Introduction to Medical Imaging (5XSA0) Module 6

Machine Learning

Sveta Zinger & Fons van der Sommen

Video Coding and Architectures Research group, TU/e
(s.zinger@tue.nl , f.v.d.sommen@tue.nl)

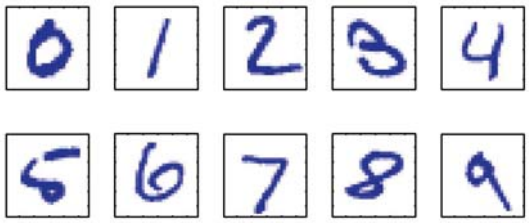
Introduction – (1)

* Example: how to recognize handwritten digits automatically?
  - We want to build a machine with
    - image of a digit as input
    - identity of the digit (0,…,9) as output
  - Why is it difficult?
    - Wide variability of handwriting
    - Rules or heuristics do not work

(The slides are based on “Pattern recognition and machine learning”, Ch. Bishop)

Introduction – (2)

Examples of handwritten digits taken from US zip-codes



Introduction – (3)

* Possible solution – machine learning
  - Train the algorithm using a training set (digits) and target vector (their identities)
  - Test it with a test set (new images of digits)
  - Generalization – ability to categorize new examples correctly
* How can we facilitate pattern recognition?
  - Preprocess data in the training set – extract features (see module 4)

Introduction – (4)

* Supervised learning
  - Training data consists of input vectors and target vectors
  - Classification - assign each input vector to one of a finite number of discrete categories
* Unsupervised learning
  - No target vector in the input data
  - Clustering – discover groups of similar examples within the data

Supervised learning

* Catogarized / labeled data
  - Objects in a picture: chair, desk, person,…
  - Handwritten digits: → 3, → 6, → 5
  - Medical diagnosis: OCT image → T2 colon cancer
* Goal: identify the class of a new data point
* Statistical modeling and machine learning
* Disctinctive properties (features)

Supervised learning: example (1/5)

* Separate lemons from oranges

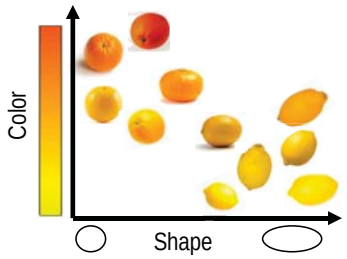
Color: orange
Shape: sphere
Ø: ± 8 cm
Weigth: ±0.1 kg

Color: yellow
Shape: elipsoid
Ø: ± 8 cm
Weigth: ±0.1 kg

* Use “color” and “shape” as features

Supervised learning: example (2/5)

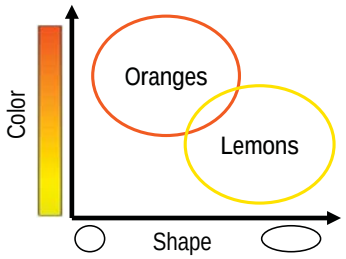
* Separate lemons from oranges



Supervised learning: example (3/5)

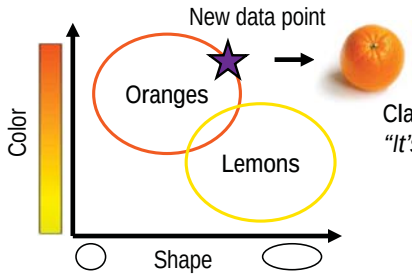
* Separate lemons from oranges

Model the given
- *training* - data



Supervised learning: example (4/5)

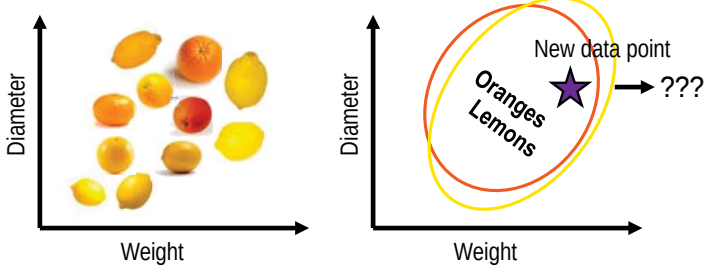
* Separate lemons from oranges



Classifier:
“It’s an orange!”

Supervised learning: example (5/5)

* What if we had chosen the wrong features?



Supervised learning

Summary

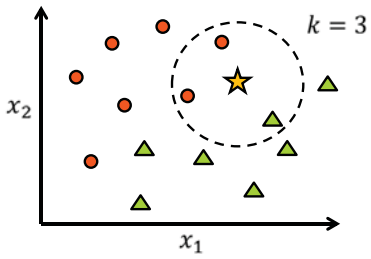
* Choose distinctive features
* Make a model based on labeled data (a.k.a. supervised learning)
* Use the *learned* model to predict the class of new, unseen data points

Models for classification

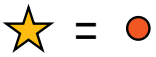
* Support Vector Machine (SVM)
* k Nearest Neighbours (k-NN)
* Random Forests
* Boosting
* Neural Networks
* …

k Nearest Neighbours (1)

* Simple concept: look at the class of the k closest neighbours in feature space

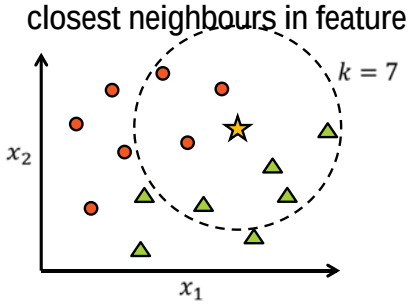


For 3 nearest neighbours:

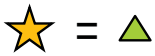


k Nearest Neighbours (2)

* Simple concept: look at the class of the k closest neighbours in feature space



For 7 nearest neighbours:



k Nearest Neighbours (3)

* Type of instance based learning
  - A.k.a. memory based learning
  - New instance compared to training instances that are stored in memory: no explicit modelling
  - Very memory-heavy classification method!
* Two important parameters
  - Number of neighbours k
  - Distance metric

k Nearest Neighbours (4)

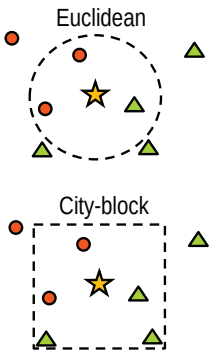
* Distance metrics
  - Euclidean distance (L^2-norm)

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{D}(p_i - q_i)^2}$$

  - City block distance (L^1-norm)

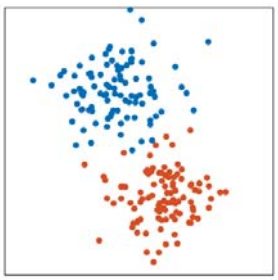
$$d(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{D}|p_i - q_i|$$

  - Many more options!

