UiO : University of Oslo

Support Vector Machines + Classification for IR

Pierre Lison
University of Oslo, Dep. of Informatics

INF3800: Søketeknologi
April 30, 2014

Outline of the lecture

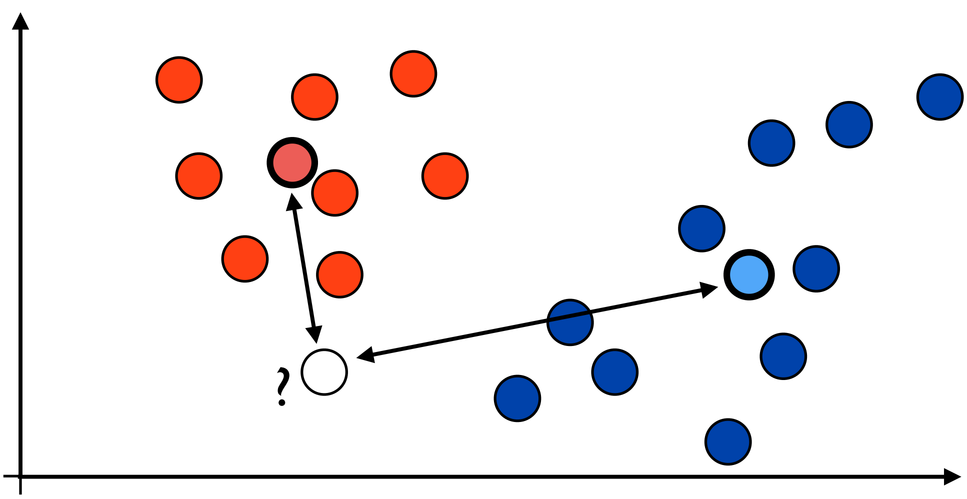
- Recap' of last week
- Support Vector Machines
- Classification for IR
  - Practical aspects
  - Relevance ranking
- Conclusion

Outline of the lecture

- Recap' of last week
- Support Vector Machines
- Classification for IR
  - Practical aspects
  - Relevance ranking
- Conclusion

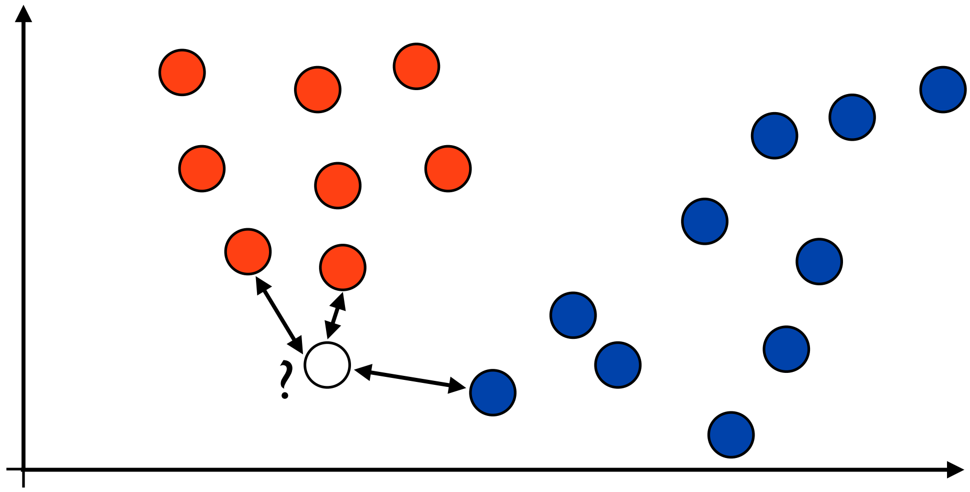
Rocchio classification

- Finds the "center of mass" for each class
- A new point x will be classified in class c if it is closest to the centroid for c



k-nearest neighbour (k-NN)

- k-NN adopts a different approach
  - Rely on *local* decisions based on the closest neighbors
  - k = number of neighbours to consider



Linear vs. non-linear classification

	Linear classifier	Non-linear classifier
Function	Linear combination of features: $y = f(\mathbf{w}^T\mathbf{x})$.	Arbitrary non-linear function
Decision boundary	Hyperplane	Non-linear, possibly discontinuous
Examples	Naive Bayes, Rocchio, logistic regression, linear SVMs	k-NN, multilayer neural networks, non-linear SVMs
Pros	Often robust, fast	Can express complex dependencies
Cons	Can fail if problem is not linearly separable	Prone to overfitting

