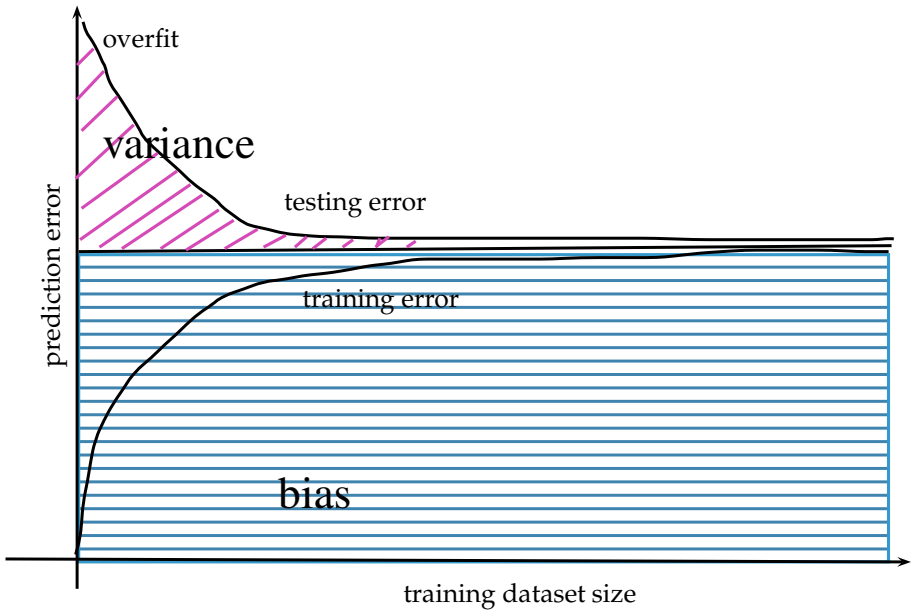
Now,

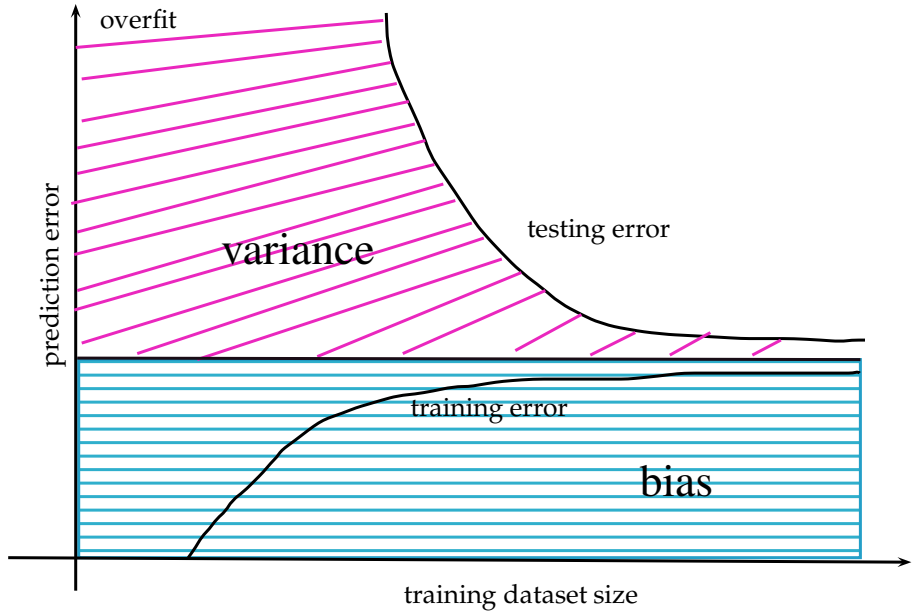
$$\mathbb{E}_{\mathcal{D}} R(f^{(\mathcal{D})}) = R(f_*) + \mathbb{E}_X \mathrm{Var}_X(\mathcal{H}, n) + \mathbb{E}_X Bias^2_X(\mathcal{H}, n)$$

Where does the prediction error come from?

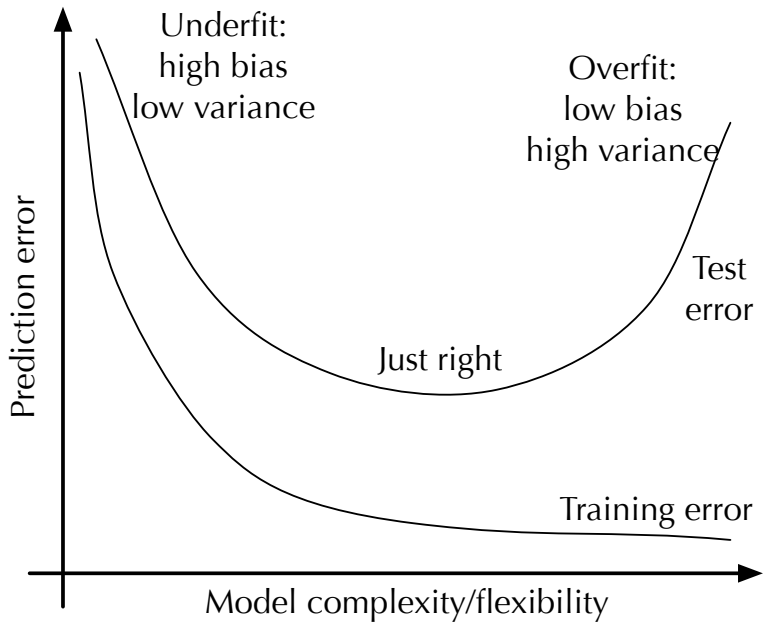
- **Noise**: Intrinsic difficulty of regression problem.
- **Bias**: How far away is the best learner in the model (average learner over all possible datasets) from the optimal one?
- **Variance**: How variable is our learning method if given different datasets?

Learning Curves





Building models to trade bias with variance



- Building a machine learning model involves trading between its bias and variance.
  - Bias reduction at the expense of a variance increase: building more complex models, e.g. adding nonlinear features and additional parameters, increasing the number of hidden units in neural nets, using decision trees with larger depth.
  - Variance reduction at the expense of a bias increase: increasing the regularization parameter, early stopping, using k-nearest neighbours with larger k.

Regularisation

- Flexible models for high-dimensional problems require many parameters.
- With many parameters, learners can easily overfit.
- **regularisation**: Limit flexibility of model to prevent overfitting.
- Add term **penalizing large values of parameters** θ.

$$\min_{\theta} \hat{R}(f_\theta) + \lambda \|\theta\|_\rho^\rho = \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(x_i)) + \lambda \|\theta\|_\rho^\rho$$

where $\rho \geq 1$, and $\|\theta\|_\rho = (\sum_{j=1}^{p} |\theta_j|^\rho)^{1/\rho}$ is the L_ρ norm of θ (also of interest when $\rho \in [0, 1)$, but is no longer a norm).

- Also known as **shrinkage** methods—parameters are shrunk towards 0.
- λ is a **tuning parameter** (or **hyperparameter**) and controls the amount of regularisation, and resulting complexity of the model.

Types of regularisation

- **Ridge regression** / **Tikhonov regularisation**: $\rho = 2$ (Euclidean norm)
- **LASSO**: $\rho = 1$ (Manhattan norm)
- **Sparsity-inducing** regularisation: $\rho \leq 1$ (nonconvex for $\rho < 1$)
- **Elastic net** regularisation: mixed L_1/L_2 penalty:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(x_i)) + \lambda \left[(1-\alpha) \|\theta\|_2^2 + \alpha \|\theta\|_1 \right]$$

Regularized Least Squares

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top x_i - b)^2 + \lambda \|w\|_2^2.$$

Note the rescaling of the regularisation term and that the bias term b is not included in the regularisation. This is important as otherwise the predictions would depend on the origin for the response variables y (i.e. adding a constant c to each target would result in different predictions from simply shifting the original predictions by c).
Differentiating and setting to zero gives again the closed form solution for w:

$$\hat{w} = \left(\mathbf{X}^\top \mathbf{X} + n\lambda I\right)^{-1} \mathbf{X}^\top \mathbf{y}.$$

L_1 promotes sparsity

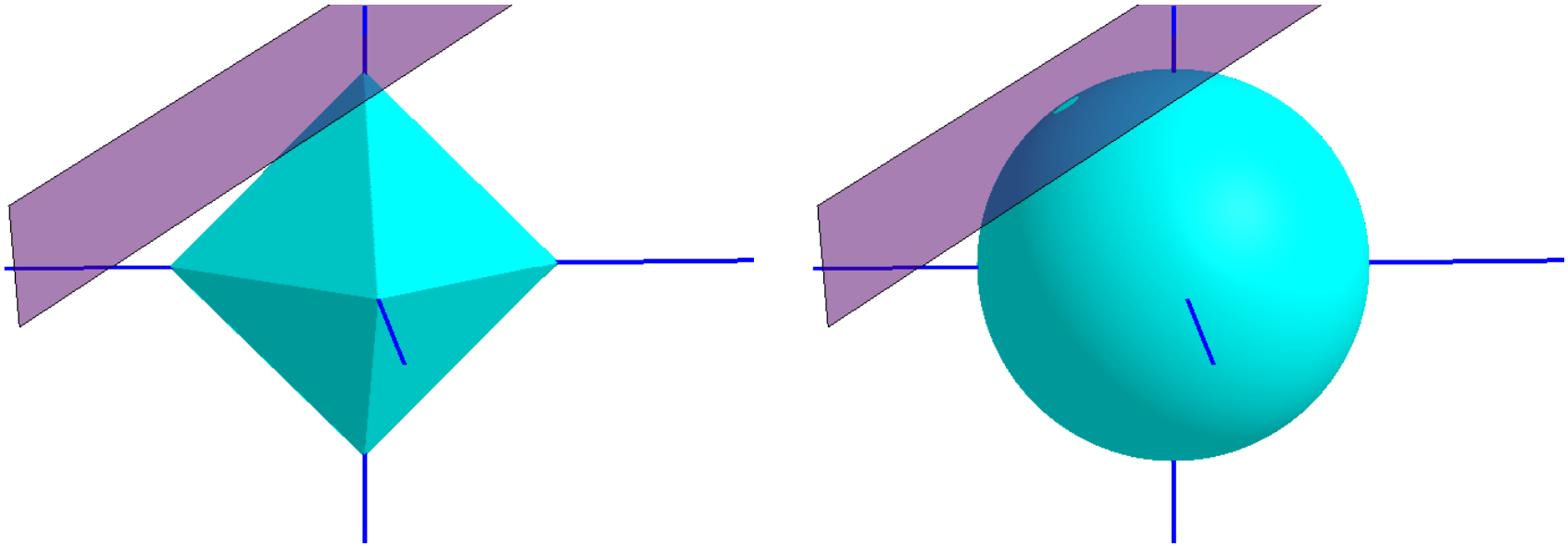


Figure: The intersection between the L_1 (left) and the L_2 (right) ball with a hyperplane.

L_1 regularisation often leads to optimal solutions with many zeros, i.e., the regression function depends only on the (small) number of features with non-zero parameters.

figure from M. Elad, Sparse and Redundant Representations, 2010.

Reading

- Hastie et al, Sections 2.4, 2.5, 2.6 & 2.9,
- Bishop, Sections 3.1 & 3.2.

Risk and the goal of supervised learning

Setup

input-output observations $\{(x_i, y_i)\}_{i=1}^n$ are viewed as i.i.d. realizations of a random pair of variables (X, Y) on $\mathbb{R}^p \times \mathcal{Y}$ with some joint distribution P_{XY} (examples are independent samples from a population).

The goal of supervised learning

Given a loss function L, find function f which minimizes the **risk**:

$$R(f) = \mathbb{E}_{P_{XY}} \left[L(Y, f(X)) \right],$$

where the expectation is with respect to the true (unknown) joint distribution of (X, Y).

The Bayes (Optimal) Classifier

- What is the optimal classifier if the joint distribution (X, Y) were **known**?
- The density g of X can be written as a mixture of K components (corresponding to each of the classes):

$$g(x) = \sum_{k=1}^{K} \pi_k g_k(x),$$

where, for $k = 1, \ldots, K$,

  - $\mathbb{P}(Y = k) = \pi_k$ are the class probabilities,
  - $g_k(x)$ is the conditional density of X, given $Y = k$.

- The **Bayes classifier** $f_{\text{Bayes}} : x \mapsto \{1, \ldots, K\}$ is the one with minimum risk (despite the name, Bayes classifier is **not Bayesian**!):

$$\begin{aligned} R(f) =& \mathbb{E}\left[L(Y, f(X))\right] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[L(Y, f(X))|X]\right] \\ =& \int_{\mathcal{X}} \mathbb{E}\left[L(Y, f(X))|X = x\right] g(x) dx \end{aligned}$$

- The minimum risk attained by the Bayes classifier is called the **Bayes risk**.
- Minimizing $\mathbb{E}[L(Y, f(X))|X = x]$ separately for each x suffices.