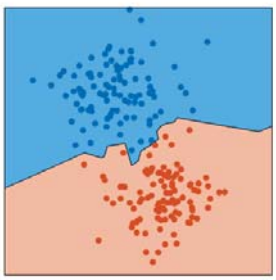


k Nearest Neighbours
neigbours & generalization (1)

2 classes in feature space

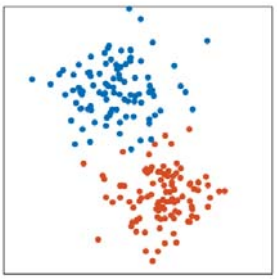


k-NN decision for k=1

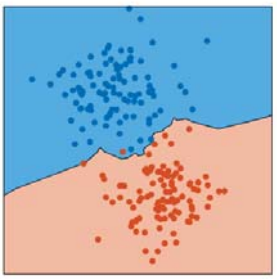


k Nearest Neighbours
neigbours & generalization (2)

2 classes in feature space

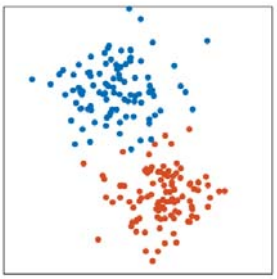


k-NN decision for k=3

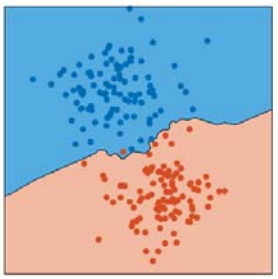


k Nearest Neighbours
neigbours & generalization (3)

2 classes in feature space

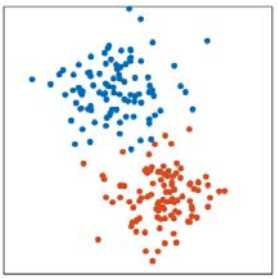


k-NN decision for k=5

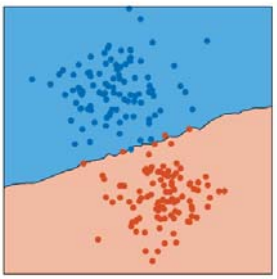


k Nearest Neighbours
neigbours & generalization (4)

2 classes in feature space

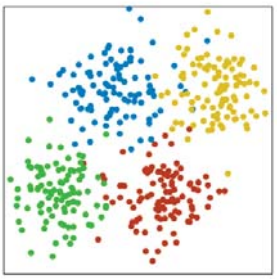


k-NN decision for k=10

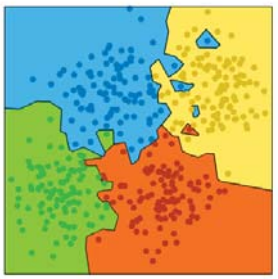


k Nearest Neighbours
neigbours & generalization (5)

4 classes in feature space

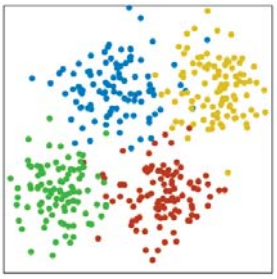


k-NN decision for k=1

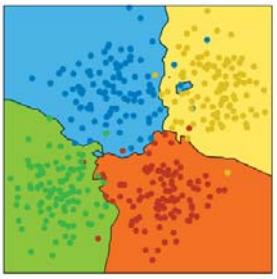


k Nearest Neighbours
neigbours & generalization (6)

4 classes in feature space

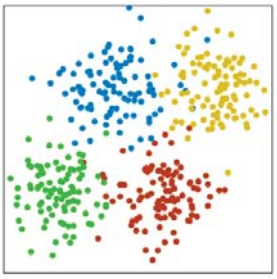


k-NN decision for k=3

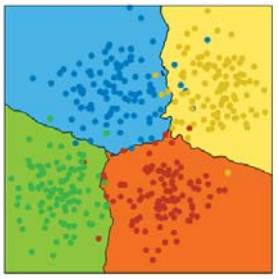


k Nearest Neighbours
neigbours & generalization (7)

4 classes in feature space

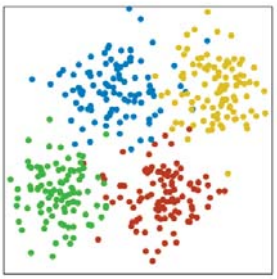


k-NN decision for k=10

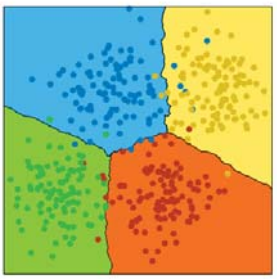


k Nearest Neighbours
neigbours & generalization (8)

4 classes in feature space



k-NN decision for k=25

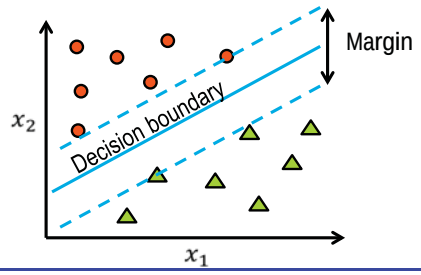


k Nearest Neighbours

- Summary
  - Instance learning: no explicit modeling
  - Memory heavy: all training samples are stored
  - Two important parameters
    1. *Number of nearest neighbours k*
    2. *Distance metric d*
  - Different parameters choices can lead to different results!
  - Higher k leads to better generalization, but also makes classification of a new sample a lot slower!

Support Vector Machine (SVM) (1)

- Find a hyperplane that separates the classes with a maximum margin

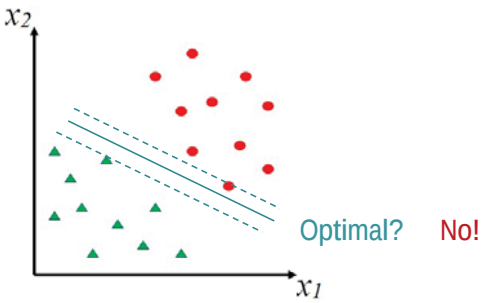


Support Vector Machine (SVM) (2)

- Based on emperical risk minimization (1960s)
  - Non-linearity added in 1992 (Boser, Guyon & Vapnik)
  - Soft-margin SVM introduced in 1995 (Cortes & Vapnik)
- Has become very popular since then
  - Easy to use, a lot of open libraries available
  - Fast learning and very fast classification
  - Good generalization properties

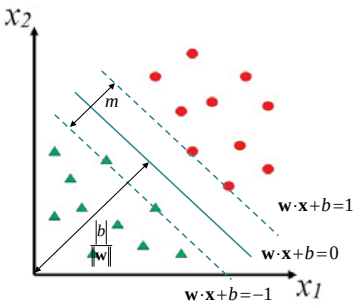
Support Vector Machine (SVM) (3)

- How to find the optimal hyperplane?



Support Vector Machine (SVM) (4)

- How to find the optimal hyperplane?



Width of the margin:

$$m = \frac{|b|+1}{\|\mathbf{w}\|} - \frac{|b|-1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Maximize margin:

$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|}$$

$$\text{subject to } \mathbf{w}^T x_i + b \begin{cases} \geq 1 & y_i = 1 \\ \leq -1 & y_i = -1 \end{cases}$$

Support Vector Machine (SVM) (5)

* We can rewrite this to

$$\min_{\mathbf{w},b} \|\mathbf{w}\|$$
$$\text{subject to } y_i(\mathbf{w}^T x_i + b) \geq 1$$

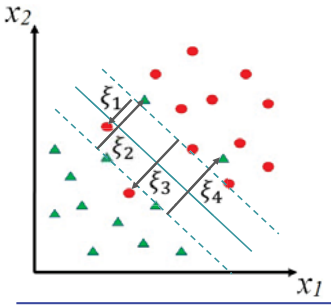
* Formulate as a Quadratic Programming problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{subject to } y_i(\mathbf{w}^T x_i + b) \geq 1$$

Efficient methods available to solve this problem!

