SC4/SM8 Advanced Topics in Statistical Machine Learning

Chapter 1: Review of Fundamentals

Dino Sejdinovic
Department of Statistics
Oxford

Slides and other materials available at:
http://www.stats.ox.ac.uk/~sejdinov/atsml19/

Course Structure

Who is the course for?

- MMath Part C & OMMS & MSc in Statistical Science

Lectures

- Tuesdays 3pm, LG.01.
- Thursdays 4pm, LG.01.

MSc

- 4 problem sheets: classes Fri 10am weeks 3,5,7,TT1, LG.01.

Part C/OMMS

- 4 problem sheets, **solutions due Fri noon in weeks 2,4,6 (sheets 1-3)**.
- 3 sets of classes on Tue mornings in weeks 3,5,7,TT1.
- Sign up details (via WebLearn) on departmental website. Check minerva class lists or course website for class times, locations, and class tutor contact details.

Background

- SB2.2/SM4 is helpful, but not a prerequisite. Ch.1 provides detailed background notes on material needed.

Lecture notes and slides will be available at the course website:

`http://www.stats.ox.ac.uk/~sejdinov/atsml19/`

Topics

1. Review of fundamentals: PCA, clustering, empirical risk minimization, regularization
2. Support vector machines
3. Kernel methods and Reproducing kernel Hilbert spaces
4. Graph Laplacians: Spectral clustering and manifold regularization
5. Latent variable models and EM algorithm
6. Collaborative filtering and matrix factorization
7. Bayesian learning
8. Variational methods, topic modelling
9. Gaussian processes
10. Bayesian optimization

What is Machine Learning?

Arthur Samuel, 1959

Field of study that gives computers the ability to **learn** without being explicitly programmed.

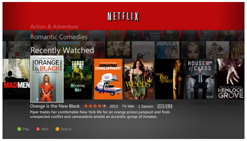
Tom Mitchell, 1997

Any computer program that **improves its performance** at some task **through experience**.

Kevin Murphy, 2012

To develop methods that can **automatically** detect **patterns in data**, and then to use the uncovered patterns to **predict** future data or other outcomes of interest.

What is Machine Learning?



Orange is the New Black ★★★★★ 2013 TV-MA 1 Season HD 5.1
Piper trades her comfortable New York life for an orange prison jumpsuit and finds unexpected conflict and camaraderie amidst an eccentric group of inmates.

recommender systems

machine translation

self-driving cars

image recognition

DQN Atari games

AlphaGo

Types of Machine Learning

Supervised learning

- Data contains “labels”: every example is an input-output pair
- classification, regression
- Goal: **prediction on new examples**

Unsupervised learning

- Extract key features of the “unlabelled” data
- clustering, signal separation, density estimation
- Goal: **representation, hypothesis generation, visualization**

Types of Machine Learning

Semi-supervised Learning

A database of examples, only a small subset of which are labelled.

Multi-task Learning

A database of examples, each of which has multiple labels corresponding to different prediction tasks.

Reinforcement Learning

An agent acting in an environment, given rewards for performing appropriate actions, learns to maximize their reward.

Unsupervised learning basics

Unsupervised Learning

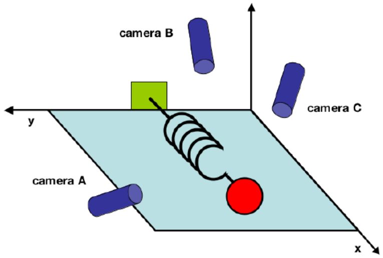
Goals:

- Find the variables that summarise the data / capture relevant information.
- Discover informative ways to visualise the data.
- Discover the subgroups among the observations.

It is often much easier to obtain unlabeled data than labeled data!

Dimensionality reduction

- deceptively many variables to measure, many of them redundant / correlated to each other (large p)
- often, there is a simple but unknown underlying relationship hiding
- example: ball on a frictionless spring recorded by three different cameras
  - our imperfect measurements obfuscate the true underlying dynamics
  - are our coordinates meaningful or do they simply reflect the method of data gathering?