The Bayes Classifier

- Consider the 0-1 loss.
- The risk simplifies to:

$$\begin{aligned}\mathbb{E}\Big[L(Y, f(X))\Big|X = x\Big] =& p(Y \neq f(X)|X = x)\\ =& 1 - p(Y = f(X)|X = x)\end{aligned}$$

- The risk is minimized by choosing the class with the greatest probability given the observation:

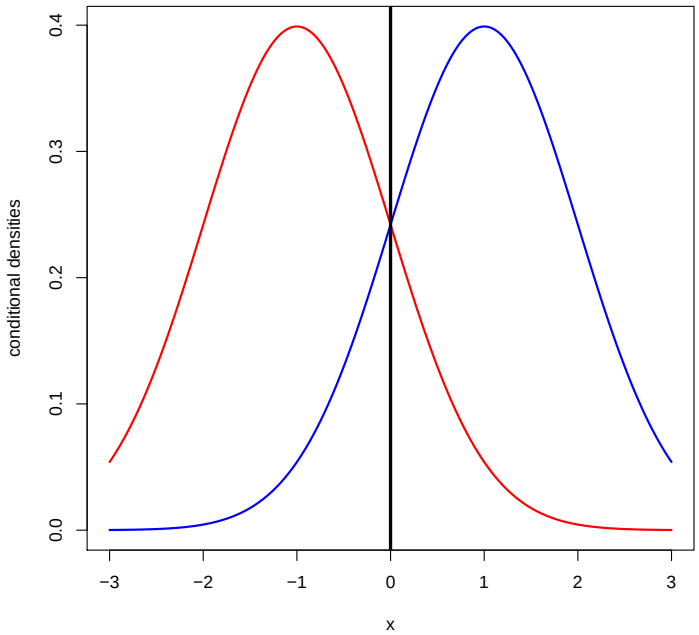
$$\begin{aligned}f_{\text{Bayes}}(x) &= \underset{k=1,\ldots,K}{\arg\max}\, p(Y = k|X = x) = \underset{k=1,\ldots,K}{\arg\max}\, \frac{p(Y = k, X = x)}{p(x)}\\ &= \underset{k=1,\ldots,K}{\arg\max}\, \frac{\pi_k g_k(x)}{\sum_{j=1}^{K} \pi_j g_j(x)} = \underset{k=1,\ldots,K}{\arg\max}\, \pi_k g_k(x).\end{aligned}$$

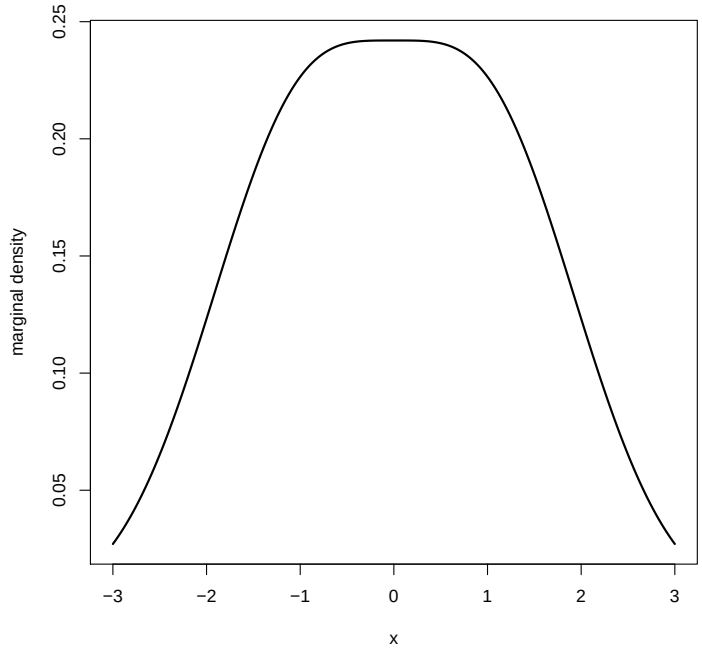
- The functions $x \mapsto \pi_k g_k(x)$ are called **discriminant functions**. The discriminant function with maximum value determines the predicted class of x.

The Bayes Classifier: Example

A simple two Gaussians example: Suppose $X \sim \mathcal{N}(\mu_Y, 1)$, where $\mu_1 = -1$ and $\mu_2 = 1$ and assume equal class probabilities $\pi_1 = \pi_2 = 1/2$.

$$g_1(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x+1)^2}{2}\right) \quad \text{and} \quad g_2(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{2}\right).$$

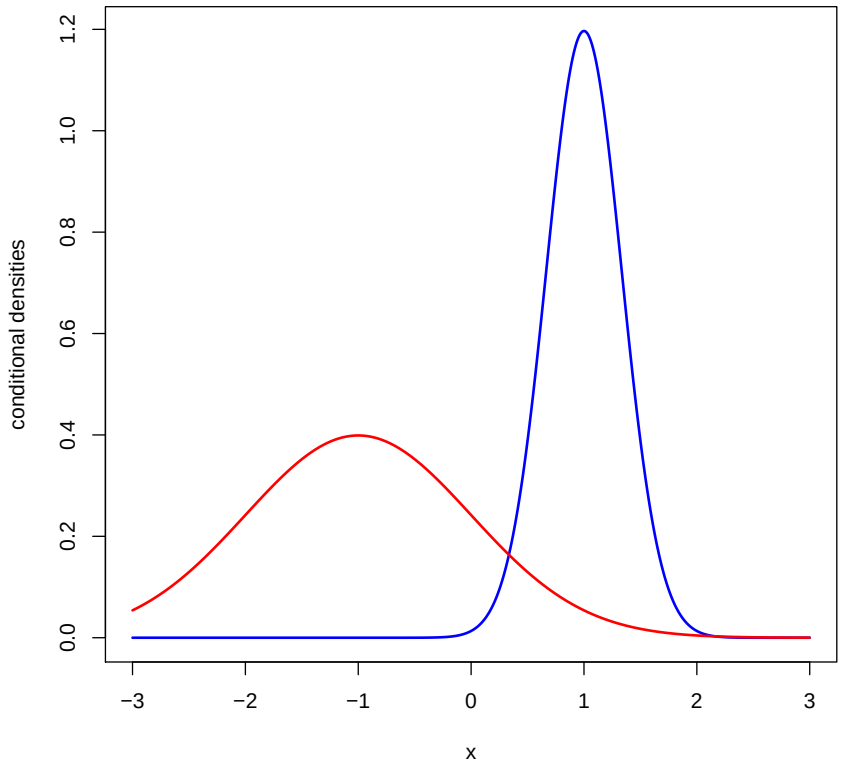


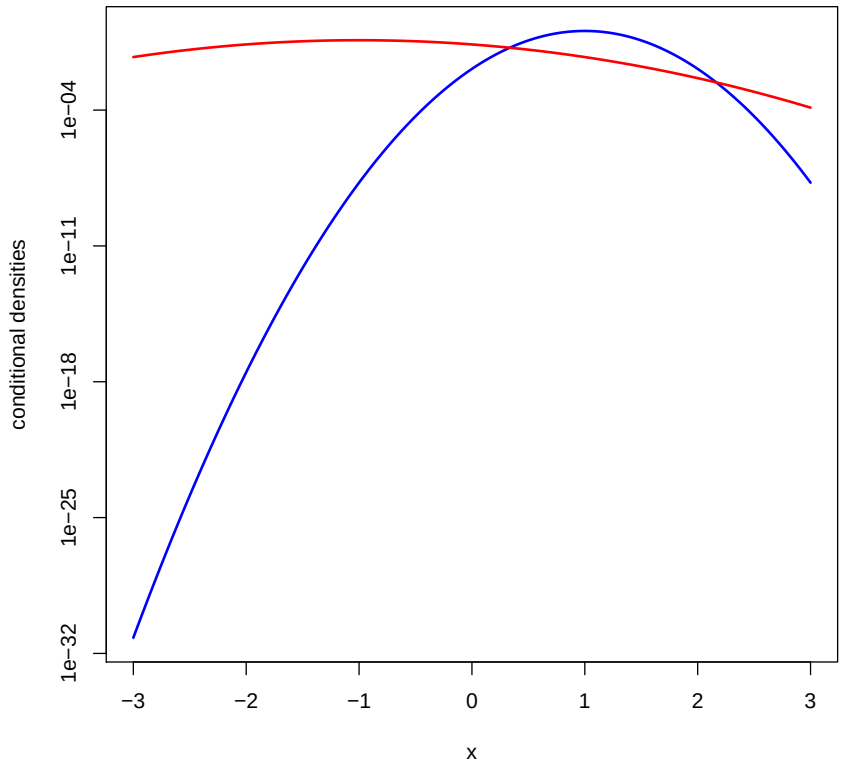


Optimal classification is $f_{\text{Bayes}}(x) = \underset{k=1,\ldots,K}{\arg\max}\ \pi_k g_k(x) = \begin{cases} 1 & \text{if } x < 0, \\ 2 & \text{if } x \geq 0. \end{cases}$

The Bayes Classifier: Example

How do you classify a new observation x if now the standard deviation is still 1 for class 1 but $1/3$ for class 2?





Looking at density in a log-scale, optimal classification is to select class 2 if and only if $x \in [0.34, 2.16]$.

Plug-in Classification

- The Bayes Classifier:

$$f_{\text{Bayes}}(x) \quad = \quad \underset{k=1,\ldots,K}{\arg\max}\, \pi_k g_k(x).$$

- We know neither the conditional densities g_k nor the class probabilities π_k!
- The **plug-in classifier** chooses the class

$$f(x) = \underset{k=1,\ldots,K}{\arg\max}\, \hat{\pi}_k \hat{g}_k(x),$$

- where we plugged in
  - estimates $\hat{\pi}_k$ of π_k and $k = 1, \ldots, K$ and
  - estimates $\hat{g}_k(x)$ of conditional densities,
- **Linear Discriminant Analysis** is an example of plug-in classification.

Linear Discriminant Analysis

- **LDA** is the most well-known and simplest example of plug-in classification.
- Assume multivariate normal conditional density $g_k(x)$ for each class k:

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma),$$

$$g_k(x) = (2\pi)^{-p/2}|\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)\right),$$

  - each class can have a **different mean** μ_k,
  - all classes share the **same covariance** Σ.
- For an observation x, the k-th log-discriminant function is

$$\log \pi_k g_k(x) = c + \log \pi_k - \frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)$$

  The quantity $(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)$ is the squared **Mahalanobis distance** between x and μ_k.
- If $\Sigma = I_p$ and $\pi_k = \frac{1}{K}$, LDA simply chooses the class k with the nearest (in the Euclidean sense) class mean.

Linear Discriminant Analysis

- Expanding the term $(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)$,

$$\log \pi_k g_k(x) = c + \log \pi_k - \frac{1}{2}\left(\mu_k^\top \Sigma^{-1}\mu_k - 2\mu_k^\top \Sigma^{-1}x + x^\top \Sigma^{-1}x\right)$$
$$= c' + \log \pi_k - \frac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \mu_k^\top \Sigma^{-1}x$$

- Setting $a_k = \log(\pi_k) - \frac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k$ and $b_k = \Sigma^{-1}\mu_k$, we obtain

$$\log \pi_k g_k(x) = c' + a_k + b_k^\top x$$

  i.e. a **linear** discriminant function in x.
- Consider choosing class k over k':

$$a_k + b_k^\top x > a_{k'} + b_{k'}^\top x \qquad \Leftrightarrow \qquad a_\star + b_\star^\top x > 0$$

  where $a_\star = a_k - a_{k'}$ and $b_\star = b_k - b_{k'}$.
- The Bayes classifier thus partitions $\mathcal{X}$ into regions with the same class predictions via **separating hyperplanes**.
- The Bayes classifier under these assumptions is more commonly known as the **LDA classifier**.