Support Vector Machine (SVM) (6)

* The data is usually not linearly separable…



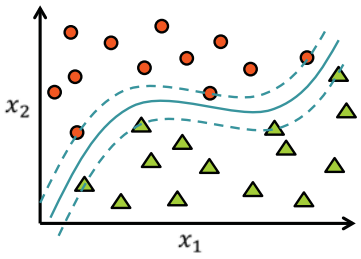
Introduce slack variables ξ_i

Put a cost *C* on crossing the margin, so the optimization problem becomes:

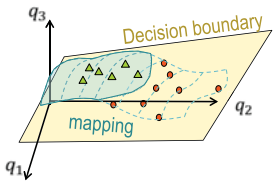
$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T + b) \geq 1$$

Support Vector Machine (SVM) (7)

* A more complex extension: non-linear SVMs

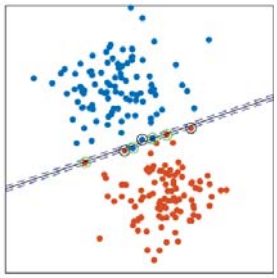


Basic idea: map the data to a higher-dimensional space, in which we can apply a linear SVM

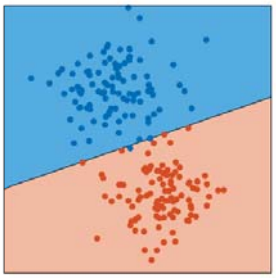


Support Vector Machine (SVM)
Cost parameter & generalization (1)

Optimal hyperplane for C=100

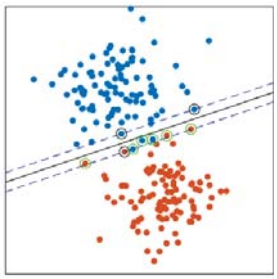


SVM decision for C=100

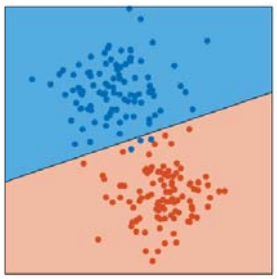


Support Vector Machine (SVM)
Cost parameter & generalization (2)

Optimal hyperplane for C=10

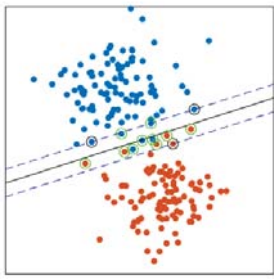


SVM decision for C=10

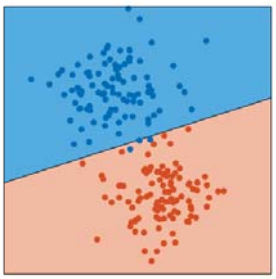


Support Vector Machine (SVM)
Cost parameter & generalization (3)

Optimal hyperplane for C=1

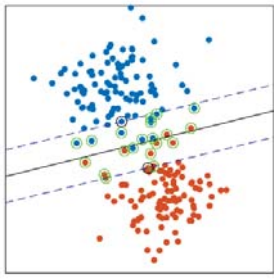


SVM decision for C=1

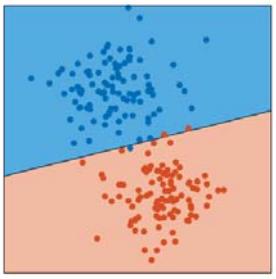


Support Vector Machine (SVM)
Cost parameter & generalization (4)

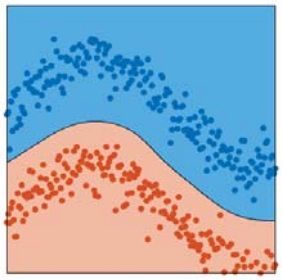
Optimal hyperplane for C=0.1

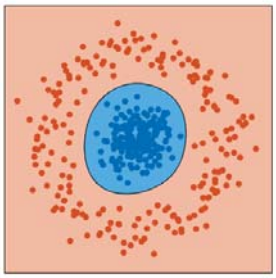


SVM decision for C=0.1



Support Vector Machine (SVM)
Non-linear SVM examples





Support Vector Machine (SVM)

* Summary
  - Fast and efficient method for binary classification
  - Splits the classes based on maximizing the margin
  - Optimal hyperplane can be computed using Quadratic Programming
  - Cost-parameter for points crossing the margin
  - Non-linear SVM can also handle more complex class distributions by mapping the data to another space

Random Forest (1)

* Build decision trees on subsets of the data
* Let the trees vote on the class of a new sample

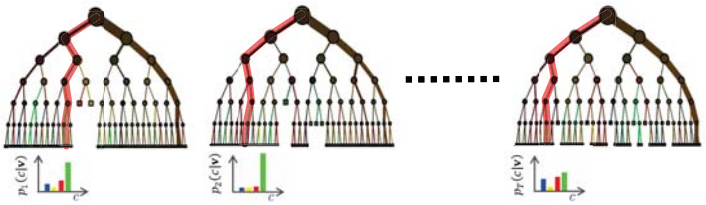


Image from a tutorial of Antonio Criminisi download at: http://research.microsoft.com/en-us/projects/decisionforests/

Random Forest (2)

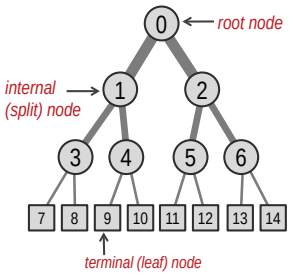
* Robustness through randomness
  - A random subset is used to train each tree
  - For training a tree, each node receives a random set of split options
* Intrinsically probabilistic output
  - Measure of confidence / uncertainty
* Automatic feature selection
* Naturally multi-class
* Runs efficiently – trees can run in parallel

Random Forest
Decision trees (1)

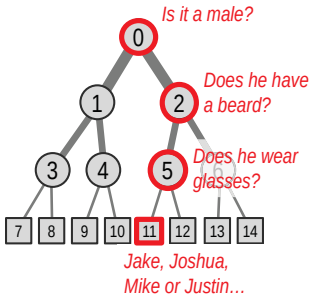
A forest consists of trees

* Start at the root node
* True/false question at each split node
* Stop when a leaf node is reached: prediction

A general tree structure



Random Forests
Decision trees (2)



Example: GUESS WHO*

*Credits to Mark Janse

Randomized Node Optimization (RNO)

Random Forests
Decision trees (3)

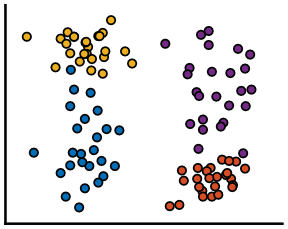
Bagging

* How to train a decision tree?
  - Start with a subset of all the training data at the root node
  - From a set of randomly chosen split options $\boldsymbol{\theta}$, select the one that maximizes some split metric (e.g. information gain)
  - Repeat this for all the nodes and stop growing a certain branch untill one of the following two criteria holds:
    - A pre-defined tree depth D is reached (# nodes of a branch)
    - All trianing samples in the node are from the same class

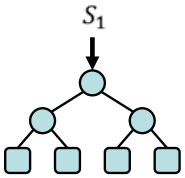
Random Forests
How to grow a tree? (1)

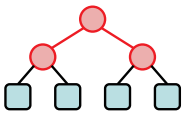
* Let's grow a tree with depth D = 2:

Subset S_1 of all availabe data S

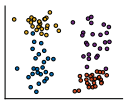


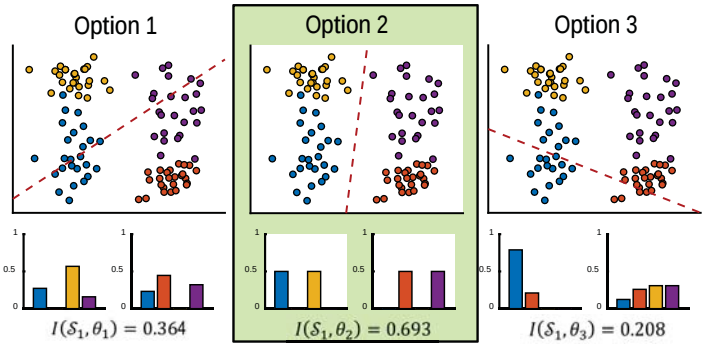
Start at the root node

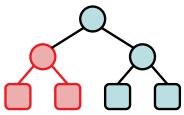




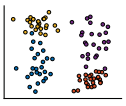
Random Forests
How to grow a tree? (2)

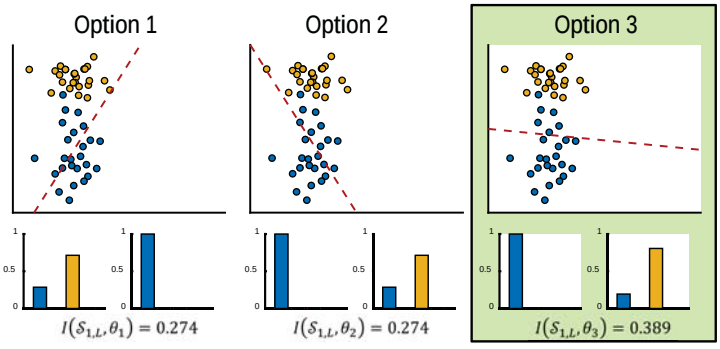


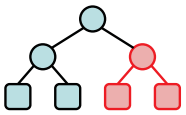




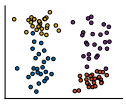
Random Forests
How to grow a tree? (3)

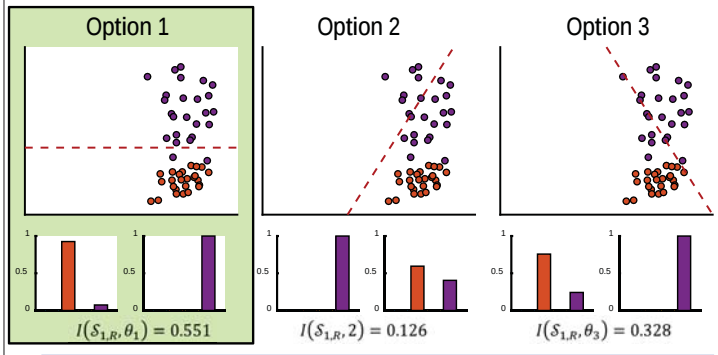






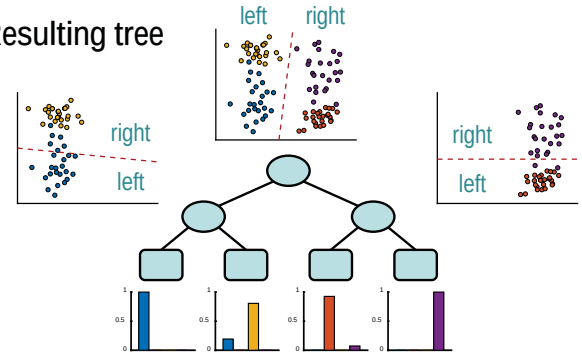
Random Forests
How to grow a tree? (4)





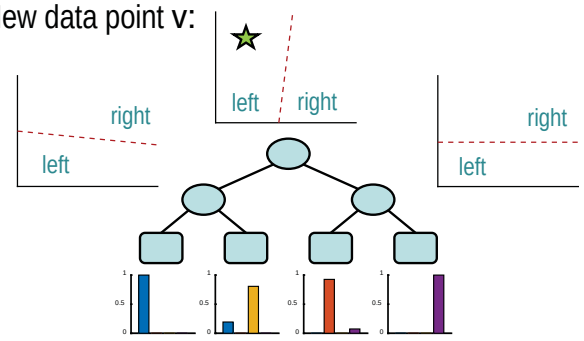
Random Forests
How to grow a tree? (5)

* Resulting tree



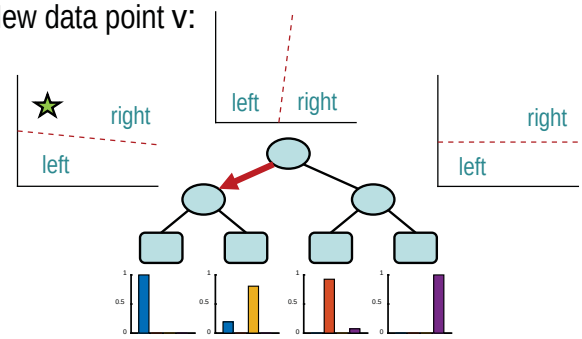
Random Forests
Classify a new data point (1)

* New data point v:



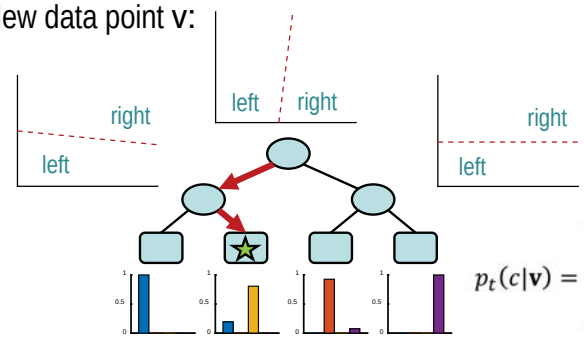
Random Forests
Classify a new data point (2)

* New data point v:



Random Forests
Classify a new data point (3)

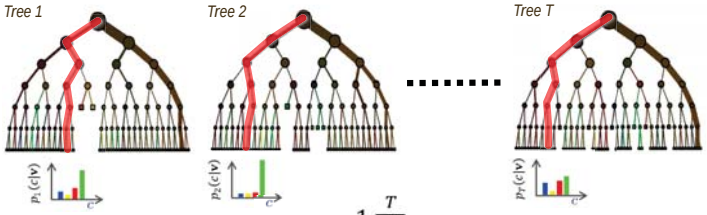
* New data point v:



$$p_t(c|\mathbf{v}) = \begin{bmatrix} 0.19 \\ 0 \\ 0.81 \\ 0 \end{bmatrix}$$

Random Forests
Classification examples

* How to combine tree output?

