UiO : University of Oslo

Classification with vector space models

Pierre Lison
University of Oslo, Dep. of Informatics

INF3800: Søketeknologi
April 23, 2014

Outline of the lecture

- Recap' of last week
- Classification in vector space
  - Rocchio
  - k-nearest neighbours
- Analysis of classifiers
- Conclusion

Outline of the lecture

- Recap' of last week
- Classification in vector space
  - Rocchio
  - k-nearest neighbours
- Analysis of classifiers
- Conclusion

Classification

- Space $\mathbb{X}$ of possible inputs
- Set $\mathbb{C} = \{c_1, c_2, \ldots c_J\}$ of possible classes or categories
- Classifier γ maps inputs to categories:

$$\gamma : \mathbb{X} \rightarrow \mathbb{C}$$

Goal: estimate this classifier with a training set $\mathbb{D}$ composed of *n* examples $\{(\mathbf{x}_i, c_i) : 1 \leq i \leq n\}$

where $\mathbf{x}_i$ is the *i*th example and c_i its category

Formalisation

- The inputs **x** ∈ $\mathbb{X}$ are often represented as feature vectors:
  - Each vector position corresponds to a particular feature, with a discrete or continuous range
  - For instance, a "bag-of-words" representation of a document may be encoded as a vector

$[\, w_1,\; w_2,\; w_3 \ldots, w_N]^T$ where w_i is the number of occurrences of the term i

- The categories c ∈ $\mathbb{C}$ must be discrete labels

Naive Bayes

- The classification of a document d is a search for the class c* such that

$$c^* = \underset{c}{\operatorname{argmax}}\, P(c|d)$$

- But the probability P(c|d) is hard to determine!
- Using Bayes rule, we can rewrite the probability:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

prior of the class

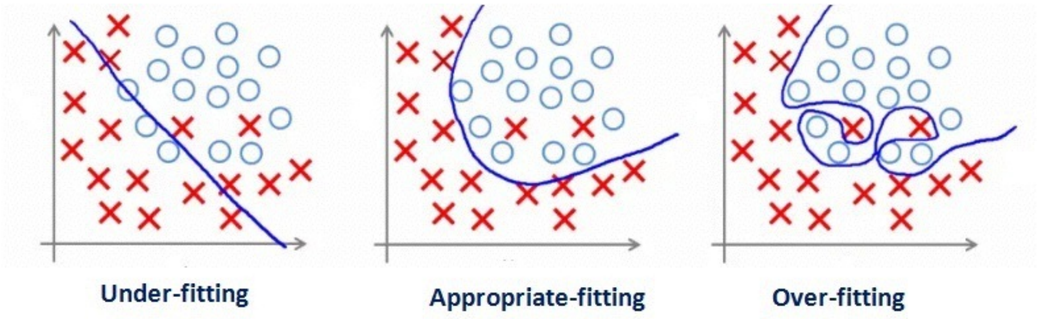
normalisation factor (often ignored)

posterior probability of the class given the document

likelihood of the document given the class

Evaluation

- High accuracy on training examples does not necessarily translate into good results on new data
  - Phenomenon of "overfitting"
  - (especially for high-dimensional spaces and/or non-linear models)



Under-fitting

Appropriate-fitting

Over-fitting

(too simple to explain the variance)

(forcefitting -- too good to be true)

Evaluation metrics

Confusion matrix on test set:

		Gold standard	
		Positive	**Negative**
Predicted by classifier	**positive**	True positive (tp)	False positive (fp)
	Negative	False negative (fn)	True negative (tn)

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Outline of the lecture

- Recap' of last week
- Classification in vector space
  - Rocchio
  - k-nearest neighbours
- Analysis of classifiers
- Conclusion

Classification in vector space

- Last week, the input documents were encoded as feature vectors:

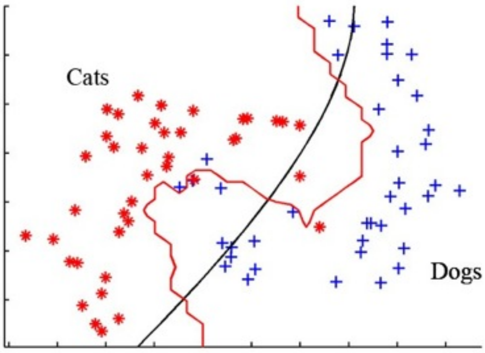
  $[w_1, w_2, \ldots w_N]$ where w_i is the presence/absence (or number of occurrences) of term *i* in the document