Parameter Estimation

- How to estimate the parameters of the LDA model?
- We can achieve this by maximum likelihood.
- Let $n_k = \#\{j : y_j = k\}$ be the number of observations in class k.

$$\ell(\pi, (\mu_k)_{k=1}^K, \Sigma) = \log p\left((x_i, y_i)_{i=1}^n \,|\pi, (\mu_k)_{k=1}^K, \Sigma\right) = \sum_{i=1}^n \log \pi_{y_i} g_{y_i}(x_i)$$

$$= c + \sum_{k=1}^K \sum_{j:y_j=k} \log \pi_k - \frac{1}{2}\left(\log|\Sigma| + (x_j - \mu_k)^\top \Sigma^{-1}(x_j - \mu_k)\right)$$

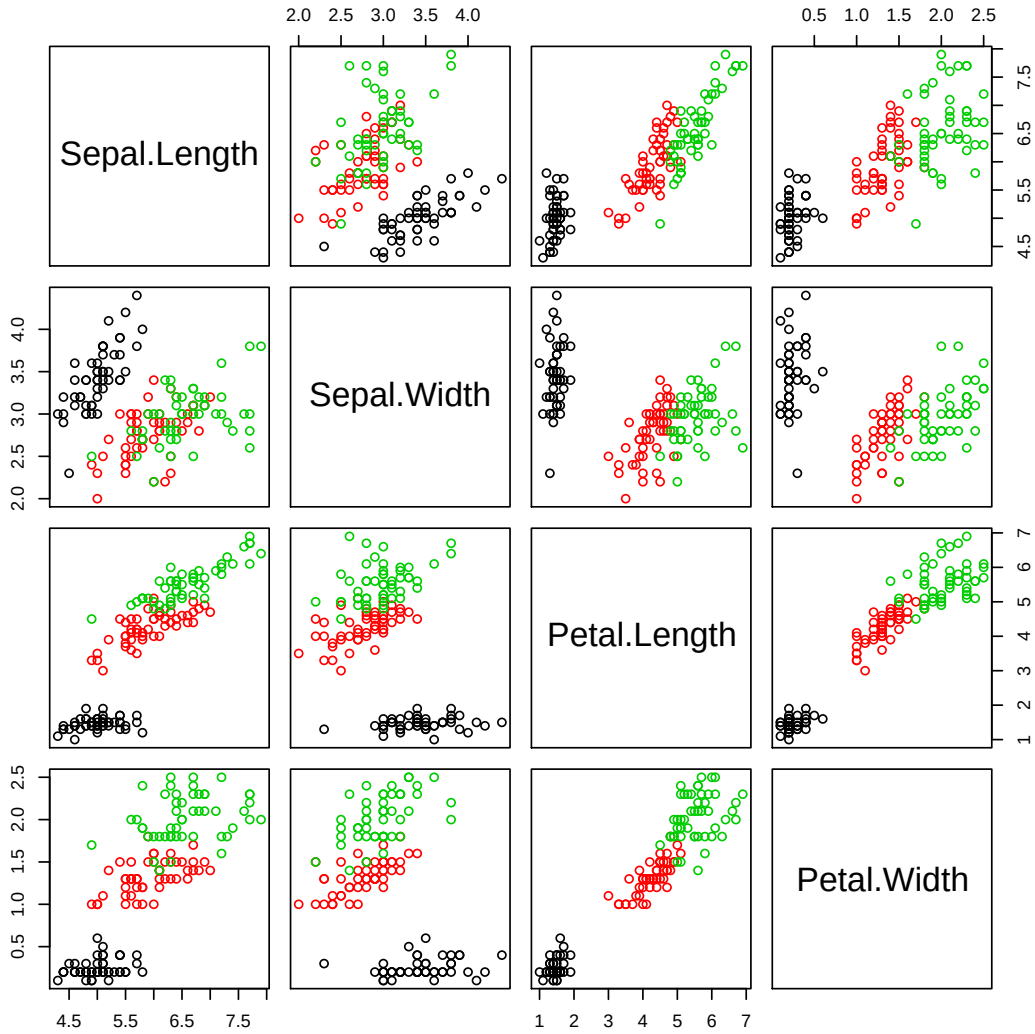
ML estimates:

$$\hat{\pi}_k = \frac{n_k}{n} \qquad \hat{\mu}_k = \frac{1}{n_k} \sum_{j:y_j=k} x_j$$

$$\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^K \sum_{j:y_j=k} (x_j - \hat{\mu}_k)(x_j - \hat{\mu}_k)^\top$$

Iris Dataset

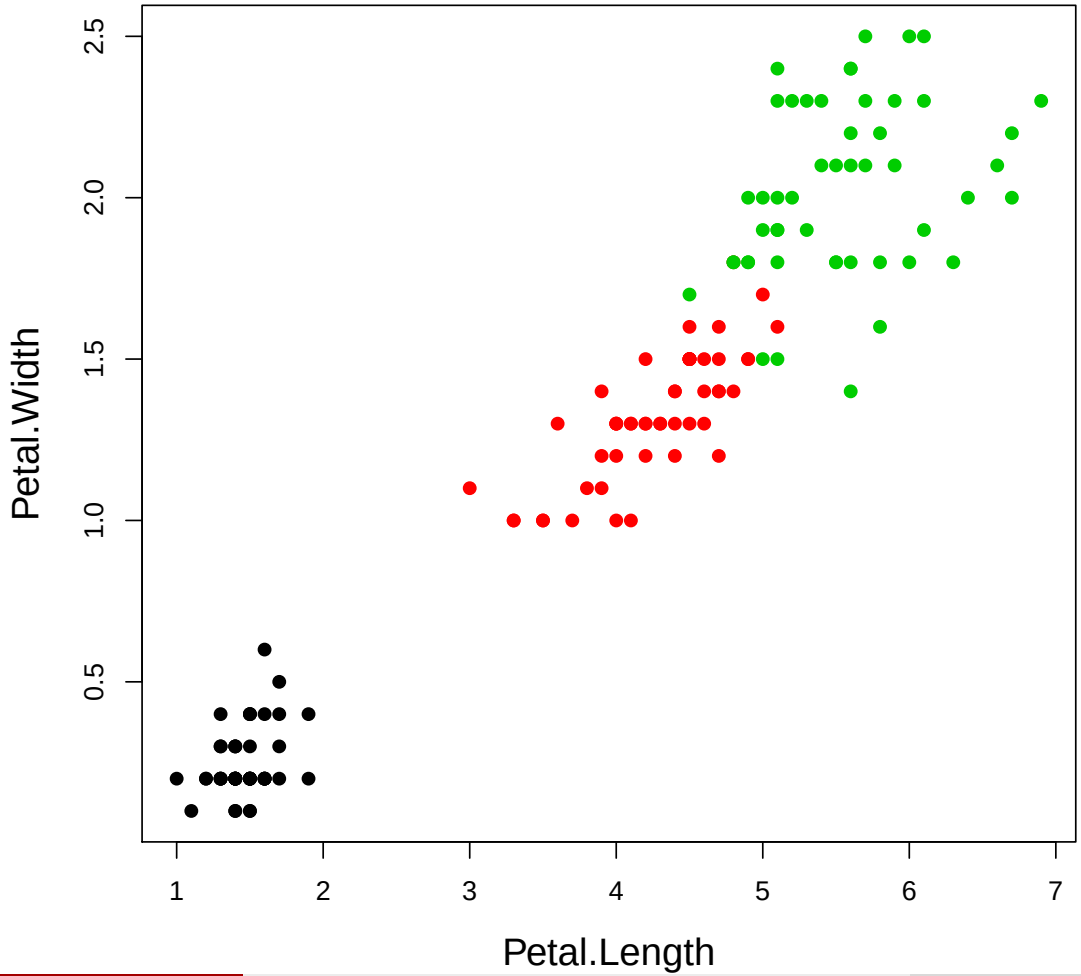
```
library(MASS)
data(iris)
##save class labels
ct <- unclass(iris$Species)
##pairwise plot
pairs(iris[,1:4],col=ct)
```



Iris Dataset

Just focus on two predictor variables.

```
iris.data <- iris[,3:4]
plot(iris.data,col=ct,pch=20,cex=1.5,cex.lab=1.4)
```



Iris Dataset

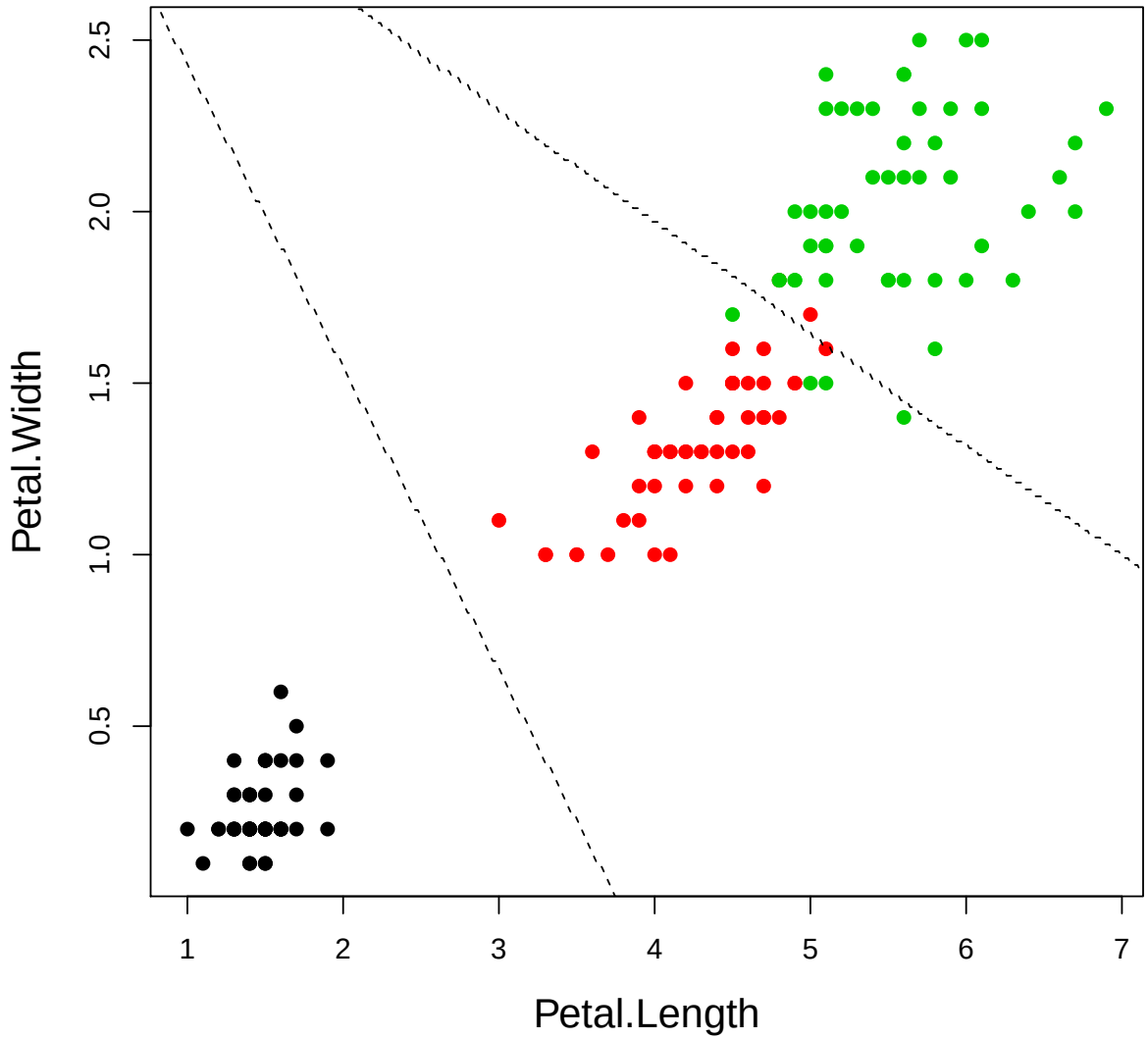
Computing and plotting the LDA boundaries.

```
##fit LDA
iris.lda <- lda(x=iris.data,grouping=ct)

##create a grid for our plotting surface
x <- seq(0,8,0.02)
y <- seq(0,3,0.02)
m <- length(x)
n <- length(y)
z <- as.matrix(expand.grid(x,y),0)

##classes are 1,2 and 3, so set contours at 1.5 and 2.5
iris.ldp <- predict(iris.lda,z)$class
contour(x,y,matrix(iris.ldp,m,n),
        levels=c(1.5,2.5), add=TRUE, d=FALSE, lty=2)
```

Iris Dataset



Generative Learning

- Classifiers we have seen so far are **generative**: we work with a joint distribution $p_{X,Y}(x, y)$ over data vectors and labels.
- A learning algorithm: construct $f : \mathcal{X} \to \mathcal{Y}$ which predicts the label of X.
- Given a loss function L, the risk R of $f(X)$ is

$$R(f) = \mathbb{E}_{p_{X,Y}}[L(Y, f(X))]$$

- For 0/1 loss in classification, Bayes classifier

$$f_{\text{Bayes}}(x) = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, p(Y = k|x) = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, p_{X,Y}(x, k)$$

  has the minimum risk (Bayes risk), but is unknown since $p_{X,Y}$ is unknown.
- Assume a parameteric model for the joint: $p_{X,Y}(x, y) = p_{X,Y}(x, y|\theta)$
- Fit $\hat{\theta} = \operatorname{argmax}_\theta \sum_{i=1}^n \log p(x_i, y_i|\theta)$ and plug in back to Bayes classifier:

$$\hat{f}(x) = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, p(Y = k|x, \theta) = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, p_{X,Y}(x, k|\hat{\theta}).$$

Generative vs Discriminative Learning

- **Generative learning**: find parameters which **explain all the data available**.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(x_i, y_i|\theta)$$

Examples: LDA, QDA, naïve Bayes.