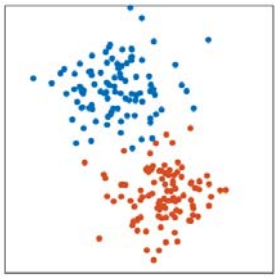


- Averaging: $p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v})$

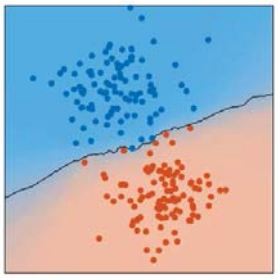
Image from a tutorial of Antonio Criminisi download at: http://research.microsoft.com/en-us/projects/decisionforests/

Random Forests
Classification example (1)

2 classes in feature space



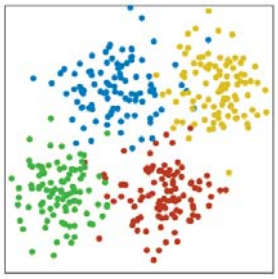
Random forest decision



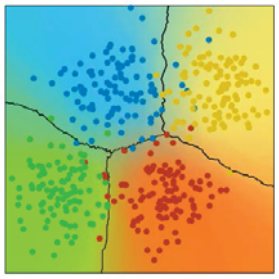
N = 100 trees, max number of nodes = 5, # candidate splits per node = 3

Random Forests
Classification example (2)

4 classes in feature space



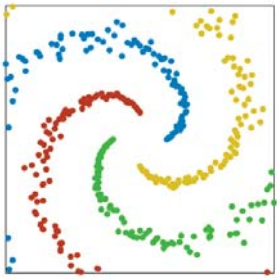
Random forest decision



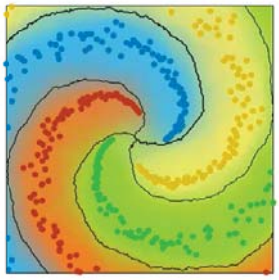
N = 100 trees, max number of nodes = 4, # candidate splits per node = 3

Random Forests
Classification example (3)

4 classes in feature space



Random forest decision



N = 100 trees, max number of nodes = 10, # candidate splits per node = 8

Random Forests

* Summary
  - Good generalization due to randomness model
    - Bagging
      - Each tree is trained on a randomly selected subset of the data
    - Randomized Node Optimization (RNO)
      - Each node receives a randomly selected subset of all possible split options.
  - Multi-class classification with probablistic output
  - Suboptimal splits lead to a robust model
  - Result depends heavily on the forest parameters

Performance evaluation

* So, now we have model, how good is it?
  - We have labeled data (ground truth), so we can validate!
* Model validation:
  - Separate sets for training and testing the model
    - Train the model using the training set
    - Use the test set to evaluate the performance
  - Compute figures of merit, which indicate the performance
  - What is a good performance metric? And how should we split the data?

Performance evaluation

Number of samples

* Some popular figures of merit:
  - Accuracy (#TP + #TN) / (#TP + #FN +#TN + #FP)
  - Sensitivitiy (#TP) / (#TP + #FN) *a.k.a. True Positive Rate*
  - Specificity (#TN) / (#TN + #FP) *a.k.a. True Negative Rate*

Where

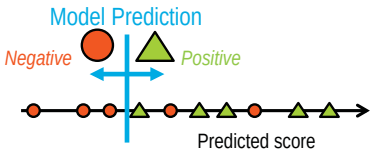
True Positive (TP):	positive sample classified as positive
True Negative (TN):	negative sample classified as negative
False Positive (FP):	negative sample classified as positive
False Negative (FN):	positive sample classified as negative

Performance evaluation

* Receiver Operating Characteristic (ROC)
  - Sensitivity / specificity give the performance for just one possible setting (i.e. decition threshold) of the model
  - We can vary this threshold and recompute these performance metrics
  - This yields a curve of possible combinations of sensitivity and specificity, called the ROC curve
  - Generally true: ↑ sensitivity ↓ specificity and vice versa

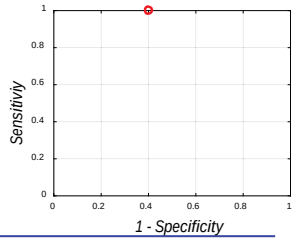
Performance evaluation

* How to compute the ROC curve?
  - For each sample we have a predicted class and a score
  - Sort the samples according to score and move the threshold



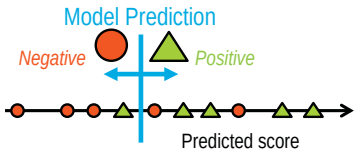
Sensitivitiy = 5 / (5+0) = 1.00

Specificity = 3 / (3+2) = 0.60



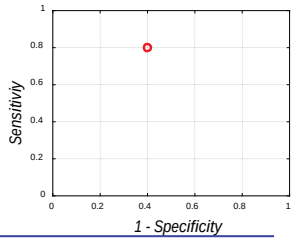
Performance evaluation

* How to compute the ROC curve?
  - For each sample we have a predicted class and a score
  - Sort the samples according to score and move the threshold



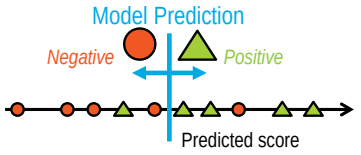
Sensitivitiy = 4 / (4+1) = 0.80

Specificity = 3 / (3+2) = 0.60



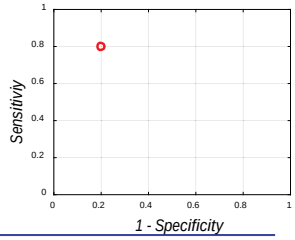
Performance evaluation

* How to compute the ROC curve?
  - For each sample we have a predicted class and a score
  - Sort the samples according to score and move the threshold



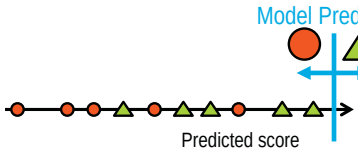
Sensitivitiy = 4 / (4+1) = 0.80

Specificity = 4 / (4+1) = 0.80



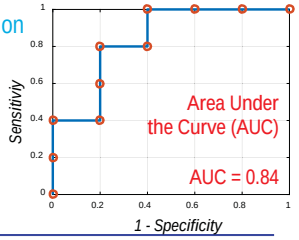
Performance evaluation

* How to compute the ROC curve?
  - For each sample we have a predicted class and a score
  - Sort the samples according to score and move the threshold



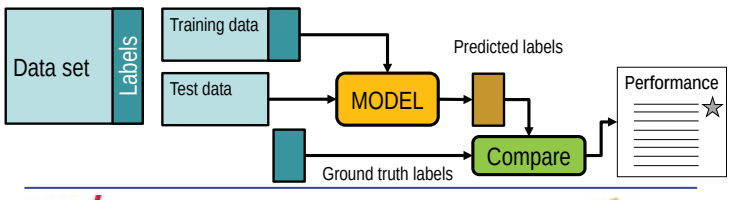
Sensitivitiy = 0 / (0+5) = 0.00

Specificity = 5 / (5+0) = 1.00



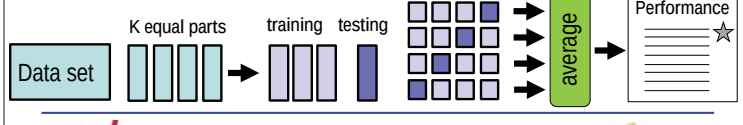
Performance evaluation

* Large data set: randomly sample half the samples for training and half for testing
  - Training and testing is time consuming for large datasets
  - The test set is probably a good reflection of the training set



Performance evaluation

* How should we split the data?
  - Different choices might lead to different results…
* K-fold cross-validation
  - Split the data in K equally sized parts
  - Use K-1 parts for training and use the left-out part of the data for testing, repeat this for each part and average:

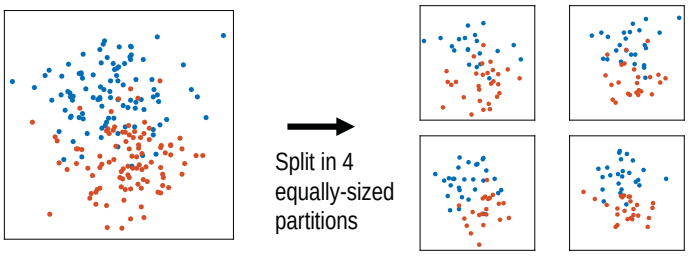


Performance evaluation

- Leave-One-Out Cross-Validation
  - Leave one sample out of the complete set and use the remaining set to train the model
  - Test the model on the left-out sample
  - Repeat this for all samples.
- Best performance indication for small data set
  - You want to use as much of the little data you have for training the model