J. Shlens, A Tutorial on Principal Component Analysis, 2005

Principal Components Analysis (PCA)

- PCA considers interesting directions to be those with greatest **variance**.
- A **linear** dimensionality reduction technique: looks for a **new basis** to represent a noisy dataset.
- Workhorse for many different types of data analysis (often used for data preprocessing before supervised techniques are applied).
- Often the first thing to run on high-dimensional data.

Data Matrix notation

Notation

- Data consists of p variables (features/attributes/dimensions) on n examples (items/observations).
- $\mathbf{X} = (x_{ij})$ is a $n \times p$-matrix with $x_{ij} :=$ the j-th variable for the i-th example

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{np} \end{bmatrix}.$$

- Denote the i-th data item by $x_i \in \mathbb{R}^p$ (we will treat it as a column vector: it is the transpose of the i-th row of $\mathbf{X}$).
- Assume $x_1, \dots, x_n$ are **independently and identically distributed** samples of a **random vector** X over $\mathbb{R}^p$. The j-th dimension of X will be denoted $X^{(j)}$.

PCA

PCA

Find an orthogonal basis $\{v_1, v_2, \dots, v_p\}$ for the data space such that:

- The first principal component (PC) v_1 is the **direction of greatest variance** of data.
- The j-th PC v_j is the **direction orthogonal to** $v_1, v_2, \dots, v_{j-1}$ **of greatest variance**, for $j = 2, \dots, p$.

- Eigendecomposition of the sample covariance matrix $S = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^\top$ (data is assumed centred).

$$S = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X} = V\Lambda V^\top.$$

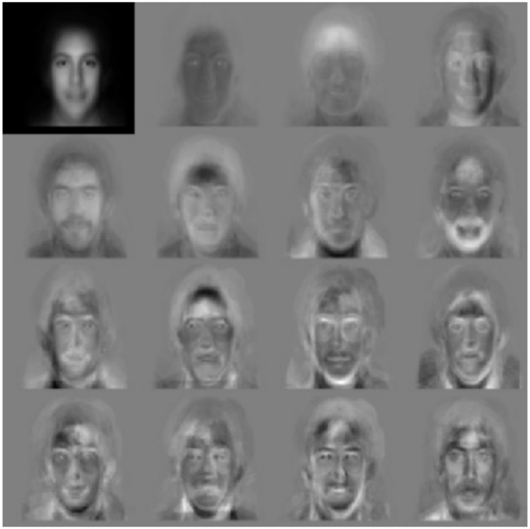
  - Λ is a diagonal matrix with eigenvalues (variances along each principal component) $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$
  - V (**loadings matrix**) is a $p \times p$ orthogonal matrix whose columns are the p eigenvectors of S, i.e. the principal components $v_1, \dots, v_p$
- Dimensionality reduction by projecting $x_i \in \mathbb{R}^p$ onto first k principal components, to obtain **scores matrix**:

$$z_i = \left[v_1^\top x_i, \dots, v_k^\top x_i\right]^\top \in \mathbb{R}^k; \qquad \mathbf{Z} = \mathbf{X}V.$$

Properties of the Principal Components

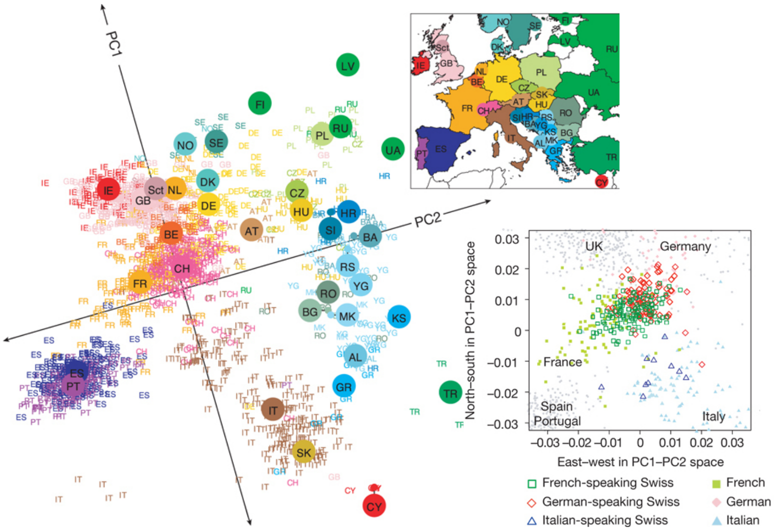
- Derived scalar variable (projection to the j-th principal component) $Z^{(j)} = v_j^\top X$ has sample variance λ_j, for $j = 1, \ldots, p$
- S is a real symmetric matrix, so eigenvectors (principal components) are orthogonal.
- Projections to principal components are **uncorrelated**: $\widehat{\text{Cov}}(Z^{(i)}, Z^{(j)}) \approx v_i^\top S v_j = \lambda_j v_i^\top v_j = 0$, for $i \neq j$.
- The **total sample variance** is given by $\text{Tr}(S) = \sum_{i=1}^p S_{ii} = \lambda_1 + \ldots + \lambda_p$, so the **proportion of total variance explained** by the j^{th} PC is $\frac{\lambda_j}{\lambda_1+\lambda_2+\ldots+\lambda_p}$

