UiO : Department of Informatics
University of Oslo

Biologically inspired computing – Lecture 8

Support Vector Machines

Ensembles

Dimensionality

This lecture

- Support vector machines
  - Optimal separation
  - Kernels

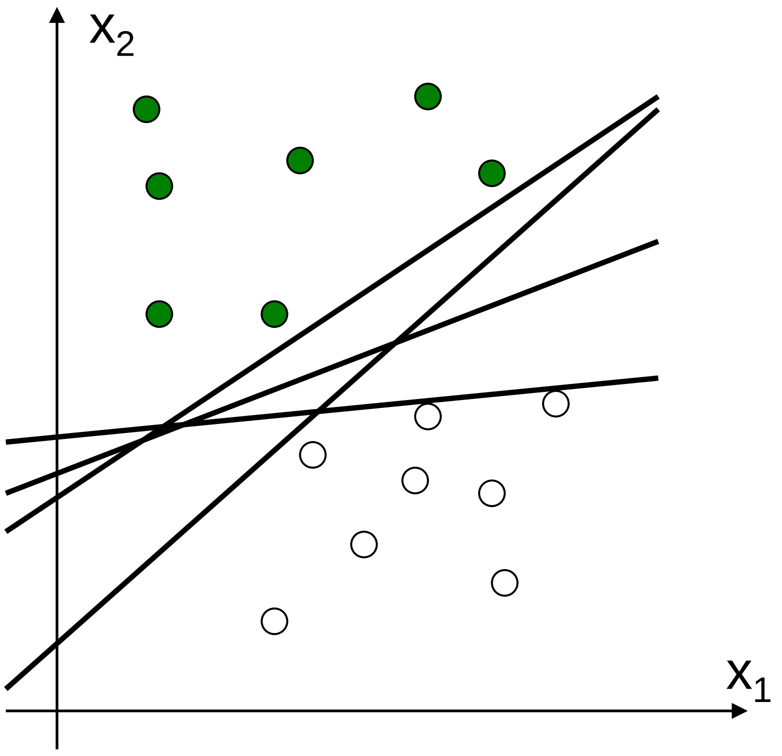
- Ensemble learning

- Dimensionality reduction
  - Principal component analysis

Optimal separation

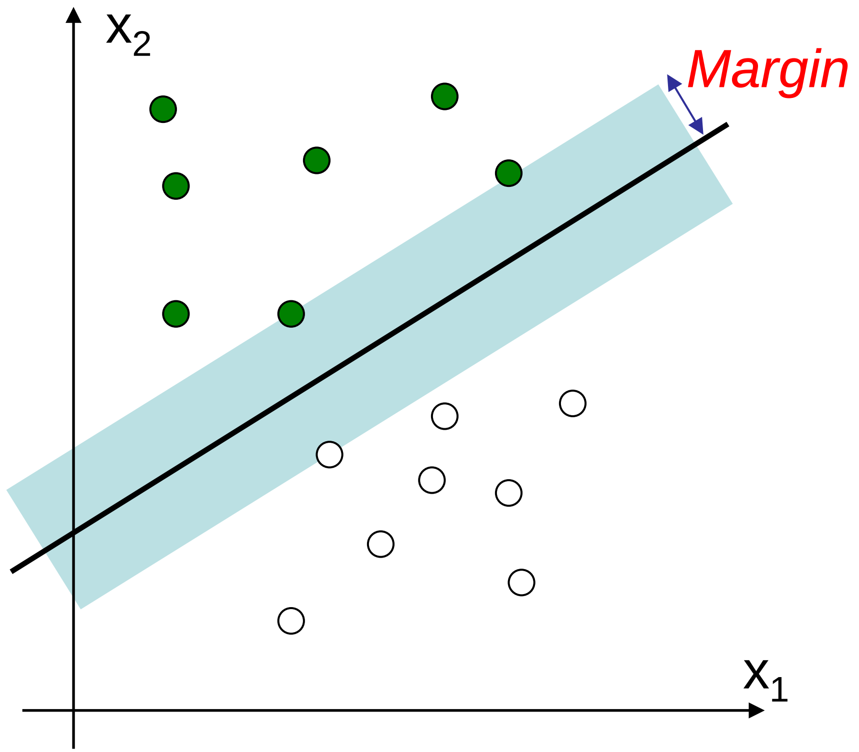
Linear separators:

Which one is best?