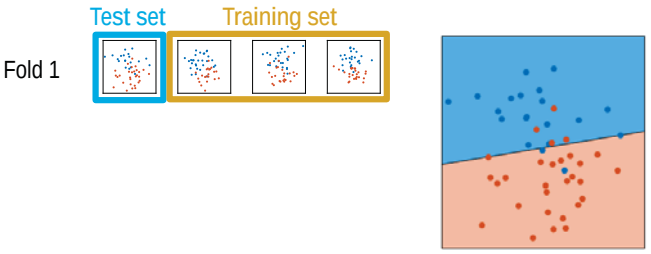
Performance evaluation

EXAMPLE: 4-fold cross validation (1)



Performance evaluation

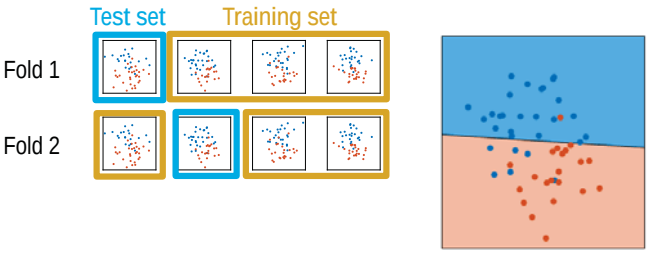
EXAMPLE: 4-fold cross validation (2)



Fold 1: Accuracy = 0.86

Performance evaluation

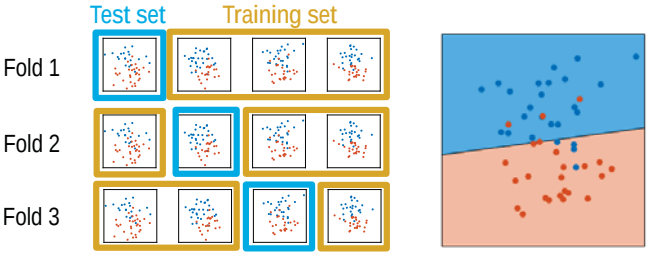
EXAMPLE: 4-fold cross validation (3)



Fold 2: Accuracy = 0.86

Performance evaluation

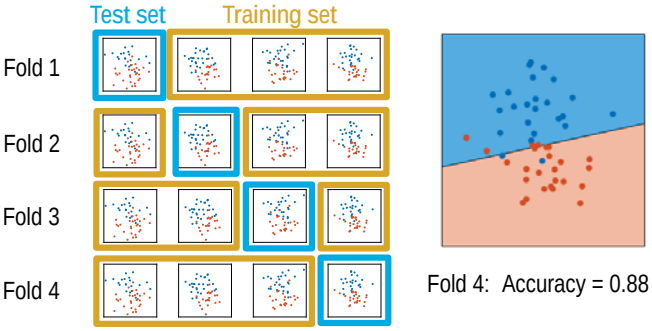
EXAMPLE: 4-fold cross validation (4)



Fold 3: Accuracy = 0.84

Performance evaluation

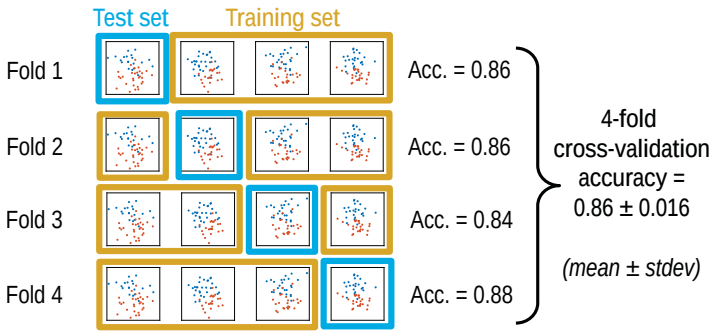
EXAMPLE: 4-fold cross validation (5)



Fold 4: Accuracy = 0.88

Performance evaluation

EXAMPLE: 4-fold cross validation (6)



Performance evaluation

- Summary:
  - In supervised learning the "ground truth" is available, so we can evaluate the prediction performance of the model
  - Split the data in two sets
    - Training set used for training the model
    - Test set to evaluate the prediction performance
  - There are different figures of merit for measuring the performance: Accuracy, Sensitivity, Specificity, AUC,…
  - Use K-fold cross-validation for reliable evaluation

Unsupervised learning

- Uncatogarized / unlabeled data
  - No target vector in the input data
- Goal: discover groups of similar data points
  - Clustering
    - K-means
    - Gaussian mixture models
  - Latent variable models
    - Principal component analysis

K-means clustering – (1)

- Problem: identify groups (clusters) of data points in multidimensional space
  - we have a data set $\{x_1,\ldots,x_N\}$,
  - variable x - D-dimensional
  - goal: partition data into K clusters, value of K is given
- Intuitive definition of cluster
  - group of data points whose inter-point distances are small compared with the distances to points outside of the cluster

K-means clustering – (2)

- Distortion measure

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\left\|x_n - \mu_k\right\|^2$$

  - where $r_{nk} \in \{0,1\}$ - binary indicator variable: $r_{nk} = 1$ if data point x_n is assigned to cluster k and $r_{nk} = 0$ otherwise,
    x_n - data point,
    μ_k - vector assigned to cluster k (center of cluster)
  - it is sum of the squares of the distances of each data point to its assigned vector

K-means clustering – (3)

- Goal: find values for $\{r_{nk}\}$ and $\{\mu_k\}$ that minimize J
- How can we find the solution?
  - Iterative procedure
    - each iteration involves two successive steps
    - successive optimizations with respect to $\{r_{nk}\}$ and $\{\mu_k\}$
    - repeat until convergence
      - no further change in the assignments
      - or until a maximum number of iterations is exceeded

K-means clustering – (4)

* Description of algorithm
  - Choose some initial values for the $\{\mu_k\}$
  - First phase
    - Minimize J with respect to $\{r_{nk}\}$ keeping $\{\mu_k\}$ fixed
  - Second phase
    - Minimize J with respect to $\{\mu_k\}$ keeping $\{r_{nk}\}$ fixed
  - Repeat until convergence

K-means clustering – (5)

* First phase of algorithm
  - Determine $\{r_{nk}\}$ - assign data points to clusters
    - Optimize for each n separately

$$r_{nk} = \begin{cases} 1 & if \quad k = \arg\min_j \|x_n - \mu_j\|^2 \\ 0 & otherwise \end{cases}$$

K-means clustering – (6)