  - Makes use of all the data available.
  - Flexible modelling framework, so can incorporate missing features or unlabeled examples.
  - Stronger modelling assumptions, which may not be realistic (Gaussianity, independence of features).

- **Discriminative learning**: find parameters that aid in **prediction**.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(x_i)) \quad \text{or} \quad \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(y_i|x_i, \theta)$$

Examples: logistic regression, neural nets, support vector machines.

  - Typically performs better on a given task.
  - Weaker modelling assumptions: essentially no model on X, only on $Y|X$.
  - Can overfit more easily.

Logistic regression

- A **discriminative classifier**. Consider binary classification with $\mathcal{Y} = \{-1, +1\}$. Logistic regression uses a parametric model on the conditional $Y|X$, not the joint distribution of (X, Y):

$$p(Y = y|X = x; a, b) = \frac{1}{1 + \exp(-y(a + b^\top x))}.$$

- a, b fitted by minimizing the empirical risk with respect to **log loss**.

Hard vs Soft classification rules

- Consider using LDA for binary classification with $\mathcal{Y} = \{-1, +1\}$. Predictions are based on linear decision boundary:

$$\begin{aligned} \hat{y}_{\text{LDA}}(x) &= \text{sign}\left\{\log \hat{\pi}_{+1} g_{+1}(x|\hat{\mu}_{+1}, \hat{\Sigma}) - \log \hat{\pi}_{-1} g_{-1}(x|\hat{\mu}_{-1}, \hat{\Sigma})\right\} \\ &= \text{sign}\left\{a + b^\top x\right\} \end{aligned}$$

  for a and b depending on fitted parameters $\hat{\theta} = (\hat{\pi}_{-1}, \hat{\pi}_{+1}, \hat{\mu}_{-1}, \hat{\mu}_{+1}, \Sigma)$.
- Quantity $a + b^\top x$ can be viewed as a soft classification rule. Indeed, it is modelling the difference between the log-discriminant functions, or equivalently, the **log-odds ratio**:

$$a + b^\top x = \log \frac{p(Y = +1|X = x; \hat{\theta})}{p(Y = -1|X = x; \hat{\theta})}.$$

- $f(x) = a + b^\top x$ corresponds to the "confidence of predictions" and loss can be measured as a function of this confidence (function of $yf(x)$).

Linearity of log-odds and logistic function

- We can treat a and b as parameters in their own right in the model of the conditional $Y|X$.

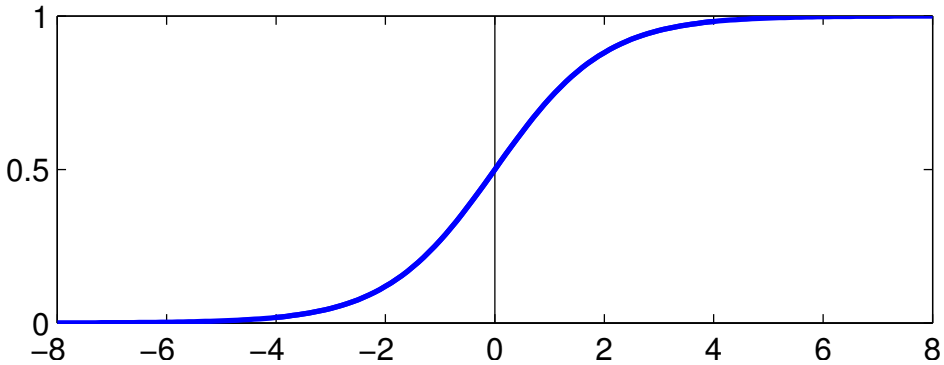
$$\log \frac{p(Y=+1|X=x;a,b)}{p(Y=-1|X=x;a,b)} = a + b^\top x.$$

- Solve explicitly for conditional class probabilities:

$$p(Y=+1|X=x;a,b) = \frac{1}{1+\exp(-(a+b^\top x))} =: s(a+b^\top x)$$

$$p(Y=-1|X=x;a,b) = \frac{1}{1+\exp(+(a+b^\top x))} = s(-a-b^\top x)$$

where $s(z) = 1/(1+\exp(-z))$ is the **logistic function**. [demo]



Fitting the parameters of the hyperplane

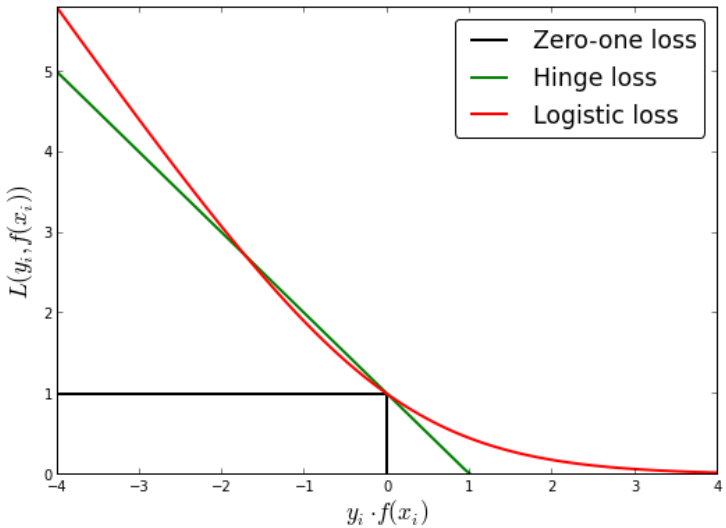
- Consider maximizing the **conditional log likelihood**:

$$\ell(a, b) = \sum_{i=1}^{n} \log p(Y = y_i | X = x_i) = \sum_{i=1}^{n} \log s(y_i(a + b^\top x_i)).$$

- Equivalent to minimizing the empirical risk associated with the **log loss**:

$$\hat{R}_{\log}(f_{a,b}) = \frac{1}{n} \sum_{i=1}^{n} -\log s(y_i(a + b^\top x_i)) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i(a + b^\top x_i)))$$

over all linear soft classification rules $f_{a,b}(x) = a + b^\top x$.



Logistic Regression

- Not possible to find optimal a, b analytically.
- For simplicity, absorb a as an entry in b by appending '1' into x vector.
- Objective function:

$$\hat{R}_{\text{log}} = \frac{1}{n} \sum_{i=1}^{n} -\log s(y_i x_i^\top b)$$

Logistic Function

$$s(-z) = 1 - s(z)$$
$$\nabla_z s(z) = s(z)s(-z)$$
$$\nabla_z \log s(z) = s(-z)$$
$$\nabla_z^2 \log s(z) = -s(z)s(-z)$$

- Differentiate wrt b:

$$\nabla_b \hat{R}_{\text{log}} = \frac{1}{n} \sum_{i=1}^{n} -s(-y_i x_i^\top b) y_i x_i$$

$$\nabla_b^2 \hat{R}_{\text{log}} = \frac{1}{n} \sum_{i=1}^{n} s(y_i x_i^\top b) s(-y_i x_i^\top b) x_i x_i^\top \succeq 0.$$

Logistic Regression

- Second derivative is positive-definite: objective function is **convex** and there is **a single unique global minimum**.
- Many different algorithms can find optimal b, e.g.:
  - Gradient descent:

$$b^{\text{new}} = b + \epsilon \frac{1}{n} \sum_{i=1}^{n} s(-y_i x_i^\top b) y_i x_i$$

  - Stochastic gradient descent:

$$b^{\text{new}} = b + \epsilon_t \frac{1}{|I(t)|} \sum_{i \in I(t)} s(-y_i x_i^\top b) y_i x_i$$

    where $I(t)$ is a subset of the data at iteration t, and $\epsilon_t \to 0$ slowly ($\sum_t \epsilon_t = \infty, \sum_t \epsilon_t^2 < \infty$).
  - Newton-Raphson:

$$b^{\text{new}} = b - (\nabla_b^2 \hat{R}_{\text{log}})^{-1} \nabla_b \hat{R}_{\text{log}}$$

    This is also called **iterative reweighted least squares**.
  - Conjugate gradient, LBFGS and other methods from numerical analysis.

Logistic Regression vs. LDA

- Both have linear decision boundaries and model log-posterior odds as

$$\log \frac{p(Y=+1|X=x)}{p(Y=-1|X=x)} = a + b^\top x$$

- LDA models the marginal density of x as a Gaussian mixture with shared covariance

$$g(x) = \pi_{-1}\mathcal{N}(x;\mu_{-1},\Sigma) + \pi_{+1}\mathcal{N}(x;\mu_{+1},\Sigma)$$

  and fits the parameters $\theta = (\mu_{-1},\mu_{+1},\pi_{-1},\pi_{+1},\Sigma)$ by maximizing joint likelihood $\sum_{i=1}^n p(x_i,y_i|\theta)$. a and b are then determined from θ.
- Logistic regression leaves the marginal density $g(x)$ as an **arbitrary density function**, and fits the parameters a,b by maximizing the conditional likelihood $\sum_{i=1}^n p(y_i|x_i;a,b)$.

Linearly separable data

Assume that the data is linearly separable, i.e. there is a scalar α and a vector β such that $y_i(\alpha + \beta^\top x_i) > 0$, $i = 1, \ldots, n$. Let $c > 0$. The empirical risk for $a = c\alpha$, $b = c\beta$ is

$$\hat{R}_{\mathsf{log}}(f_{a,b}) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-cy_i(\alpha + \beta^\top x_i)))$$

which can be made arbitrarily close to zero as $c \to \infty$, i.e. soft classification rule becomes $\pm\infty$ (overconfidence).

Multi-class logistic regression

The **multi-class/multinomial** logistic regression uses the **softmax** function to model the conditional class probabilities $p\left(Y = k|X = x; \theta\right)$, for K classes $k = 1, \ldots, K$, i.e.,

$$p\left(Y = k|X = x; \theta\right) = \frac{\exp\left(w_k^\top x + b_k\right)}{\sum_{\ell=1}^{K} \exp\left(w_\ell^\top x + b_\ell\right)}.$$

Parameters are $\theta = (b, W)$ where $W = (w_{kj})$ is a $K \times p$ matrix of weights and $b \in \mathbb{R}^K$ is a vector of bias terms.

Logistic Regression: Summary

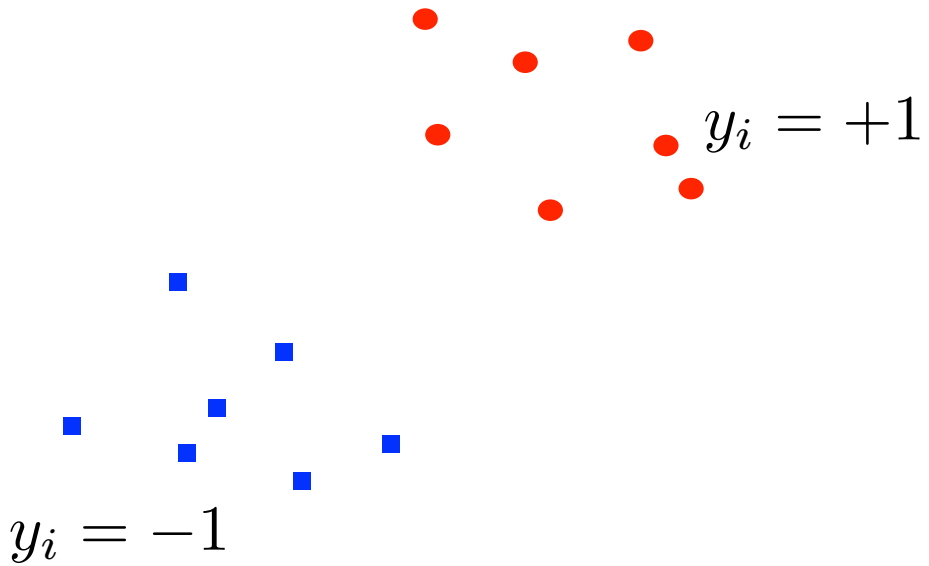
- Makes less modelling assumptions than generative classifiers: often resulting in better prediction accuracy.
- Diverging optimal parameters for linearly separable data: need to **regularise** / pull them towards zero.
- A simple example of a generalised linear model (GLM), for which there is a well established statistical theory:
  - Assessment of fit via deviance and plots,
  - Well founded approaches to removing insignificant features (drop-in deviance test, Wald test).