PCA on Face Images: Eigenfaces



Turk and Pentland, CVPR 1995

PCA on European Genetic Variation



Genes mirror geography within Europe, Nature 2008

Singular Value Decomposition (SVD)

SVD

Any real-valued $n \times p$ matrix $\mathbf{X}$ can be written as $\mathbf{X} = UDV^\top$ where

- U is an $n \times n$ orthogonal matrix: $UU^\top = U^\top U = I_n$
- D is a $n \times p$ matrix with decreasing **non-negative** elements on the diagonal (the singular values) and zero off-diagonal elements.
- V is a $p \times p$ orthogonal matrix: $VV^\top = V^\top V = I_p$

- SVD **always** exists, even for non-square matrices.
- Fast and numerically stable algorithms for SVD are available in most packages, e.g. the relevant R command is `svd`.

SVD and PCA

- Let $\mathbf{X} = UDV^\top$ be the SVD of the $n \times p$ data matrix $\mathbf{X}$.
- Note that

$$(n-1)S = \mathbf{X}^\top \mathbf{X} = (UDV^\top)^\top (UDV^\top) = VD^\top U^\top UDV^\top = VD^\top DV^\top,$$

  using orthogonality ($U^\top U = I_n$) of U.
- The eigenvalues of S are thus the diagonal entries of $\Lambda = \frac{1}{n-1} D^\top D$.
- We also have

$$\mathbf{X}\mathbf{X}^\top = (UDV^\top)(UDV^\top)^\top = UDV^\top VD^\top U^\top = UDD^\top U^\top,$$

  using orthogonality ($V^\top V = I_p$) of V.

Gram matrix

$\mathbf{K} = \mathbf{X}\mathbf{X}^\top$, $\mathbf{K}_{ij} = x_i^\top x_j$ is called the Gram matrix of dataset $\mathbf{X}$.
$\mathbf{K}$ and $(n-1)S = \mathbf{X}^\top \mathbf{X}$ have the same nonzero eigenvalues, equal to the non-zero squared singular values of $\mathbf{X}$.

PCA projections from Gram matrix

If we consider projections to **all principal components**, the transformed data matrix is

$$\mathbf{Z} = \mathbf{X}V = UDV^\top V = UD, \tag{1}$$

If $p \leq n$ this means

$$z_i = \left[U_{i1}D_{11}, \ldots, U_{ip}D_{pp}\right]^\top, \tag{2}$$

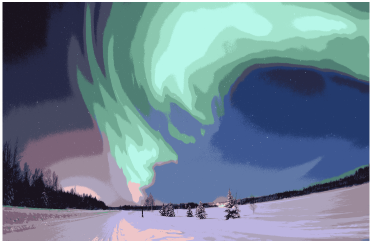
and if $p > n$ only the first n projections are defined (sample covariance will have rank at most n):

$$z_i = \left[U_{i1}D_{11}, \ldots, U_{in}D_{nn}, 0, \ldots, 0\right]^\top. \tag{3}$$

Thus, $\mathbf{Z}$ can be obtained from the eigendecomposition of Gram matrix $\mathbf{K}$. When $p \gg n$, eigendecomposition of $\mathbf{K}$ requires much less computation, $O(n^3)$, than the eigendecomposition of the covariance matrix, $O(p^3)$, so is the preferred method for PCA in that case.