Optimal separation

Choose the one with the best margin!



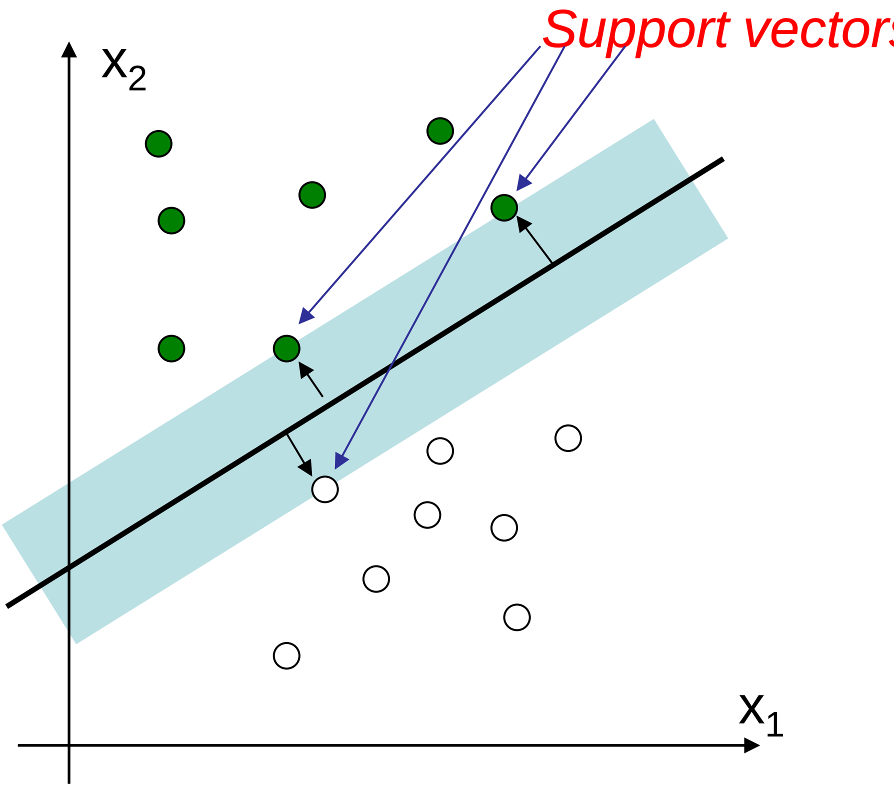
Why?

- New data near the training data points will likely be of the same class

Optimal separation

Support vectors

- The training data defining the margin
- The rest of the data can be discarded when we are done learning



Optimal separation

- Distance to hyperplane:

$$\boldsymbol{w} \cdot \boldsymbol{x}_i - b = \begin{cases} > 0 & \text{above plane} - \text{class A:} \quad y_i = 1 \\ < 0 & \text{below plane} - \text{class B:} \quad y_i = -1 \end{cases}$$