Reading

- Hastie et al, 4,
- Murphy, 8.

Linearly separable points

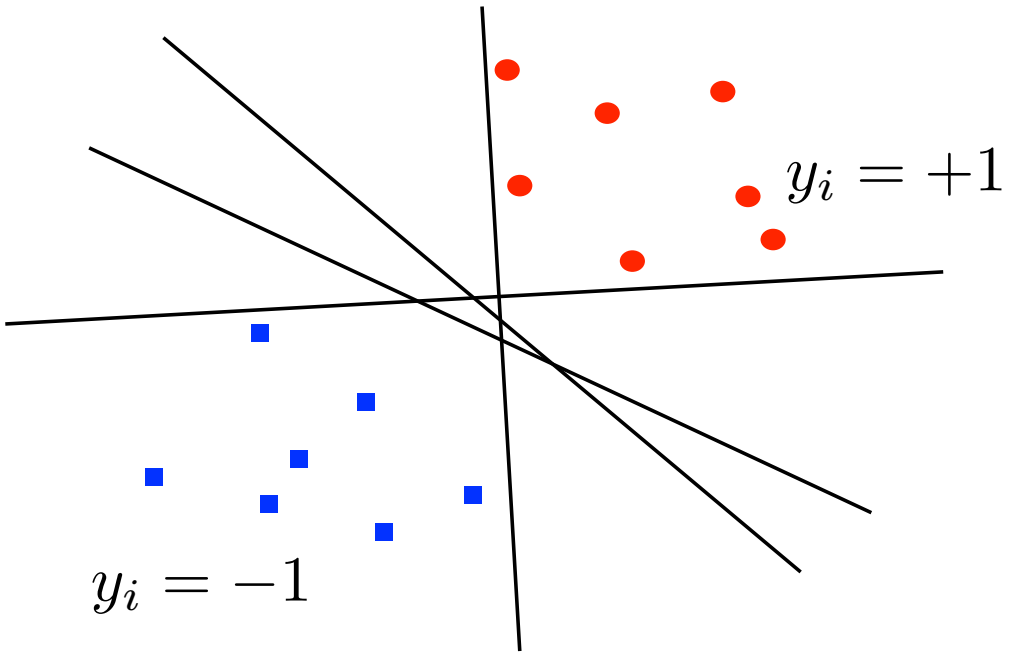
Classify two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error.



Data given by $\{x_i, y_i\}_{i=1}^n$, $x_i \in \mathbb{R}^p$, $y_i \in \{-1, +1\}$

Linearly separable points

Classify two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error.

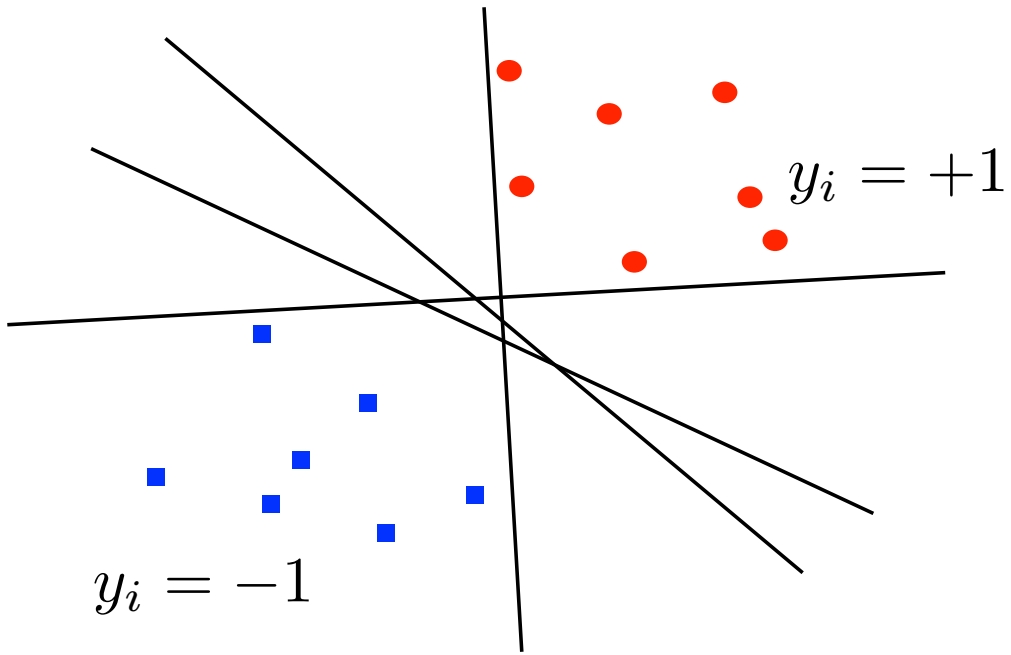


Hyperplane equation $w^\top x + b = 0$. Linear discriminant given by

$$\hat{y}(x) = \text{sign}(w^\top x + b)$$

Linearly separable points

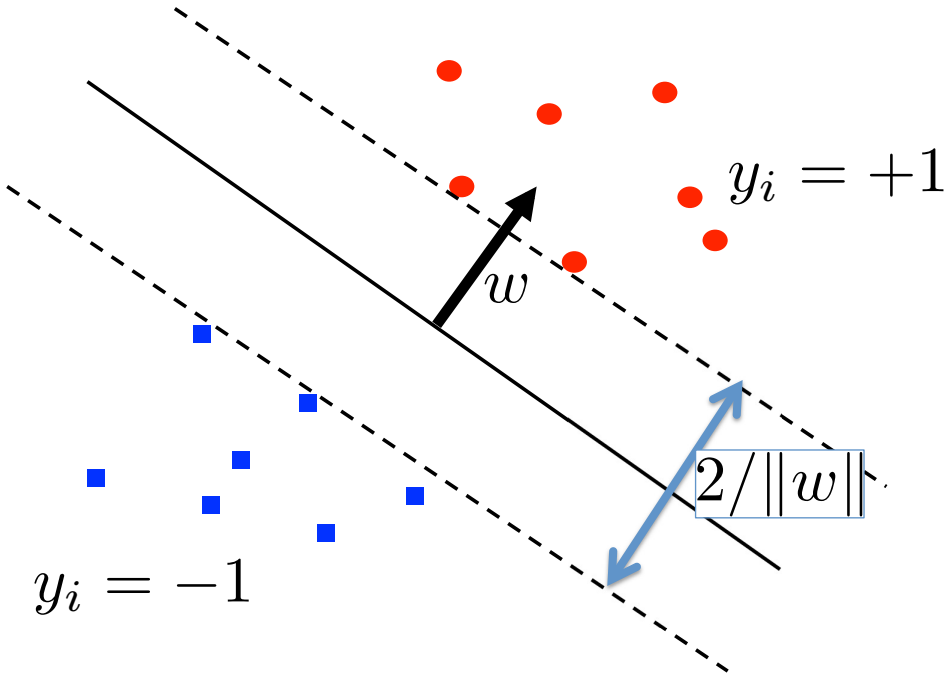
Classify two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error.



For a datapoint close to the decision boundary, a small change leads to a change in classification. Can we make the classifier more robust?

Linearly separable points

Classify two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error.



Smallest distance from each class to the separating hyperplane $w^\top x + b$ is called the **margin**.

Maximum margin classifier, linearly separable case

This problem can be expressed as follows:

$$\max_{w,b} (\text{margin}) = \max_{w,b} \left(\frac{1}{\|w\|} \right)$$

subject to

$$\begin{cases} w^\top x_i + b \geq 1 & i \ : \ y_i = +1, \\ w^\top x_i + b \leq -1 & i \ : \ y_i = -1. \end{cases}$$

The resulting classifier is

$$\hat{y}(x) = \text{sign}(w^\top x + b),$$

We can rewrite to obtain a **quadratic program**:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

subject to

$$y_i(w^\top x_i + b) \geq 1.$$

Maximum margin classifier: with errors allowed

Allow "errors": points within the margin, or even on the wrong side of the decision boundary. Ideally:

$$\min_{w,b} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \mathbb{I}[y_i \left(w^\top x_i + b\right) < 0] \right),$$

where C controls the tradeoff between maximum margin and loss. Replace with **convex upper bound**:

$$\min_{w,b} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} h\left(y_i \left(w^\top x_i + b\right)\right) \right).$$

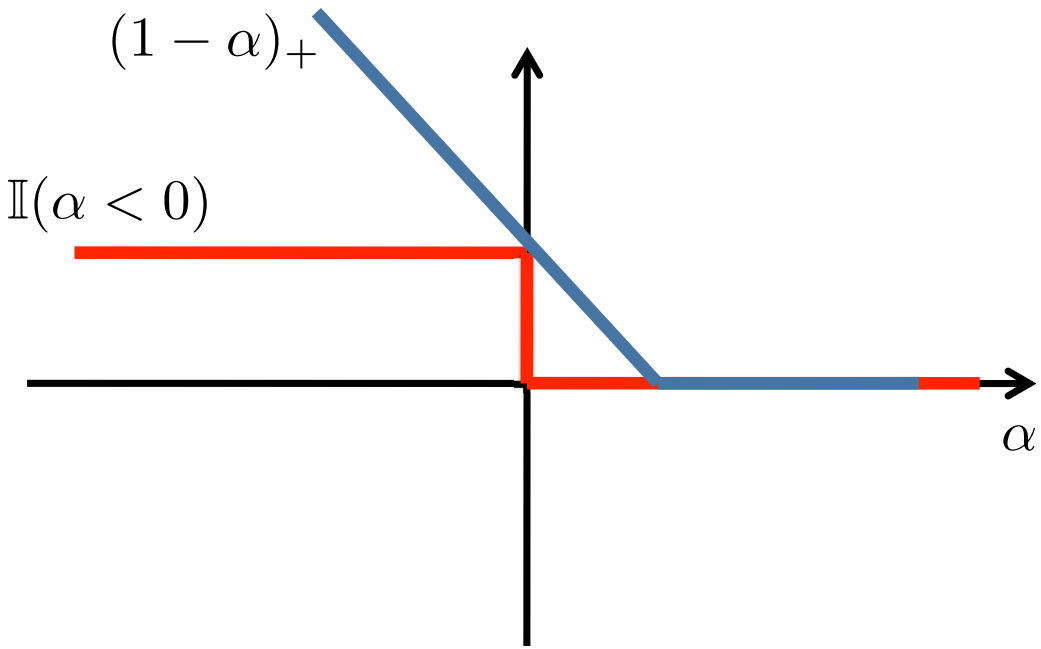
with hinge loss,

$$h(\alpha) = (1-\alpha)_+ = \begin{cases} 1-\alpha, & 1-\alpha > 0 \\ 0, & \text{otherwise.} \end{cases}$$

Hinge loss

Hinge loss:

$$h(\alpha) = (1-\alpha)_+ = \begin{cases} 1-\alpha, & 1-\alpha > 0 \\ 0, & \text{otherwise.} \end{cases}$$



Support vector classification

Substituting in the hinge loss, we get a standard regularised empirical risk minimisation problem - where regularisation naturally arises from the margin penalty.

$$\min_{w,b} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} h\left(y_i \left(w^\top x_i + b\right)\right) \right).$$