Clustering

- Many datasets consist of multiple heterogeneous subsets.
- **Cluster analysis**: Given an unlabelled data, want algorithms that automatically group the datapoints into coherent subsets/clusters. Examples:
  - market segmentation of shoppers based on browsing and purchase histories
  - different types of breast cancer based on the gene expression measurements
  - discovering communities in social networks
  - image segmentation



Types of Clustering

- **Model-free** clustering:
  - Defined by **similarity/dissimilarity** among instances within clusters.
- **Model-based** clustering:
  - Each cluster is described using a probability model.

Model-free clustering

- notion of similarity/dissimilarity between data items is central: many ways to define and the choice will depend on the dataset being analyzed and dictated by domain specific knowledge
- most common approach is **partition-based** clustering: one divides n data items into K clusters $C_1, \dots, C_K$ where for all $k, k' \in \{1, \dots, K\}$,

$$C_k \subset \{1, \dots, n\}, \qquad C_k \cap C_{k'} = \emptyset \ \ \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \dots, n\}.$$

- Intuitively, clustering aims to group similar items together and to place separate dissimilar items into different groups
- two objectives can contradict each other (similarity is not a transitive relation, while being in the same cluster is an equivalence relation)

"Impossibility" of clustering

Clustering method is a map $\mathcal{F} : (\mathcal{D} = \{x_i\}_{i=1}^n, \rho) \mapsto \{C_1, \ldots, C_K\}$ which takes as an input dataset $\mathcal{D}$ and a dissimilarity function ρ and returns a partition of $\mathcal{D}$. Three basic properties required

- **Scale invariance.** For any $\alpha > 0$, $\mathcal{F}(\mathcal{D}, \alpha\rho) = \mathcal{F}(\mathcal{D}, \rho)$.
- **Richness.** For any partition $C = \{C_1, \ldots, C_K\}$ of $\mathcal{D}$, there exists dissimilarity ρ, such that $\mathcal{F}(\mathcal{D}, \rho) = C$.
- **Consistency.** If ρ and ρ' are two dissimilarities such that for all $x_i, x_j \in \mathcal{D}$ the following holds:

$$\begin{aligned} x_i, x_j \text{ belong to the same cluster in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \leq \rho(x_i, x_j) \\ x_i, x_j \text{ belong to different clusters in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \geq \rho(x_i, x_j), \end{aligned}$$

  then $\mathcal{F}(\mathcal{D}, \rho') = \mathcal{F}(\mathcal{D}, \rho)$.

Kleinberg (2003) proves that there exists no clustering method that satisfies all three properties!

Examples of Model-free Clustering

- **K-means clustering**: a partition-based method into K clusters. Finds groups such that variation within each group is small. The number of clusters K is usually fixed beforehand or various values of K are investigated as a part of the analysis.
- **Spectral clustering**: Similarity/dissimilarity between data items defines a graph. Find a partition of vertices which does not “cut" many edges. Can be interpreted as nonlinear dimensionality reduction followed by K-means.
- **Hierarchical clustering**: nearby data items are joined into clusters, then clusters into super-clusters forming a hierarchy. Typically, the hierarchy forms a binary tree (a **dendrogram**) where each cluster has two “children” clusters. Dendrogram allows to view the clusterings for each possible number of clusters, from 1 to n (number of data items).

K-means

The K-means algorithm is a widely used method that returns a **local optimum** of the objective function W, given by the total within-cluster deviance:

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2$$

using iterative and alternating minimization.