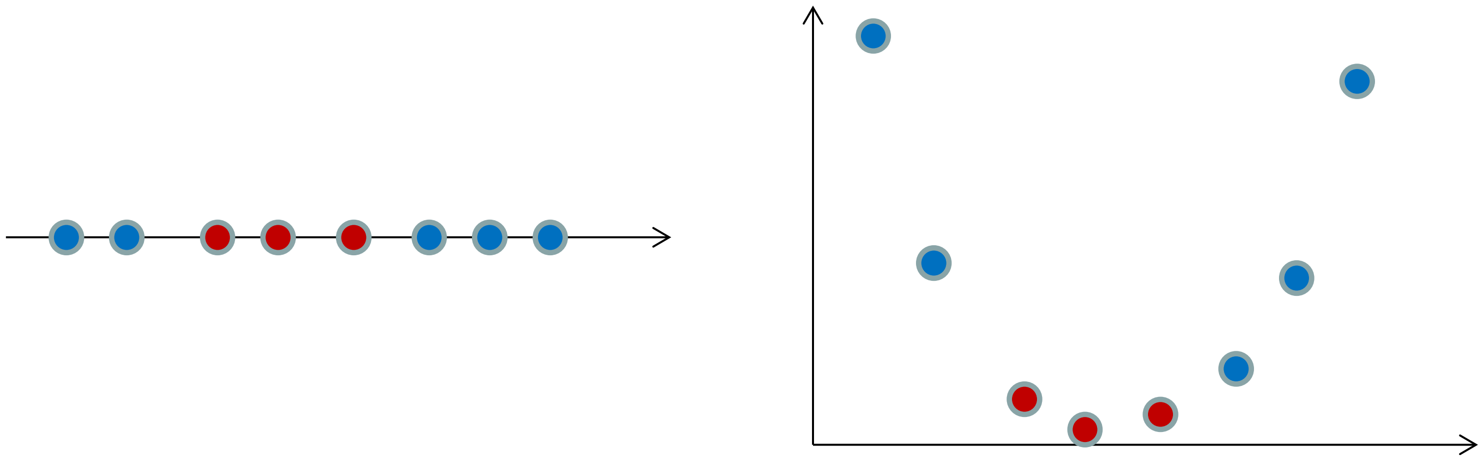
- If we require that $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - b) \geq 1$ then the margin is $M = 1/(2|\boldsymbol{w}|)$
  - Maximizing the margin ⇔ minimizing $\boldsymbol{w} \cdot \boldsymbol{w}$
  - Exact solution can be found, along with a list of support vectors, using quadratic programming

Nonlinearity

- How to classify linearly inseparable data?
  - Combine many linear SVMs?
    - Similar to multilayer neural networks
    - But what are the target outputs for the hidden layers?
  - A different idea:
    - Map inputs into a higher-dimensional space
    - Hope that they are linearly separable there.

Increase dimensionality

$$\varphi: (x) \rightarrow (x, x^2)$$



High dimensionality

- SVMs typically map to feature spaces of much higher dimension
  - With enough dimensions, it becomes very likely that the data becomes linearly separable

Kernels

- Finding the hyperplane only requires the dot product between vectors, not the actual vectors
  - Calculating $\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$ might be much easier than $\varphi(\boldsymbol{x}_i)$!

- $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$ is called the *kernel* of φ
  - Common kernels include
    - None: $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$
    - Polynomial: $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i \cdot \boldsymbol{x}_j)^p$
    - Sigmoid: $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\kappa \boldsymbol{x}_i \cdot \boldsymbol{x}_j - \delta)$
    - Radial basis function: $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^2 / 2\sigma^2\right)$

Overfitting

- Any data set is linearly separable in a feature space of sufficient complexity
- We have to be wary of overfitting: Use cross-validation and early stopping!
  - If there are noisy outliers (esp. mislabeled examples), we need to take stronger measures: soft margin.

Soft margins

- Instead of perfectly separating all data, allow some misclassifications
- Introduce slack variables
  - Optimize tradeoff between maximum margin and misclassification penalty
  - Tradeoff is balanced by penalty factor C
- Useful when some error is tolerated, or when there are chances of mislabeled training data

Applications

- Classification
  - Multi-class can be achieved via multiple outputs (1v1 or 1vMany)
- Regression
- Supervised and unsupervised learning
- Object detection & recognition
- Content-based image retrieval
- Text recognition
- Speech recognition
- Biometrics
- Etc.

Considerations

- Quite powerful
  - Must beware of overfitting
- Robust to some noise, if margin is managed properly
- Fast to apply
- Difficult to interpret

- How to pick kernel?
  - Start with Gaussian RBF or polynomial
  - May require domain-specific knowledge
  - Can combine kernels for heterogeneous data
  - Consult experts

Ensemble learning

- "Decision by committee"
  - Train multiple classifiers to be slightly different
    - An "ensemble"
  - Make classifications based on the combined results of all of them
- Two common types of training differentiation
  - Boosting: change the importance of data points
  - Bagging: change the data sample

Boosting - AdaBoost

- Iteratively trains classifiers
- Each data point is assigned a weight
  - For the first classifier all the weights are equal
  - For the next classifier the weights of the data points that were misclassified previously is raised
  - This is continued until the combined error of the classifiers trained so far is sufficiently low
- Dependent on the classifier's ability to consider the weights in their training