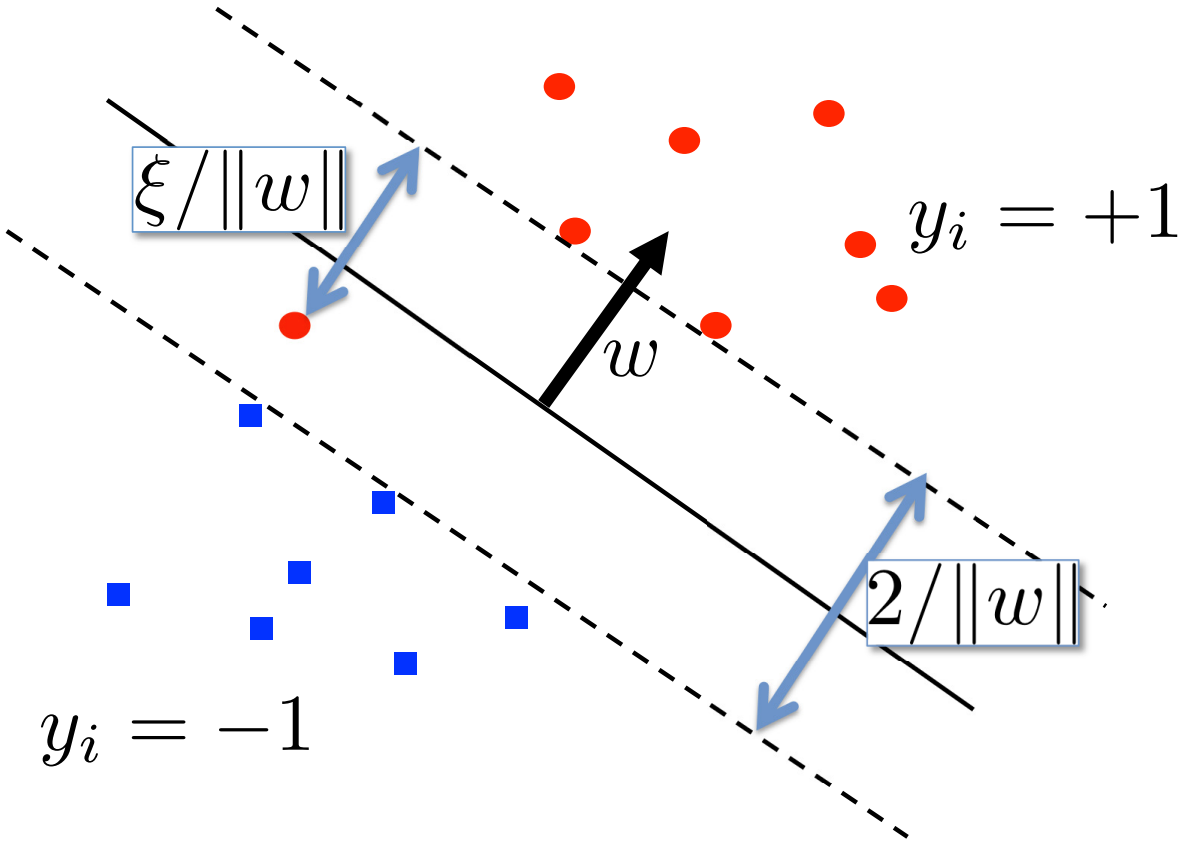
Using substitution $\xi_i = h\left(y_i \left(w^\top x_i + b\right)\right)$, we obtain an equivalent formulation (standard C-SVM):

$$\min_{w,b,\xi} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \right)$$

subject to

$$\xi_i \geq 0 \qquad y_i \left(w^\top x_i + b\right) \geq 1 - \xi_i$$

Support vector classification



Duality

As a convex constrained optimization problem with affine constraints in w, b, ξ, strong duality holds.

$$\begin{aligned} \text{minimize} \quad & f_0(w, b, \xi) := \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & f_i(w, b, \xi) := 1 - \xi_i - y_i \left(w^\top x_i + b\right) \leq 0, \; i = 1, \ldots, n \\ & f_{n+i}(w, b, \xi) := -\xi_i \leq 0, \; i = 1, \ldots, n. \end{aligned}$$

Support vector classification: Lagrangian

The Lagrangian: $L(w, b, \xi, \alpha, \lambda) =$

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - \xi_i - y_i\left(w^\top x_i + b\right)\right) + \sum_{i=1}^{n}\lambda_i(-\xi_i)$$

with dual variable constraints

$$\alpha_i \geq 0, \qquad \lambda_i \geq 0.$$

Minimize wrt the primal variables w, b, and ξ.
Derivative wrt w:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0 \qquad w = \sum_{i=1}^{n}\alpha_i y_i x_i.$$

Derivative wrt b:

$$\frac{\partial L}{\partial b} = \sum_i y_i \alpha_i = 0.$$

Support vector classification: Lagrangian

Derivative wrt ξ_i:

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i = 0 \qquad \alpha_i = C - \lambda_i.$$

Since $\lambda_i \geq 0$,

$$\alpha_i \leq C.$$

Now use complementary slackness:

Non-margin SVs (margin errors): $\alpha_i = C > 0$:

1. We immediately have $y_i \left(w^\top x_i + b\right) = 1 - \xi_i$.
2. Also, from condition $\alpha_i = C - \lambda_i$, we have $\lambda_i = 0$, so $\xi_i \geq 0$

Margin SVs: $0 < \alpha_i < C$:

1. We again have $y_i \left(w^\top x_i + b\right) = 1 - \xi_i$.
2. This time, from $\alpha_i = C - \lambda_i$, we have $\lambda_i > 0$, hence $\xi_i = 0$.

Non-SVs (on the correct side of the margin): $\alpha_i = 0$:

1. From $\alpha_i = C - \lambda_i$, we have $\lambda_i > 0$, hence $\xi_i = 0$.
2. Thus, $y_i \left(w^\top x_i + b\right) \geq 1$

The support vectors

We observe:

1. The solution is sparse: points which are neither on the margin nor "margin errors" have $\alpha_i = 0$
2. The support vectors: only those points on the decision boundary, or which are margin errors, contribute.
3. Influence of the non-margin SVs is bounded, since their weight cannot exceed C.

Support vector classification: dual function

Thus, our goal is to maximize the dual,

$$
\begin{aligned}
g(\alpha, \lambda) &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(w^\top x_i + b\right) - \xi_i\right) \\
&\quad + \sum_{i=1}^{n}\lambda_i(-\xi_i) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j \\
&\quad - b\underbrace{\sum_{i=1}^{n}\alpha_i y_i}_{0} + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}(C - \alpha_i)\xi_i \\
&= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j.
\end{aligned}
$$

Dual C-SVM

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j,$$

subject to the constraints

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

This is a quadratic program. From α, obtain the hyperplane with

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

(follows from complementary slackness in the derivation of the dual). Offset b can be obtained from any of the margin SVs (for which $\alpha_i \in (0, C)$): $1 = y_i \left(w^\top x_i + b \right)$.

Solution depends on data through inner products only

Dual program

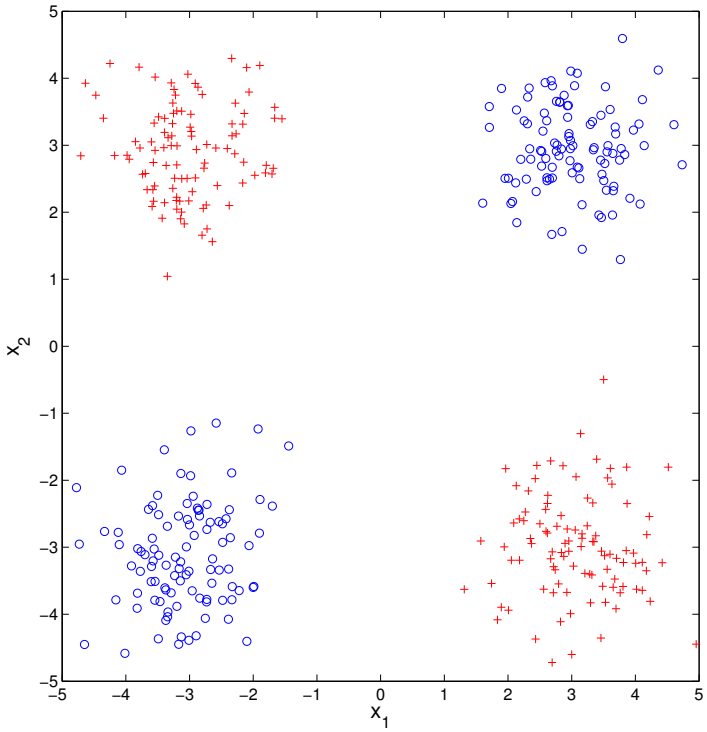
$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j \qquad \text{subject to} \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \preceq \alpha \preceq C \end{cases}$$

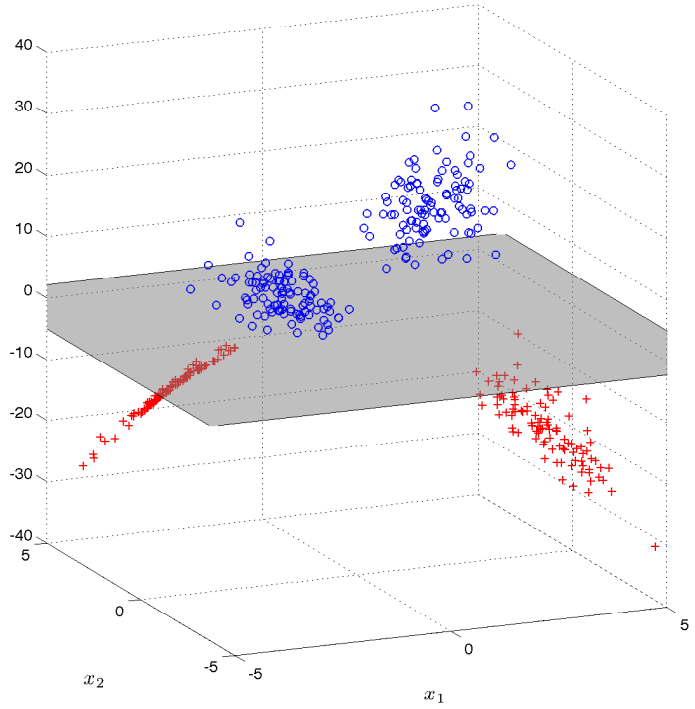
only depends on inputs x_i through their inner products (similarities) with other inputs.
Decision function

$$\hat{y}(x) = \text{sign}(w^\top x + b) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i x_i^\top x + b)$$

also depends only on the similarity of a test point x to the training points x_i.
Thus, we do not need explicit inputs - just their pairwise similarities.
Key property: even if $p > n$, it is still the case that $w \in \text{span}\{x_i : i = 1, \ldots, n\}$ (normal vector of the hyperplane lives in the subspace spanned by the datapoints).

Beyond Linear Classifiers





- No linear classifier separates red from blue.
- Linear separation after mapping to a **higher dimensional feature space**:

$$\mathbb{R}^2 \ni \begin{pmatrix} x^{(1)} & x^{(2)} \end{pmatrix}^\top = x \mapsto \varphi(x) = \begin{pmatrix} x^{(1)} & x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix}^\top \in \mathbb{R}^3$$

Non-Linear SVM

- Consider the dual C-SVM with explicit non-linear transformation $x \mapsto \varphi(x)$:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \varphi(x_i)^\top \varphi(x_j) \quad \text{subject to} \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \preceq \alpha \preceq C \end{cases}$$

- Suppose $p = 2$, and we would like to introduce quadratic non-linearities,

$$\varphi(x) = \left(1, \sqrt{2}x^{(1)}, \sqrt{2}x^{(2)}, \sqrt{2}x^{(1)}x^{(2)}, \left(x^{(1)}\right)^2, \left(x^{(2)}\right)^2\right)^\top.$$

  Then

$$\varphi(x_i)^\top \varphi(x_j) = 1 + 2x_i^{(1)}x_j^{(1)} + 2x_i^{(2)}x_j^{(2)} + 2x_i^{(1)}x_i^{(2)}x_j^{(1)}x_j^{(2)} + \left(x_i^{(1)}\right)^2 \left(x_j^{(1)}\right)^2 + \left(x_i^{(2)}\right)^2 \left(x_j^{(2)}\right)^2 = (1 + x_i^\top x_j)^2$$

- Since only inner products are needed, non-linear transform need not be computed explicitly - inner product between features can be a simple function (**kernel**) of x_i and x_j: $k(x_i, x_j) = \varphi(x_i)^\top \varphi(x_j) = (1 + x_i^\top x_j)^2$
- d-order interactions can be implemented by $k(x_i, x_j) = (1 + x_i^\top x_j)^d$ (**polynomial kernel**). Never need to compute explicit feature expansion of dimension $\binom{p+d}{d}$ where this inner product happens!

Kernel SVM: Kernel trick

- Kernel SVM with $k(x_i, x_j)$. Non-linear transformation $x \mapsto \varphi(x)$ still present, but **implicit** (coordinates of the vector $\varphi(x)$ are never computed).

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad \text{subject to} \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0, \\ 0 \preceq \alpha \preceq C. \end{cases}$$

- Prediction? $\hat{y}(x) = \text{sign}\left(w^\top \varphi(x) + b\right)$, where $w = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i)$.
  - No need to compute w either! Just need

$$w^\top \varphi(x) = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i)^\top \varphi(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x).$$

  - Get offset from any margin support-vector x_j ($\alpha_j \in (0, C)$):

$$b = y_j - w^\top \varphi(x_j) = y_j - \sum_{i=1}^{n} \alpha_i y_i k(x_i, x_j).$$

- Fitted a separating hyperplane in a high-dimensional feature space without ever mapping explicitly to that space.