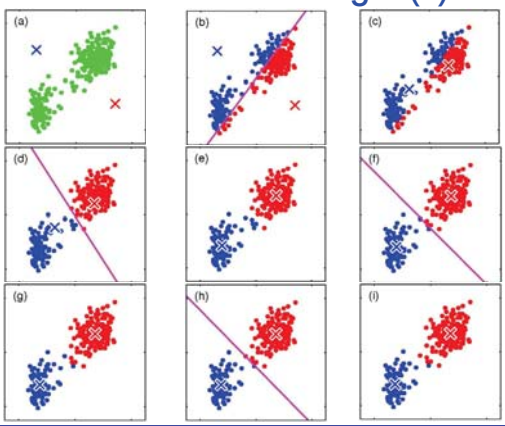
* Second phase of algorithm
  - Determine $\{\mu_k\}$ - compute the cluster means
    - J is a quadratic function of $\{\mu_k\}$, set its partial derivative to zero for finding its minimum

$$\frac{dJ}{d\mu_k} = 2\sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0$$

then $$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

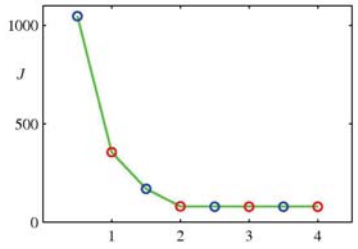
K-means clustering – (7)

Example



K-means clustering – (8)

* Example: minimization of cost function J
  - Blue points – after assigning data points to clusters
  - Red points – computing cluster means
  - Algorithm converges after the third step, final cycle produces no changes



K-means clustering – (9)

* What are the limits of this algorithm?
  - Algorithm is based on Euclidean distance as the measure of dissimilarity between a data point and a prototype vector
    - ⇒ Data types are limited (for example, categorical labels cannot be used)
    - ⇒ determination of the cluster is not robust to outliers

K-means clustering – (10)

* ***K***-medoids algorithm
  - Generalization of the K-means
  - Introduces a more general dissimilarity measure $\mathcal{V}$
  - The distortion measure to minimize is then

$$\tilde{J} = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\mathcal{V}(x_n, \mu_k)$$

  - But the computation of centers of clusters is more complicated

K-means clustering – (11)

* Property of ***K***-means algorithm
  - Every data point is assigned uniquely to one of the clusters
  - but some data points lie roughly midway between cluster centers
  - and it is not clear that the hard assignment to the nearest cluster is most appropriate
* What kind of assignment would be better?
  - Adopt a probabilistic approach => soft assignments

K-means clustering
Image segmentation and compression – (1)

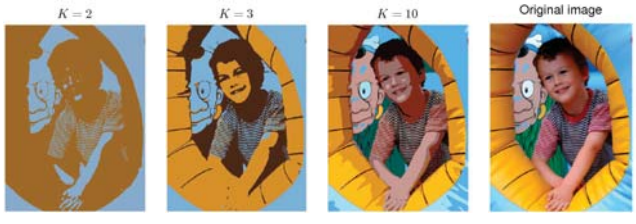
* Some applications of ***K***-means algorithm
  - Image segmentation
  - Image compression
* What is the goal of segmentation?
  - Partition an image into regions each of which
    - has a reasonably homogeneous visual appearance or
    - corresponds to objects or parts of object

K-means clustering
Image segmentation and compression – (2)

* Segmentation
  - Each pixel is a separate {R,G,B} 3D data point
  - Apply K-means to these points
  - Redraw the image replacing each pixel vector with the {R,G,B} intensity triplet given by the center $\boldsymbol{\mu}_k$ to which this pixel is assigned

K-means clustering
Image segmentation and compression – (3)

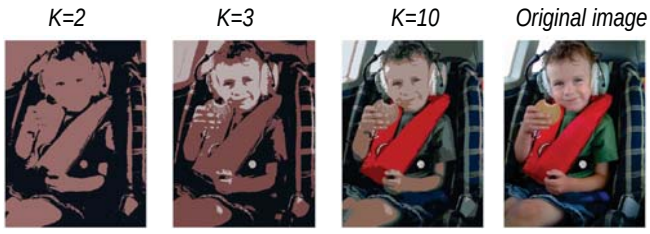
Example: for a given value of K, the algorithm represents the image using a palette of only K colors



Smaller values of K => higher compression => poorer image quality

K-means clustering
Image segmentation and compression – (4)

Example



K-means is not a sophisticated approach to image segmentation because it takes no account of the spatial proximity of different pixels

K-means clustering

Image segmentation and compression – (5)

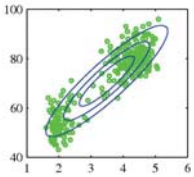
- Application of the *K*-means to lossy data compression
  - For each of the N data points, store only the identity k of the cluster to which it is assigned
  - Store the values of the K cluster centers
    - Requires less data provided that $K<N$
  - Each data point is approximated by its nearest center

It is called vector quantization; cluster centers are called code-book vectors

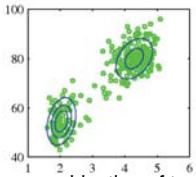
Mixture models

Gaussian mixture – (1)

- Gaussian distribution
  - has some important analytical properties
  - but suffers from limitations when modeling real data sets



Single Gaussian distribution fails to capture the nature of data

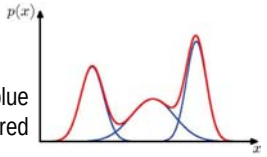


Linear combination of two Gaussians fits better

Mixture models

Gaussian mixture – (2)

- Mixture distributions
  - linear combinations of basic distributions
- To approximate almost any continuous density
  - use sufficient number of Gaussians
  - adjust means, covariances, coefficients in the linear combination



Three Gaussians in blue and their sum in red

Mixture models

Gaussian mixture – (3)

Superposition of K Gaussian densities

$$p(x)=\sum_{k=1}^{K}\pi_k\mathcal{N}\left(x|\mu_\kappa,\Sigma_\kappa\right)$$

is called a mixture of Gaussians, where

π_k - mixing coefficients, $0\le\pi_k\le 1$

$$\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)=\frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Each Gaussian density has its mean μ_k and covariance Σ_k

Mixture models

Gaussian mixture – (4)

- Gaussian mixture distribution is governed by parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

$$\boldsymbol{\pi}=\{\pi_1,\ldots,\pi_K\},\quad \boldsymbol{\mu}=\{\mu_1,\ldots,\mu_K\},\quad \boldsymbol{\Sigma}=\{\Sigma_1,\ldots,\Sigma_K\}$$

- How can we find these parameters?
  - Possible solution – use maximum likelihood
  - Likelihood function expresses how probable the observed data is for a given set of parameters

Mixture models

Gaussian mixture – (5)

- Log of the likelihood function:

$$\ln p(\mathbf{X}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma})=\sum_{n=1}^{N}\ln\left(\sum_{k=1}^{K}\pi_k\mathcal{N}\left(x_n|\mu_\kappa,\Sigma_\kappa\right)\right)$$

$$\mathbf{X}=\{x_1,\ldots,x_N\}$$

  - No easy analytical solution
  - Expectation-maximization (EM) can be used

Mixture models
Gaussian mixture – (6)

* Where are Gaussian mixture models used?
  - Data mining
  - Pattern recognition
  - Machine learning
  - Statistical analysis
* How are their parameters determined?
  - Maximum likelihood using the EM algorithm

Principal Component Analysis
Introduction – (1)

* So far – we discussed probabilistic models having discrete latent variables, such as mixture of Gaussians
* Now – explore models in which some or all of the latent variables are continuous
  - Motivation: property of many data sets – data points can be represented by fewer dimensions than those in the original data space

Principal Component Analysis
Introduction – (2)

* Consider an artificial data set
  - constructed by taking images of digits, represented by 64x64 pixel grey-scale image,
  - and embedding them in larger images of size 100x100 by padding with pixels having the value 0
  - Location and orientation of digits is varied at random

Principal Component Analysis
Introduction – (3)