Bagging

- Makes a random sample of the training data for each classifier – *bootstrap samples*
  - Same size as the training data
  - *With replacement*
  - Some data points will occur at least twice!
  - Variance will be reduced
  - Each classifier will have different views of the training data

Combining the classifiers

- Which classifiers do we listen to when the ensemble is in disagreement?
  - Majority voting
    - The most "popular" class is chosen
  - Weighted voting
    - Some classifiers have greater influence than others
  - Mixture of experts
    - A meta-machine learning algorithm decides which classifiers are most likely to be correct

Majority voting

What to do when there is disagreement

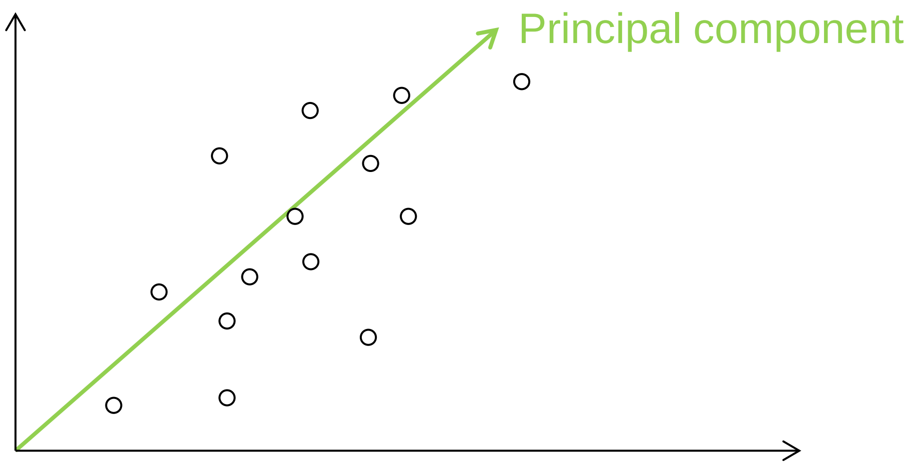
- Refuse to classify?
- Classify only if more than half agree?
- Return the most common vote?

Dimensionality reduction – Why reduce dimensionality?

- Reduces time complexity: Less computation
- Reduces space complexity: Less parameters
- Saves the cost of acquiring irrelevant features
- Simpler models are more robust
- Easier to interpret; simpler explanation
- Data visualization (structure, groups, outliers, etc.) if plotted in 2 or 3 dimensions

Principal components

- The directions along with the most variation
  - Don't have to correspond to the coordinate axes



Principal component analysis

- Rotate the axes to lie along the principal components

- Remove the axes with the least variation
  - Keep a certain number of dimensions
  - Or: keep a certain percentage of the variation

Calculating the principal components

- Calculate the covariance matrix of the data
- Calculate the eigenvalues and eigenvectors of the covariance matrix
- Transform the data with the eigenvectors for the largest eigenvalues as the new basis

Calculating the covariance matrix

The variance of feature i:

$$\sigma_i^2 = \sigma_{ii} = \frac{1}{N}\sum_{k=1}^{N}(x_{ki} - \mu_i)^2$$

The covariance between feature i and j:

$$\sigma_{ij} = \frac{1}{N}\sum_{k=1}^{N}(x_{ki} - \mu_i)\left(x_{kj} - \mu_j\right)$$

Calculating the covariance matrix

The covariance matrix is composed of the variances and covariances of every combination of feature:

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix}$$

The covariance eigenvectors

The eigenvectors $\boldsymbol{v}_i$ and eigenvalues λ_i are the n unique values of matrix C such that

$$\lambda_i \boldsymbol{v}_i = C\boldsymbol{v}_i$$

- The eigenvectors of the covariance matrix describe the directions of the principal components
- The eigenvalues tell us how large part of the total variation in the data that is accounted for by that principal component

Notes on PCA

- PCA is a linear transformation
  - Does not directly help with data that is not linearly separable
  - However, may make learning easier because of reduced complexity
- PCA removes some information from the data
  - Might just be noise
  - Might provide helpful nuances that may be of help to some classifiers