Bias-variance trade-off

- The learning error of a classification method is the expectation (averaged) over the possible training sets:

$$\text{learning-error}(\Gamma) = E_{\mathbb{D}}\left[E_x[\gamma(x) - P(c|x)]^2\right]$$
$$\ldots$$
$$= E_x[\underbrace{\text{bias}(\Gamma, x)}_{\text{how often the classifier prediction deviates from the “true” class}} + \underbrace{\text{variance}(\Gamma, x)}_{\text{amount of variation in the classifier prediction depending on the training data}}]$$

how often the classifier prediction deviates from the “true” class

amount of *variation* in the classifier prediction depending on the training data

Outline of the lecture

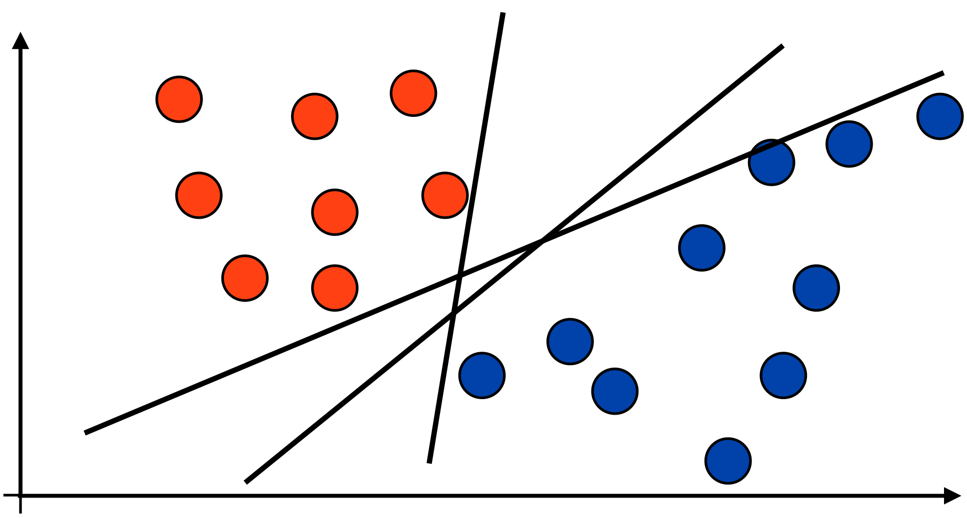
- Recap' of last week
- **Support Vector Machines**
- Classification for IR
  - Practical aspects
  - Relevance ranking
- Conclusion

Support Vector Machines

- State-of-the-art classification method
  - Both linear and non-linear variants
  - Non-parametric & discriminative
  - Good generalisation properties (quite resistant to overfitting)
  - Extensions for multiclass classification, structured prediction, regression

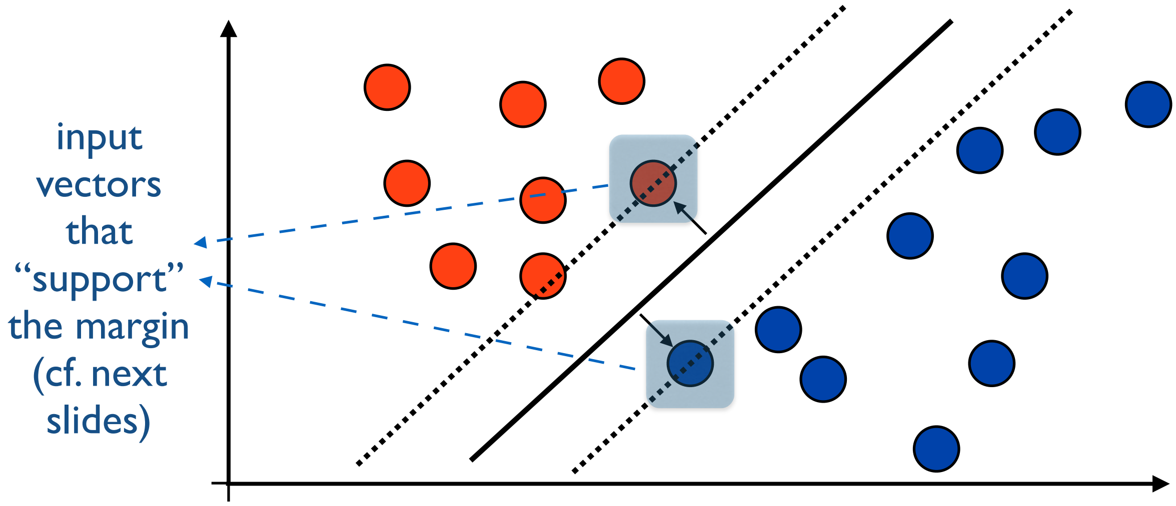
SVMs: linear case

- We have seen that linear classifiers (such as Rocchio, NB, etc.) create a hyperplane separating the classes
- But there is an infinite number of possible hyperplanes!