Reading

- Bishop, 7.1,
- Hastie et al, 12.2.
- http://scikit-learn.org/stable/modules/svm.html

Kernel Methods and Functional Spaces

Kernel: an inner product between feature maps

Definition (kernel)

Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **kernel** if there exists a **Hilbert space**[a] and a map $\varphi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$,

$$k(x, x') := \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}} .$$

[a] Hilbert space is an infinite-dimensional generalization of a classical vector space $\mathbb{R}^d$, in which there is a notion of an inner (dot) product denoted $\langle \cdot, \cdot \rangle$

- Almost no conditions on $\mathcal{X}$ (eg, $\mathcal{X}$ itself need not have an inner product, e.g., documents).
- Think of kernel as a **similarity measure between features**

What are some simple kernels? E.g., for text documents? For images?

- A single kernel can correspond to multiple sets of underlying features.

$$\varphi_1(x) = x \qquad \text{and} \qquad \varphi_2(x) = \begin{pmatrix} x/\sqrt{2} & x/\sqrt{2} \end{pmatrix}^\top$$

Positive semidefinite functions

If we are given a "measure of similarity" with two arguments, $k(x, x')$, how can we determine if it is a valid kernel?

1. Find a feature map?
   - Sometimes not obvious (especially if the feature vector is infinite dimensional)
2. A simpler direct property of the function: positive semidefiniteness.

Positive semidefinite functions

Definition (Positive semidefinite functions)

A symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive semidefinite if $\forall n \geq 1,\ \forall (a_1, \dots a_n) \in \mathbb{R}^n,\ \forall (x_1, \dots, x_n) \in \mathcal{X}^n$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \kappa(x_i, x_j) \geq 0.$$

- Kernel $k(x, y) := \langle \varphi(x), \varphi(y) \rangle_{\mathcal{H}}$ for a Hilbert space $\mathcal{H}$ is positive semidefinite.

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(x_i, x_j) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle a_i \varphi(x_i), a_j \varphi(x_j) \rangle_{\mathcal{H}} \\ &= \left\| \sum_{i=1}^{n} a_i \varphi(x_i) \right\|_{\mathcal{H}}^2 \geq 0. \end{aligned}$$

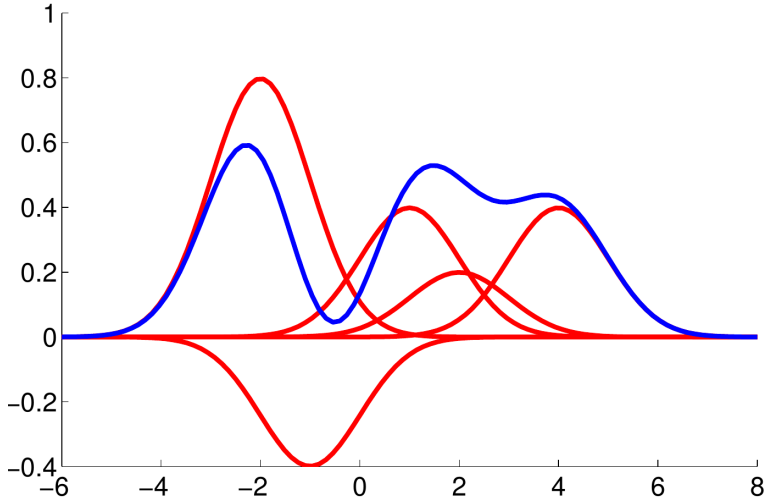
Positive semidefinite functions are kernels

Moore-Aronszajn Theorem

Every positive semidefinite function is a kernel for some Hilbert space $\mathcal{H}$.

- There is a special $\mathcal{H}$ called **Reproducing kernel Hilbert space - RKHS**.

Gaussian RBF kernel $k(x, x') = \exp\left(-\frac{1}{2\gamma^2}\|x - x'\|^2\right)$ has an infinite-dimensional $\mathcal{H}$ with elements $h(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x)$
(recall that $w^\top \varphi(x)$ in SVM has exactly this form!).



Reproducing kernel

Definition (Reproducing kernel)

Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ defined on a non-empty set $\mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called **a reproducing kernel** of $\mathcal{H}$ if it satisfies

- $\forall x \in \mathcal{X}, \quad k_x = k(\cdot, x) \in \mathcal{H}$,
- $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \quad \langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x)$ (the reproducing property).

In particular, for any $x, y \in \mathcal{X}$, $k(x, y) = \langle k(\cdot, y), k(\cdot, x) \rangle_{\mathcal{H}} = \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}}$.

Can forget all about $\varphi(x)$ and just treat $k(\cdot, x)$ as a feature of x (it is a perfectly valid Hilbert-space valued feature)!

Back to SVMs

Maximum margin classifier in RKHS: Looking for a decision function of form sign$(f(x))$ where $f \in \mathcal{H}_k$. Because we are in an RKHS, $f(x) = \langle f, k(\cdot, x)\rangle_{\mathcal{H}_k}$.

$$\min_{f \in \mathcal{H}_k} \left(\frac{1}{2}\|f\|^2_{\mathcal{H}_k} + C \sum_{i=1}^{n} \left(1 - y_i \langle f, k(\cdot, x_i)\rangle_{\mathcal{H}_k}\right)_+ \right)$$

for the RKHS $\mathcal{H}$ with kernel $k(x, x')$.
Maximizing the margin equivalent to minimizing $\|f\|^2_{\mathcal{H}}$: for many RKHSs a smoothness constraint on function f.
Why can we solve this infinite-dimensional optimization problem? Because we know that $f \in$ span $\{k(\cdot, x_i) : i = 1, \ldots, n\}$ – Representer Theorem.

Representer theorem

Standard supervised learning setup: we are given a set of paired observations $(x_1, y_1), \ldots (x_n, y_n)$.
Goal: find the function f^* in the RKHS $\mathcal{H}$ which solves the regularized empirical risk minimization problem.

$$\min_{f \in \mathcal{H}} \hat{R}(f) + \Omega\left(\|f\|_{\mathcal{H}}^2\right),$$

where empirical risk is

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)),$$

and Ω is a non-decreasing function and L is any loss function.

- Classification: $L(y, f(x), x) = (1 - yf(x))_+$ or a logistic loss $L(y, f(x), x) = \log\left(1 + \exp(-yf(x)\right)$.
- Regression: $L(y, f(x)) = (y - f(x))^2$ or $L(y, f(x)) = |y - f(x)|$.

Representer theorem

Theorem (Representer Theorem)

There is a solution to

$$\min_{f \in \mathcal{H}} \; \hat{R}(f) + \Omega\left(\|f\|_{\mathcal{H}}^2\right)$$

that takes the form

$$f^* = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i).$$

If Ω is strictly increasing, all solutions have this form.

Representer theorem: proof

Proof: Denote f_s projection of f onto the subspace

$$\operatorname{span}\{k(\cdot, x_i) : \ i = 1, \ldots, n\}$$

such that

$$f = f_s + f_\perp,$$

where $f_s = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i)$ and $f_\perp$ is orthogonal to $\operatorname{span}\{k(\cdot, x_i) : \ i = 1, \ldots, n\}$.

Regularizer:

$$\|f\|_{\mathcal{H}}^2 = \|f_s\|_{\mathcal{H}}^2 + \|f_\perp\|_{\mathcal{H}}^2 \geq \|f_s\|_{\mathcal{H}}^2 ,$$

then

$$\Omega\left(\|f\|_{\mathcal{H}}^2\right) \geq \Omega\left(\|f_s\|_{\mathcal{H}}^2\right).$$

Representer theorem: proof

Proof (cont.): Individual terms $f(x_i)$ in the loss:

$$f(x_i) = \langle f, k(\cdot, x_i)\rangle_{\mathcal{H}} = \langle f_s + f_\perp, k(\cdot, x_i)\rangle_{\mathcal{H}} = \langle f_s, k(\cdot, x_i)\rangle_{\mathcal{H}}\,,$$

so

$$L(y_i, f(x_i)) = L(y_i, f_s(x_i)),\ \forall i \in \{1, \ldots, n\} \quad \Rightarrow \quad \hat{R}(f) = \hat{R}(f_s).$$

Hence

- The empirical risk only depends on the components of f lying in the subspace spanned by canonical features.
- Regularizer $\Omega(\ldots)$ is minimized when $f = f_s$.
- If Ω is strictly non-decreasing, then $\|f_\perp\|_{\mathcal{H}} = 0$ is required at the minimum.

Regularised Least Squares

We are given n training points $\{x_i\}_{i=1}^n$ in $\mathbb{R}^p$: Define some $\lambda > 0$. Our goal is:

$$\begin{aligned} w^* &= \arg\min_{w \in \mathbb{R}^p} \left(\sum_{i=1}^n (y_i - x_i^\top w)^2 + \lambda \|w\|^2 \right) \\ &= \arg\min_{w \in \mathbb{R}^p} \left(\|\mathbf{y} - \mathbf{X}w\|^2 + \lambda \|w\|^2 \right), \end{aligned}$$

Solution is:

$$w^* = \left(\mathbf{X}^\top \mathbf{X} + \lambda I\right)^{-1} \mathbf{X}^\top \mathbf{y},$$

which is the standard regularised least squares solution.

Kernel ridge regression

Use features $\phi(x_i)$ in the place of x_i:

$$w^* \quad = \quad \arg\min_{w \in \mathcal{H}} \left(\sum_{i=1}^{n} \left(y_i - \langle w, \phi(x_i) \rangle_{\mathcal{H}} \right)^2 + \lambda \|w\|_{\mathcal{H}}^2 \right).$$

E.g. for finite dimensional feature spaces,

$$\phi_p(x) = \begin{bmatrix} x \\ x^2 \\ \vdots \\ x^\ell \end{bmatrix} \qquad \phi_s(x) = \begin{bmatrix} \sin(x) \\ \cos(x) \\ \sin(2x) \\ \vdots \\ \cos\left(\frac{\ell}{2} x\right) \end{bmatrix}$$

In finite dimensions, w is a vector of length ℓ giving weight to each of these features so that learned function is $f_w(x) = w^\top \phi(x)$. Feature vectors can also have **infinite** length.

Kernel ridge regression

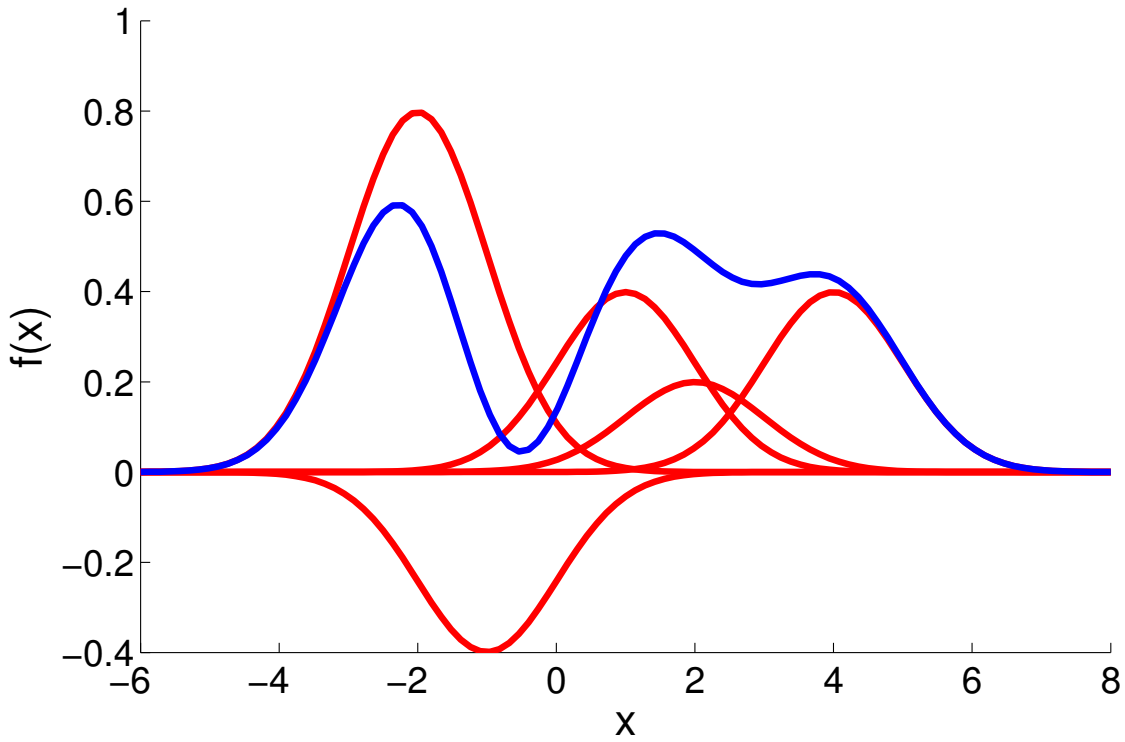
Recall that feature maps ϕ and feature spaces $\mathcal{H}$ are not unique, but RKHS $\mathcal{H}_k$ is. Thus, we can identify w with the function f_w (there is an isometry between w and f_w: $\|w\|_{\mathcal{H}} = \|f_w\|_{\mathcal{H}_k}$ regardless of the choice of the feature space $\mathcal{H}$) and write

$$\begin{aligned} f^* &= \arg\min_{f \in \mathcal{H}_k} \left(\sum_{i=1}^{n} \left(y_i - \langle f, k(\cdot, x_i)\rangle_{\mathcal{H}}\right)^2 + \lambda \|f\|_{\mathcal{H}_k}^2 \right) \\ &= \arg\min_{f \in \mathcal{H}_k} \left(\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \|f\|_{\mathcal{H}_k}^2 \right). \end{aligned}$$