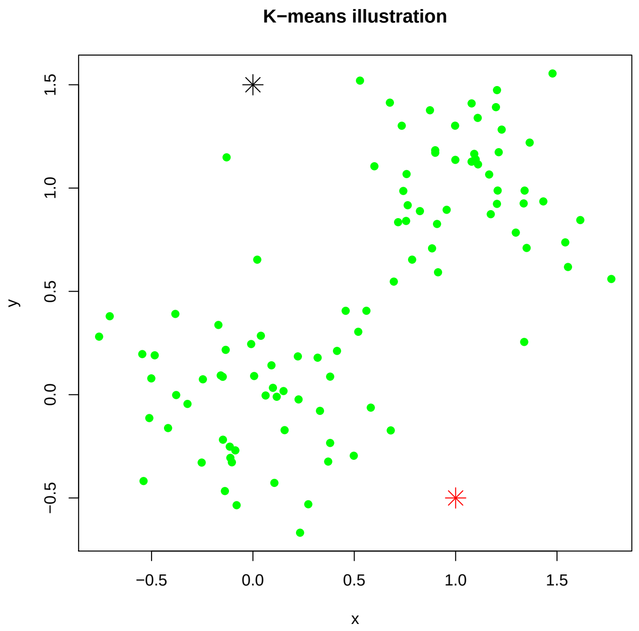
1. Randomly initialize K cluster centroids $\mu_1, \ldots, \mu_K$.
2. **Cluster assignment:** For each $i = 1, \ldots, n$, assign each x_i to the cluster with the nearest centroid,

   $$c_i := \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2$$

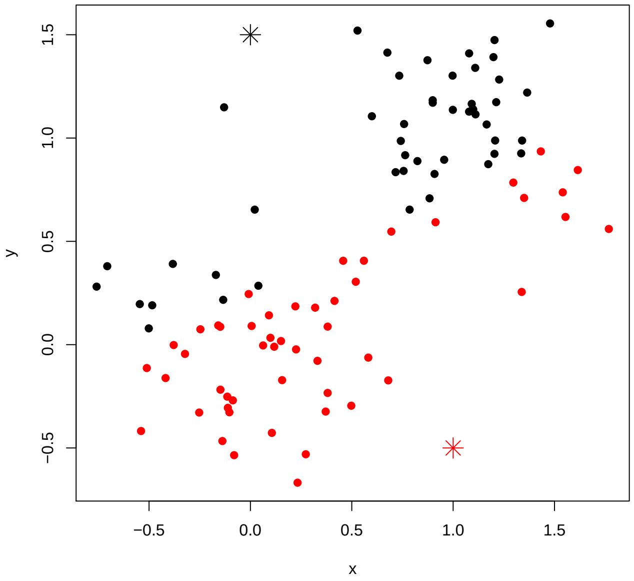
   Set $C_k := \{i : c_i = k\}$ for each k.
3. **Move centroids:** Set $\mu_1, \ldots, \mu_K$ to the averages of the new clusters:

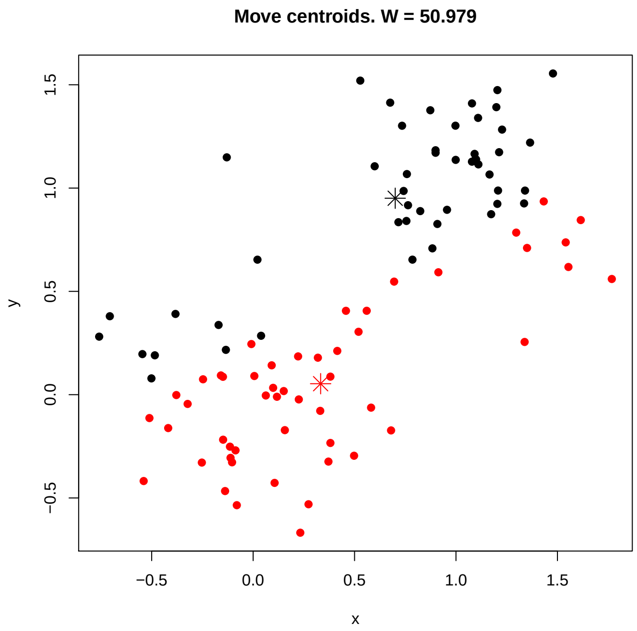
   $$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