* Synthetic data set
  - multiple copies of digit images where the digit is randomly displaced and rotated within some larger image field

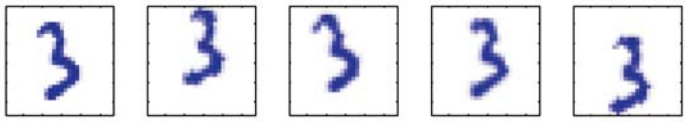


Image size: 100x100 = 10000 pixels

Principal Component Analysis
Introduction – (4)

* Resulting images
  - Represented by a point in the 10000-dimensional data space
  - However, across a data set of these images, there are only three degrees of freedom of variability
    - vertical translation
    - horizontal translation
    - rotation
  - Intrinsic dimensionality is three

Principal Component Analysis
Introduction – (5)

* Translation and rotation parameters in this example
  - latent variables
    - because we observe only the images and are not told which values of the translation or rotation variables where used to create them
  - Real digit image data => more degrees of freedom
    - Scaling, handwriting variability
  - but still smaller amount than the data set dimensionality

Principal Component Analysis
Introduction – (6)

* Such latent variables can be used for
  - data compression
  - density modeling
  - data modeling
    - select a point according to some latent variable distribution
    - generate an observed data point by adding noise, drawn from some conditional distribution of the data variables given the latent variables

Principal Component Analysis – (1)

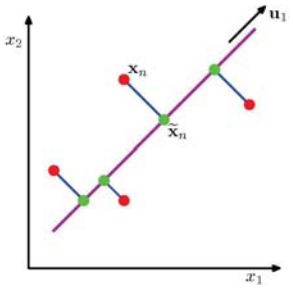
* PCA (Principal Component Analysis) is widely used for
  - dimensionality reduction
  - lossy data compression
  - feature extraction
  - data visualization
* Also known as Karhunen-Loève transform

Principal Component Analysis – (2)

* Two definitions of PCA (give rise to the same algorithm)
  - Orthogonal projection of the data onto a lower dimensional space (principal subspace), such that the variance of the projected data is maximized
  - Linear projection that minimizes the average projection cost, defined as the mean squared distance between the data points and their projections

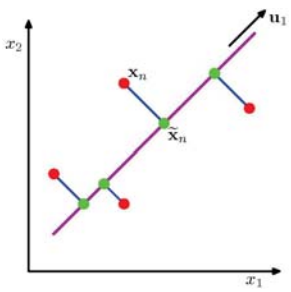
Principal Component Analysis – (3)

PCA seeks a space of lower dimensionality, denoted by the magenta line, such that the orthogonal projection of the data points (red dots) onto this subspace maximizes the variance of the projected points (green dots)



Principal Component Analysis – (4)

Alternative definition of PCA: minimize the sum of squares of the projection errors (blue lines)



Principal Component Analysis – (5)

* PCA steps
  - Evaluate the mean
  - and the covariance matrix of the data set
  - Find M eigenvectors of the covariance matrix that correspond to M largest eigenvalues
* What are the eigenvalues and eigenvectors?

Principal Component Analysis – (6)

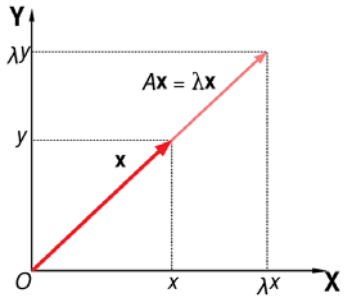
* Eigenvalue and eigenvector
  - Given a linear transformation $\boldsymbol{A}$, a non-zero vector $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}$ if it satisfies the eigenvalue equation $\mathbf{Ax} = \lambda \boldsymbol{x}$ for some scalar λ
  - The scalar λ is called eigenvalue of $\boldsymbol{A}$ corresponding to the eigenvector $\boldsymbol{x}$

(source: http://en.wikipedia.org/wiki/Eigenvector)

Principal Component Analysis – (7)

Geometrically the eigenvalue equation means that under the transformation $\boldsymbol{A}$ eigenvectors do not change their direction.

The eigenvalue λ is simply the amount of "stretch" or "shrink" to which a vector is subjected when transformed by $\boldsymbol{A}$.



(source: http://en.wikipedia.org/wiki/Eigenvector)

Principal Component Analysis
Applications – (1)

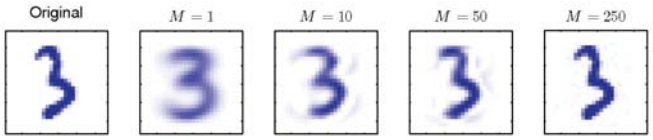
* PCA approximation to a data vector $\boldsymbol{x_n}$

$$\tilde{x}_n = \bar{x} + \sum_{i=1}^{M} \left(x_n^T u_i - \bar{x}^T u_i \right) u_i$$

  - where $\bar{x}$ – mean of the data set,
  - u_i – eigenvectors of the covariance matrix for the original data set $\{x_n\}$
  - M – number of principal components

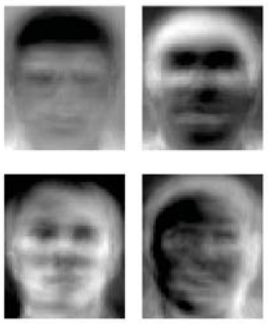
Principal Component Analysis
Applications – (2)

* Example: PCA reconstruction obtained by retaining $\boldsymbol{M}$ principal components
  - As M increases, the reconstruction becomes more accurate
  - It becomes perfect when $M = D = 28\text{x}28 = 784$



Principal Component Analysis
Applications – (3)

* Example: PCA for human face recognition
  - Eigenfaces – eigenvectors used for human face recognition
  - Obtained by PCA applied to a set of images of human faces



(source: http://en.wikipedia.org/wiki/Eigenvector)

Summary and conclusions – (1)

* Machine learning
  - Supervised learning: labeled data
    - *Goal: Find a decision boundary between classes*
    - *Classification: K-Nearest Neigbours (kNN), Support Vector Machine (SVM), Random Forest*
  - Unsupervised learning: unlabeled data
    - *Goal: Discover groups of similar data points*
    - *Clustering: K-means, Mixture Models*
    - *Latent variable models: PCA*

Summary and conclusions – (2)

* k-Nearest Neigbhours
  - For a new data point: assign most common class among k nearest neighbours in feature space
  - Instance learning: examples are stored
* Support Vector Machine (SVM)
  - Optimization problem that finds the hyperplane with maximum margin between two classes
  - Fast learning and very fast classification
  - Memory-efficient: only support vectors are stored

Summary and conclusions – (3)

* Random Forest
  - Good generalization due to randomness model
  - Multi-class classification with probablistic output
* K-means clustering
  - can be used for segmentation, unsupervised classification
  - simple, easy to implement
  - assigns one data point to one cluster, no soft assignment

Summary and conclusions – (4)

* Mixture of Gaussians
  - useful for modeling probability densities, allows flexibility necessary for it
* PCA
  - provides principal components for a data set
  - successfully used for dimensionality reduction (see eigenfaces)
  - assumes high signal-to-noise ratio
    (large variance => important dynamics)

References

- Christopher M. Bishop, “Pattern Recognition and Machine Learning”, Springer, 2002
  - Chapter 7
  - Chapter 9
  - Chapter 12
- A. Criminisi, “Decision Forests for Computer Vision and Medical Image Analysis” , Springer, 2013
  - Chapter 3
  - Chapter 4