Kernel ridge regression

Recall the representer theorem: f is a linear combination of feature space mappings of data points

$$w = \sum_{i=1}^{n} \alpha_i \phi(x_i), \quad f_w = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot).$$



Kernel ridge regression

Recall the representer theorem: f is a linear combination of feature space mappings of data points

$$f = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i).$$

Then

$$\begin{aligned} \sum_{i=1}^{n} \left(y_i - \langle f, k(\cdot, x_i)\rangle_{\mathcal{H}_k}\right)^2 + \lambda \|f\|_{\mathcal{H}_k}^2 &= \|\mathbf{y} - \mathbf{K}\alpha\|^2 + \lambda \alpha^\top \mathbf{K} \alpha \\ &= \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top K\alpha + \alpha^\top \left(\mathbf{K}^2 + \lambda \mathbf{K}\right) \alpha \end{aligned}$$

Differentiating wrt α and setting this to zero, we get

$$\alpha^* = (\mathbf{K} + \lambda I_n)^{-1} y.$$

Recall: $\frac{\partial \alpha^\top U \alpha}{\partial \alpha} = (U + U^\top)\alpha, \qquad \frac{\partial v^\top \alpha}{\partial \alpha} = \frac{\partial \alpha^\top v}{\partial \alpha} = v$

Parameter selection for KRR

Given the objective

$$f^* \quad = \quad \arg\min_{f \in \mathcal{H}_k} \left(\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \|f\|^2_{\mathcal{H}_k} \right).$$

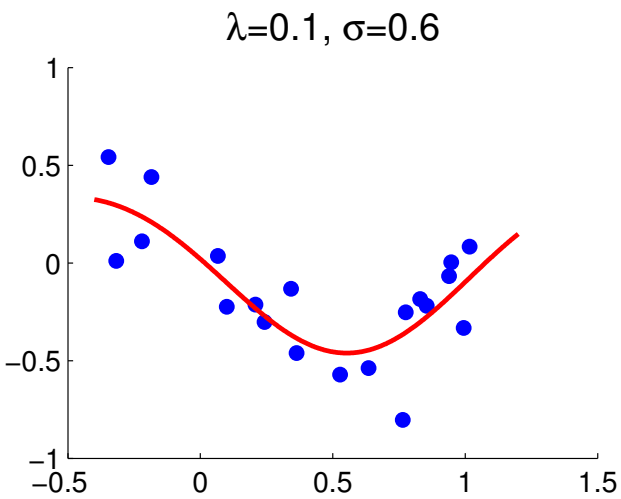
How do we choose

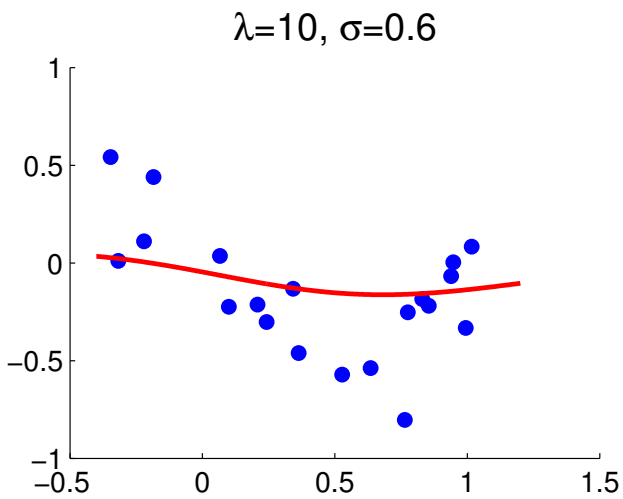
- The regularization parameter λ?
- The kernel parameter: for Gaussian kernel, σ in

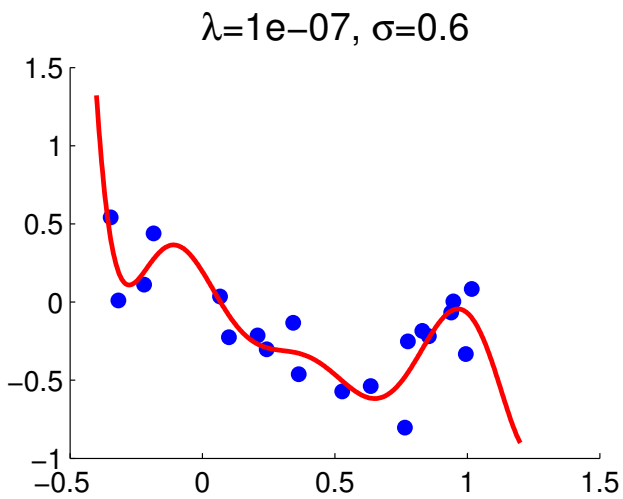
$$k(x, y) = \exp\left(\frac{-\|x - y\|^2}{\sigma} \right).$$

Beware: Gaussian kernel has many different parametrisations in the literature and software packages!
Typically use cross-validation.

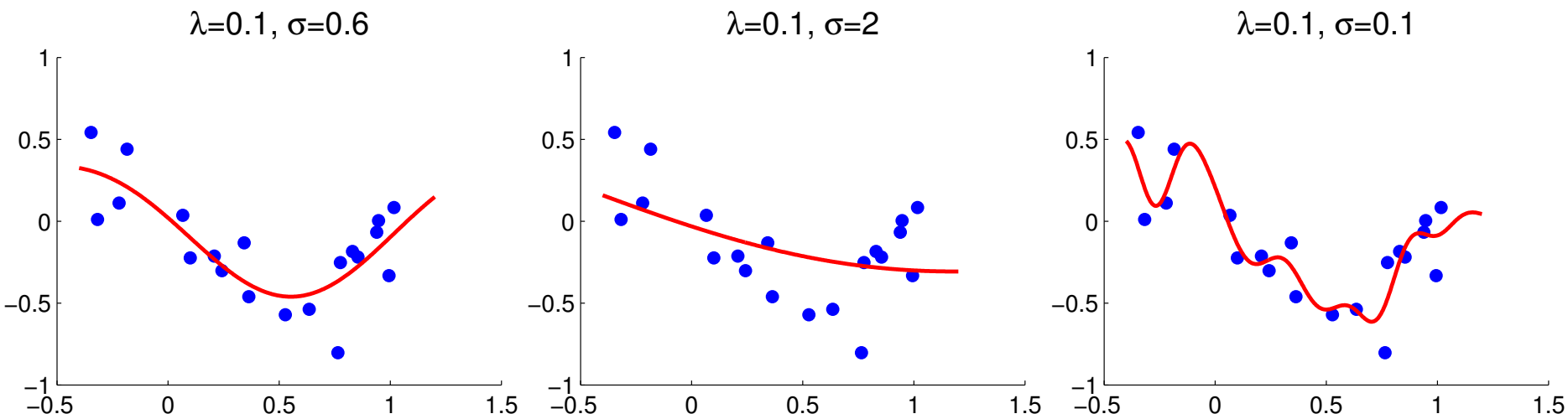
Choice of λ







Choice of σ



Examples of kernels

- **Linear**: $k(x, x') = x^\top x'$.
- **Polynomial**: $k(x, x') = (c + x^\top x')^m$, $c \in \mathbb{R}$, $m \in \mathbb{N}$.
- **Exponential**: $k(x, x') = \exp(\frac{x^\top x'}{\gamma})$, $\gamma > 0$.
- **Gaussian RBF**: $k(x, x') = \exp\left(-\frac{1}{2\gamma^2}\|x - x'\|^2\right)$, $\gamma > 0$.
- **Laplacian**: $k(x, x') = \exp\left(-\frac{1}{2\gamma^2}\|x - x'\|\right)$, $\gamma > 0$.
- **Rational quadratic**: $k(x, x') = \left(1 + \frac{\|x-x'\|^2}{2\alpha\gamma^2}\right)^{-\alpha}$, $\alpha, \gamma > 0$.
- **Brownian covariance**: $k(x, x') = \frac{1}{2}\left(\|x\|^\gamma + \|x'\|^\gamma - \|x - x'\|^\gamma\right)$, $\gamma \in [0, 2]$.

New kernels from old: sums, transformations

The great majority of useful kernels are built from simpler kernels.

Lemma (Sums of kernels are kernels)

Given $\alpha > 0$ and k, k_1 and k_2 all kernels on $\mathcal{X}$, then αk and $k_1 + k_2$ are kernels on $\mathcal{X}$.

To prove this, just check inner product definition (features get scaled with $\sqrt{\alpha}$ or concatenated). A difference of kernels need not be a kernel (**why?**)

Lemma (Space transformation)

Let $\mathcal{X}$ and $\widetilde{\mathcal{X}}$ be sets, and consider any map $s : \mathcal{X} \to \widetilde{\mathcal{X}}$. Let $\tilde{k}$ be a kernel on $\widetilde{\mathcal{X}}$. Then $k(x, x') = \tilde{k}(s(x), s(x'))$ is a kernel on $\mathcal{X}$.

Proof: if $\tilde{\varphi}$ is a feature map for $\tilde{k}$, then $\varphi = \tilde{\varphi} \circ s$ is a feature map for k.

New kernels from old: products

Lemma (Products of kernels are kernels)

Given k_1 on $\mathcal{X}_1$ and k_2 on $\mathcal{X}_2$, then $k_1 \times k_2$ is a kernel on $\mathcal{X}_1 \times \mathcal{X}_2$.

Proof.

Sketch for finite-dimensional spaces only. Assume $\mathcal{H}_1$ corresponding to k_1 is $\mathbb{R}^m$, and $\mathcal{H}_2$ corresponding to k_2 is $\mathbb{R}^n$. Define:

- $k_1 := u^\top v$ for $u, v \in \mathbb{R}^m$ (e.g.: kernel between two images)
- $k_2 := p^\top q$ for $p, q \in \mathbb{R}^n$ (e.g.: kernel between two captions)

Is the following a kernel?

$$K\left[(u,p);(v,q)\right] = k_1 \times k_2$$

(e.g. kernel between one image-caption pair and another)

□

New kernels from old: products

Proof.

(continued)

$$\begin{aligned} k_1 k_2 &= \left(u^\top v\right)\left(q^\top p\right) \\ &= \operatorname{trace}(u^\top v q^\top p) \\ &= \operatorname{trace}(p u^\top v q^\top) \\ &= \langle A, B \rangle , \end{aligned}$$

where $A := pu^\top$, $B := qv^\top$ (features of image-caption pairs) Thus $k_1 k_2$ is a valid kernel, since inner product between $A, B \in \mathbb{R}^{m \times n}$ is

$$\langle A, B \rangle = \operatorname{trace}(AB^\top).$$

□

Kernel Methods – Discussion

- Kernel methods allows for very flexible and powerful machine learning models.
- **Nonparametric** method: parameter space (e.g., normal vector w in SVM) can be infinite-dimensional
- Kernels can be defined over more complex structures than vectors, e.g. graphs, strings, images, bags of instances, probability distributions.
- In naïve implementation, computational cost is at least quadratic in the number of observations, often $O(n^3)$ computation and $O(n^2)$ memory, but there are various approximations with good scaling up properties.
- Further reading:
  - Schölkopf and Smola, Learning with Kernels, 2001.
  - Rasmussen and Williams, Gaussian Processes for Machine Learning, 2006.
  - Steinwart and Christmann, Support Vector Machines, 2008.
  - Bishop, Pattern Recognition and Machine Learning, Chapter 6.