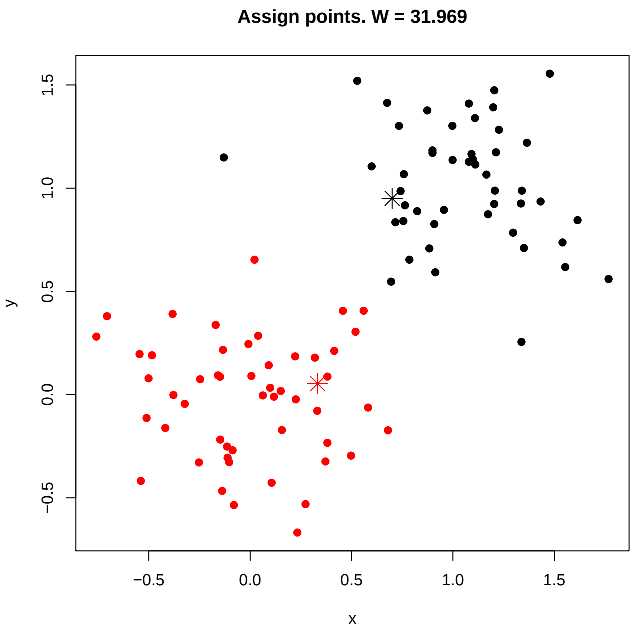
4. Repeat steps 2-3 until convergence.
5. Return the partition $\{C_1, \ldots, C_K\}$ and means $\mu_1, \ldots, \mu_K$.