SVMs: linear case

- Are some hyperplanes better than others?
  - A "good" boundary should be as far as possible from the training points
  - SVM idea: find the hyperplanes with a *maximum margin* of separation



UiO : University of Oslo

SVMs: linear case

- Why maximise the margin?
  - Points close to the boundary are very uncertain!
  - Small margins between the boundary and the training points make the classifier very sensitive to variations in the training data (= high variance, cf. last week)
  - … and we want to keep the variance as low as possible to ensure the classifier generalizes well to new data (= is resistant to overfitting)

SVMs: linear case

- Classification function:

bias term

Classification output: {+1,-1}

$$f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} + b\right)$$

Input vector

Weight vector

$$\text{sign}(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x = 0 \\ \text{-1 if } x < 0 \end{cases}$$

- Training data $\mathbb{D}$ is composed of *n* examples $\{(\mathbf{x}_i, y_i) : 1 \leq i \leq n\}$, where $y_i = \{+1 -1\}$

Note: Math conventions for SVMs slightly different than for other classification techniques

SVMs: linear case

- The goal is to find the values for the vector **w** and bias *b* that will maximize the margin
- It can be shown (cf. textbook) that this goal is equivalent to the following problem:

Find **w** and *b* such that:
- $\mathbf{w}^T\mathbf{w}/2$ is minimized
- for all $\{(\mathbf{x}_i, y_i)\}$, $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Quadratic optimisation problem!

SVMs: linear case

- Many algorithms are available for solving such class of optimisation problems
- The resulting weight vector can be expressed as a combination of the training points:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

Lagrange multipliers (determined during the optimisation)

Most α_i will be zero. The points ($\mathbf{x}_i$,y) which have a non-zero α_i are precisely the **support vectors** for the margin!

[the bias term b can be inferred from the weight vector, cf. textbook]

SVMs: linear case

- The classification function for the SVM can be rewritten as:

$$f(\mathbf{x}) = \text{sign}\left(\sum_i \alpha y_i \mathbf{x}_i^T \mathbf{x} + b\right)$$

- Combination of dot products between the vector **x** to classify and the vectors $\mathbf{x}_i$ from the training data
- Multipliers α_i calculated by solving the optimization problem
- Only the vectors with $\alpha_i \neq 0$ (the support vectors) need to be considered in the classification!

SVMs with soft margin

- Real-world classification problems are not always 100% linearly separable
  - **Causes**: Noise in the data set, outliers, etc.
- Extension of SVMs with a "soft" margin
  - Allows a few training points to be misclassified
  - But the misclassification of each point has a cost!

UiO : University of Oslo

SVMs with soft margin

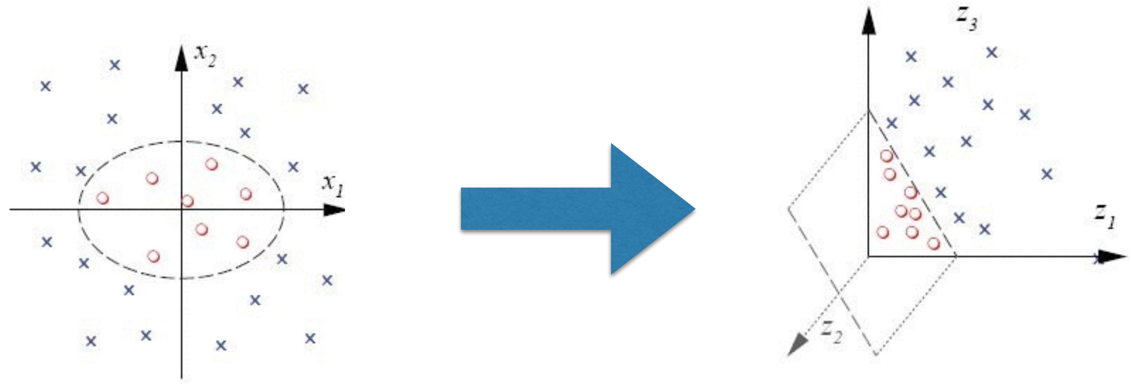
- Idea: introduce *slack variables* ξ_i
  - The slack variable ξ_i measures the degree of misclassification of the point $\mathbf{x}_i$
- Corresponding optimisation problem:

> Find **w** and *b* such that:
> - $\mathbf{w}^T\mathbf{w}/2 + C \sum \xi_i$ is minimized
> - for all $\{(\mathbf{x}_i, y_i)\}$, $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$

C is a parameter that controls the "softness" of the margin

Non-linear SVMs

- The algorithms presented so far are purely linear
- But SVMs can also solve non-linear problems!
  - **Key idea**: map each point from the initial input space into a *higher-dimensional space* in which the training data is linearly separable
  - … and do linear classification in this hyperspace



Non-linear SVMs

- How is this mapping performed?
- First idea: Create a mapping function Φ(**x**) from the original space to the hyperspace, and rewrite the classifier as

$$f(\mathbf{x}) = \text{sign}\left(\sum_i \alpha y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b\right)$$

- Problem: this is not very efficient!
  - Need to perform the mapping for every point

Non-linear SVMs

- **Kernel trick**: replace the dot product $\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x})$ by a *kernel function* $K(\mathbf{x}_i, \mathbf{x})$
  - No need to use (or even specify) a mapping function Φ!
  - Numerous kernels can be employed
- Classifier becomes:

$$f(\mathbf{x}) = \text{sign}\left(\sum_i \alpha y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

- High-dimensional space "embedded" by the kernel (resulting space may even be infinite-dimensional!)

UiO : University of Oslo

Non-linear SVMs

- The kernel function must satisfy some properties (be continuous, symmetric, and positive definite)
- Popular kernels:
  - Polynomial kernels: $K(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i^T\mathbf{x}_j + 1)^d$
  - Gaussian kernels: $K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-(x_i-x_j)^2/2\sigma^2)$
  - String and tree kernels for NLP tasks
- Need to find the most appropriate kernel to use for a given classification task

Outline of the lecture

- Recap' of last week
- Support Vector Machines
- **Classification for IR**
  - **Practical aspects**
  - **Relevance ranking**
- Conclusion

Classification: Practical aspects

- How to practically choose a classifier?
- Key question: *where can I get data (and how much)*?

No data:
design hand-crafted rules

Fairly little data:
high-bias (e.g. linear) classifier

Tons of (good) data:
Log. regression or SVMS often good choice

- *Computational complexity* also an important factor
- Classifiers can be combined ("ensemble learning")

Classification: Practical aspects

- Assembling data resources is often the real bottleneck in classification
  - Collect, store, organise, quality-check the data
  - Financial and legal aspects (ownership, privacy)
- ML-based classifiers must sometimes be overlaid with hand-crafted rules
  - To enforce particular business rules, or allow the user to control the classification

Classification: Practical aspects

- Which features to use?
  - Designing the right features is often key to success
  - If too few features: not informative enough
  - If too many features: data sparseness!
- In text classification, the most basic features are the document terms:
  - But preprocessing is important to filter/modify some tokens
  - Other features, such as document length, zones, links, etc.

Relevance ranking

- Many tasks in information retrieval are classification problems
  - Document preprocessing (segmentation etc.)
  - Determining whether a document is relevant or not
- Simple way to calculate the relevance of a document d to a query q:
  - Extract features from (d,q), such as cosine score, proximity window ω, static quality, document age, etc.
  - Two categories: relevant or non-relevant

Relevance ranking

- But classification as relevant/non-relevant is a crude way to solve the problem
  - What we want is to *rank* the relevance of documents
- Ranking is an *ordinal regression* problem
  - The exact "score" of each document is not important, what counts is the relative ordering
  - Midway between classification and regression

Relevance ranking

- SVMs can be applied on ranking problems
  - We first collect training data, where each query q is mapped to a list of documents ordered by relevance
  - To build the classifier, we construct a feature vector ψ for each document/ query pair (d_i, q)
  - Then create a vector $\Phi(d_i,d_j,q)$ of feature differences: $\Phi(d_i,d_j,q) = \psi(d_i,q) - \psi(d_j,q)$
  - Finally, we can build a classifier on this vector:

$$\mathbf{w}^T \Phi(d_i, d_j, q) > 0 \quad \text{iff } d_i \text{ precedes } d_j$$

Outline of the lecture

- Recap' of last week
- Support Vector Machines
- Classification for IR
  - Practical aspects
  - Relevance ranking
- Conclusion

Conclusion

- *Support Vector Machines* constitute a powerful classification method
    - *Maximum-margin* classifiers, solved by quadratic programming
    - *Slack variables* to allow for "softer" margins
    - *Kernel functions* for capture non-linear problems
- *Data collection* and *feature engineering* are crucial questions to build practical classifiers
- *Ranking* classifiers can be employed to order documents by order of relevance to a query