- This week, we adopt a representation for the feature vectors based on the *vector space model*:

  $[w_1, w_2, \ldots w_N]$ where w_i is a (real-valued) TF-IDF weight for the term i

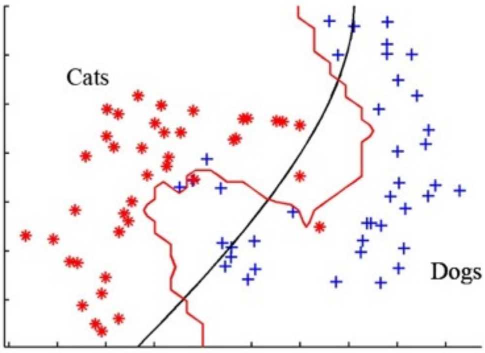
Classification in vector space

- The goal remains the same
  - We want to build a classifier $\gamma : \mathbb{X} \to \mathbb{C}$
  - And we construct this classifier on the basis of a training set $\mathbb{D}$ of n examples $\{(\mathbf{x}_i, c_i) : 1 \leq i \leq n\}$
- Each input **x** is here a vector $\in \Re^N$



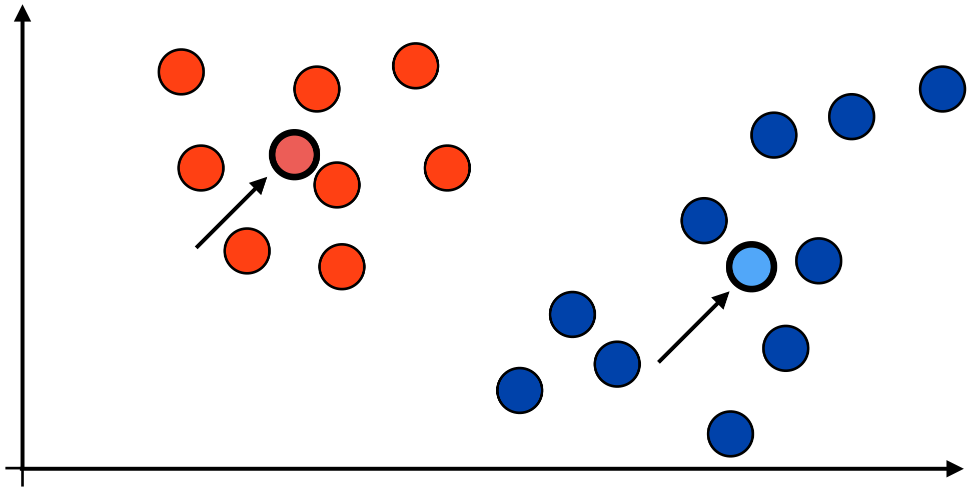
Classification in vector space

- We cover two new classifiers today:
  - Rocchio classification
  - k-nearest neighbours classification
- They rely on a notion of *distance* between points



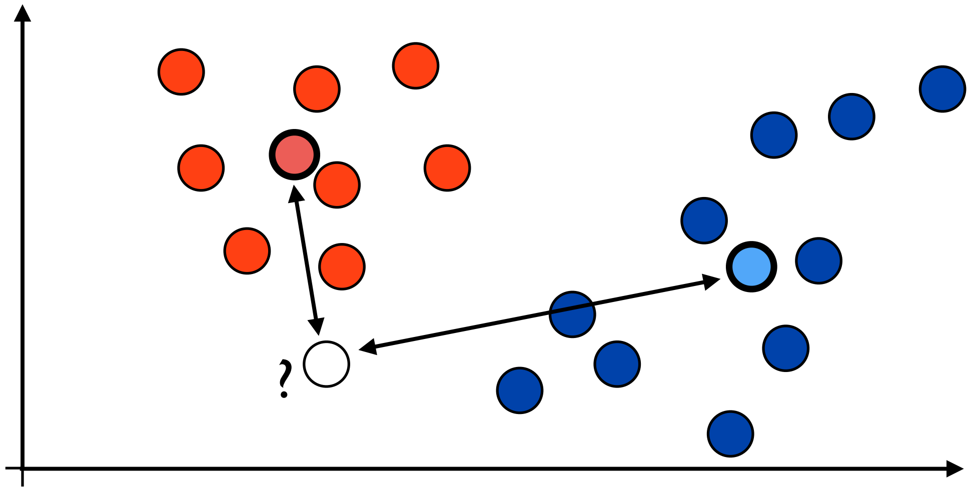
Rocchio classification

- Finds the "center of mass" for each class
- Centroids = "prototypical" examples