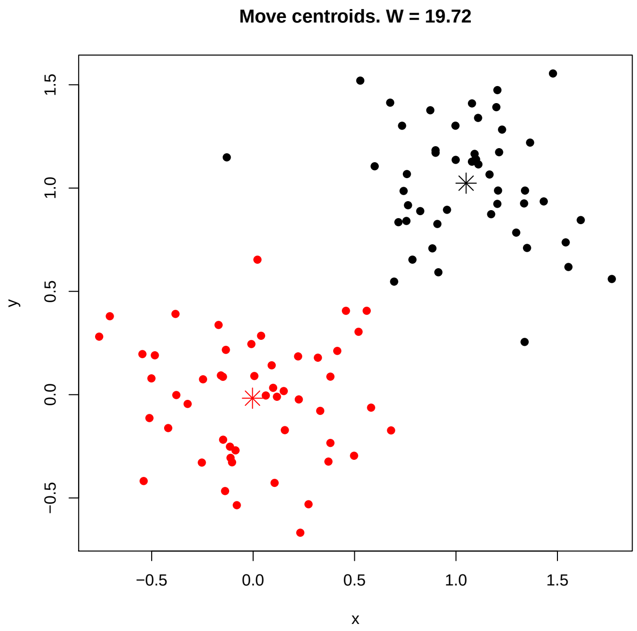

K−means illustration
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5

Assign points. W = 128.1

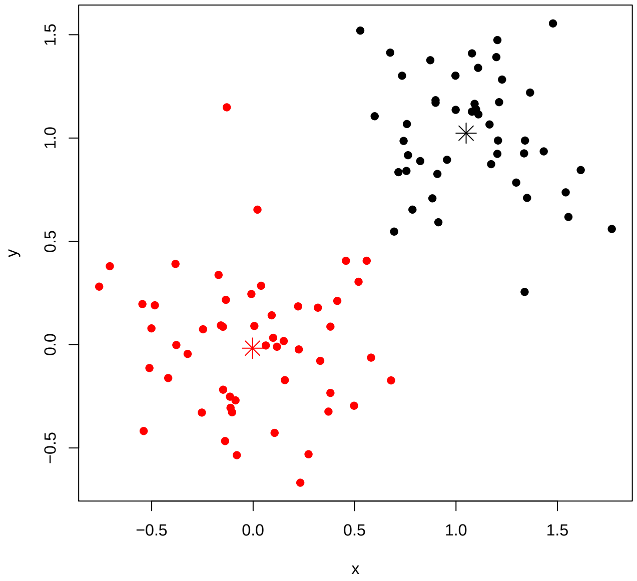


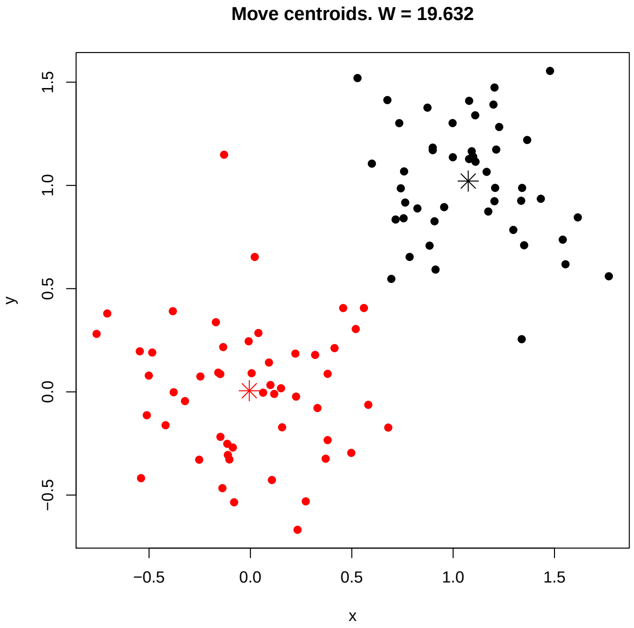

Move centroids. W = 50.979
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Assign points. W = 31.969
1.5
1.0
0.5
0.0
−0.5
y
−0.5
0.0
0.5
1.0
1.5
x


Move centroids. W = 19.72
y
x
−0.5
0.0
0.5
1.0
1.5

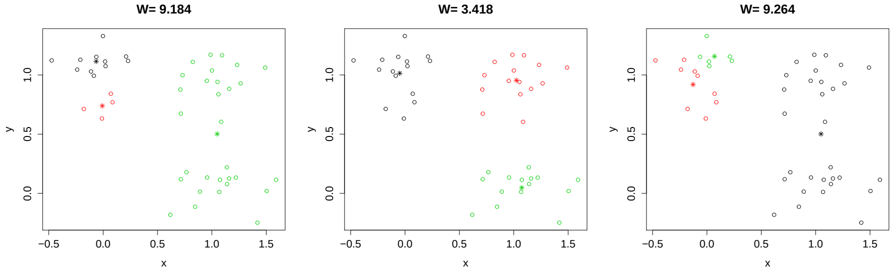
Assign points. W = 19.688




Move centroids. W = 19.632
y
x
−0.5
0.0
0.5
1.0
1.5
−0.5
0.0
0.5
1.0
1.5

K-means

- **The algorithm stops in a finite number of iterations.** Between steps 2 and 3, W either stays constant or it decreases, this implies that we never revisit the same partition. As there are only finitely many partitions, the number of iterations cannot exceed this.
- **The K-means algorithm need not converge to global optimum.** K-means is a heuristic search algorithm so it can get stuck at suboptimal configurations. The result depends on the starting configuration. Typically perform a number of runs from different configurations, and pick the end result with minimum W.



K-means Additional Comments

- **Good practice initialization.** Set centroids $\mu_1, \mu_2, \ldots, \mu_K$ equal to a subset of training examples (chosen without replacement). Initialization using weighted sampling of training examples (**K-means++**) has precise theoretical guarantees[1]
- **Sensitivity to distance measure.** Euclidean distance can be greatly affected by measurement unit and by strong correlations. Can use Mahalanobis distance instead:

$$\|x - y\|_M = \sqrt{(x-y)^\top M^{-1}(x-y)}$$

  where M is positive semi-definite matrix, e.g. sample covariance.
- **Determination of K.** The K-means objective will always improve with larger number of clusters K. Determination of K requires an additional **regularization** criterion. E.g., in **DP-means**[2], use

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 + \lambda K$$

Supervised learning basics

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 $\{1, \dots, K\}$.
  - 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 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).

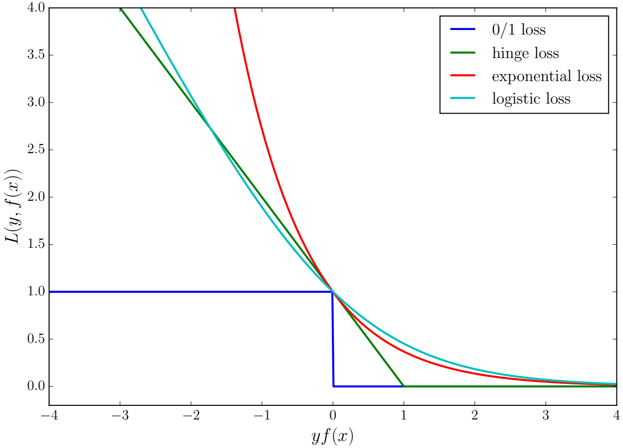


Figure: Loss functions for binary classification

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).

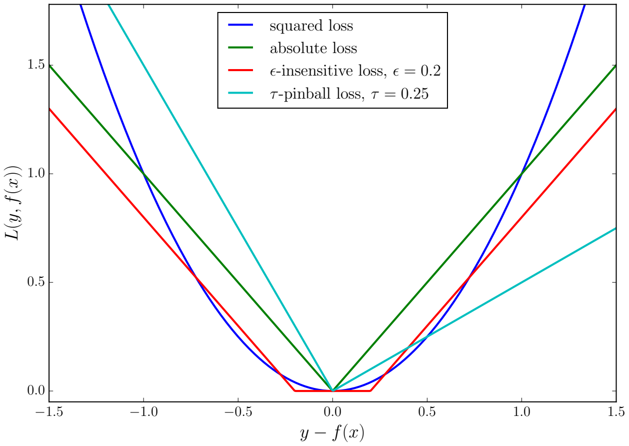


Figure: Loss functions for regression

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**:

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

Hypothesis Space and Empirical Risk Minimization

- The goal of learning is to find the function in **hypothesis space** $\mathcal{H}$ which minimises the risk:

$$f_\star = \operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{X,Y}[L(Y, f(X))]$$

- **Empirical Risk Minimization** (ERM): minimize the empirical risk instead, since we typically do not know $P_{X,Y}$.

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))$$

- Hypothesis space $\mathcal{H}$ is the space of functions f under consideration.
- How complex should we allow functions f to be? If hypothesis space $\mathcal{H}$ is “too large”, ERM can lead to **overfitting**.

$$\hat{f}(x) = \begin{cases} y_i & \text{if } x = x_i, \\ 0 & \text{otherwise} \end{cases}$$

will have zero empirical risk, but is useless for generalization, since it has simply “memorized” the dataset.

Examples of Hypothesis Spaces

Say $\mathcal{X} \subseteq \mathbb{R}^p$.

- all linear functions $f(x) = w^\top x + b$, parametrized by $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$
- consider a specific **nonlinear feature expansion** $\varphi : \mathcal{X} \to \mathbb{R}^D$, with $D > p$ and use functions linear in those features: $f(x) = w^\top \varphi(x) + b$, but nonlinear in the original inputs $\mathcal{X}$, parametrized by $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$. For example, starting with $\mathcal{X} = \mathbb{R}^2$, we can consider $\varphi\left(\begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}\right) = [x_{i1}, x_{i2}, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2]^\top$, such that the resulting function can depend on quadratic and interaction terms as well.
- In this course, we will study an important type of hypothesis space: **Reproducing Kernel Hilbert Space (RKHS)**.

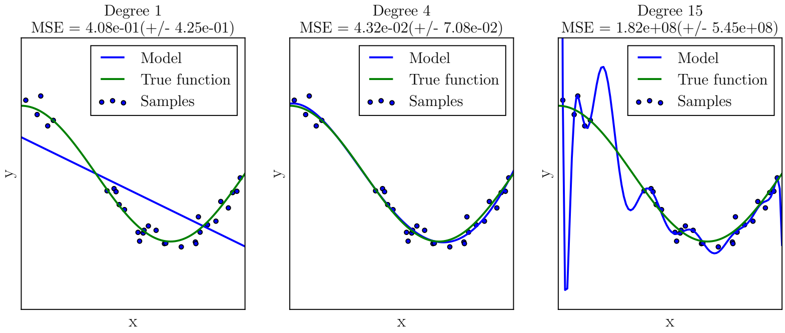


Figure: Underfitting and Overfitting

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:

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

- directly penalise some notion of **smoothness** of function f, e.g. for $\mathcal{X} = \mathbb{R}$, the regularisation term can consist of the **Sobolev norm**

$$\|f\|_{W^1}^2 = \int_{-\infty}^{+\infty} f(x)^2 dx + \int_{-\infty}^{+\infty} f'(x)^2 dx, \tag{4}$$

which penalises functions with large derivative values.