Rocchio classification

- A new point x will be classified in class c if it is closest to the centroid for c



Rocchio classification

- The centroid for class c is defined as:

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

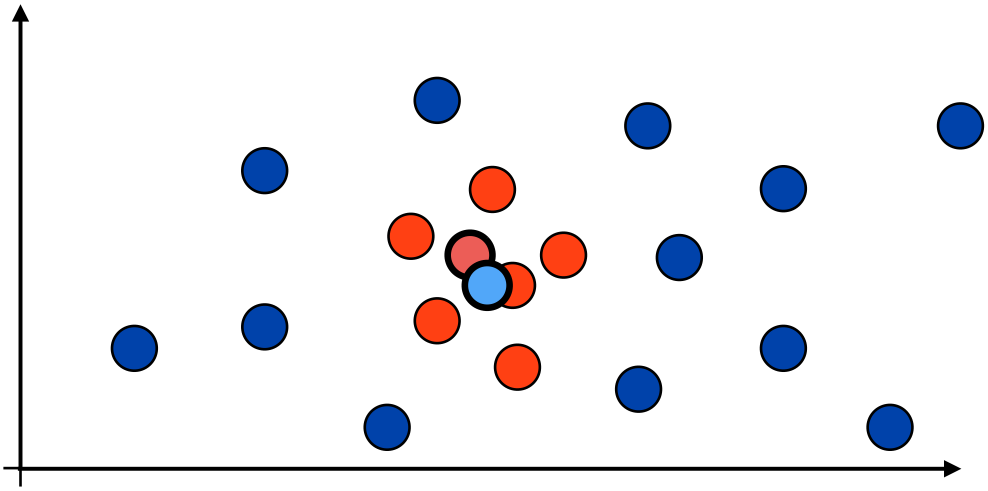
- Once the centroids are calculated, we can classify a new input d as:

$$c^* = \underset{c}{\operatorname{argmin}} |\vec{\mu}(c) - \vec{v}(d)|$$

(according to some distance metric)

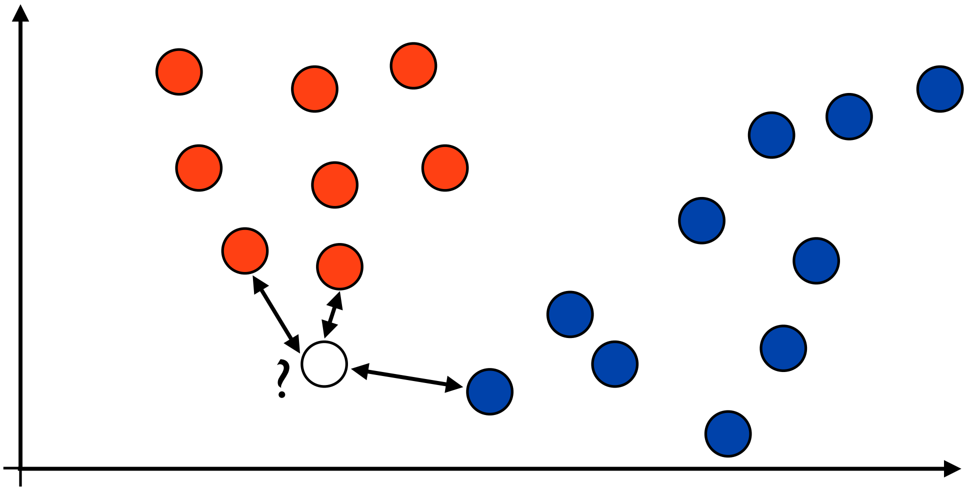
Limitations of Rocchio

- Fails to deal with non-contiguous regions
- Rocchio assumes that the classes correspond to spheres of equal radii



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

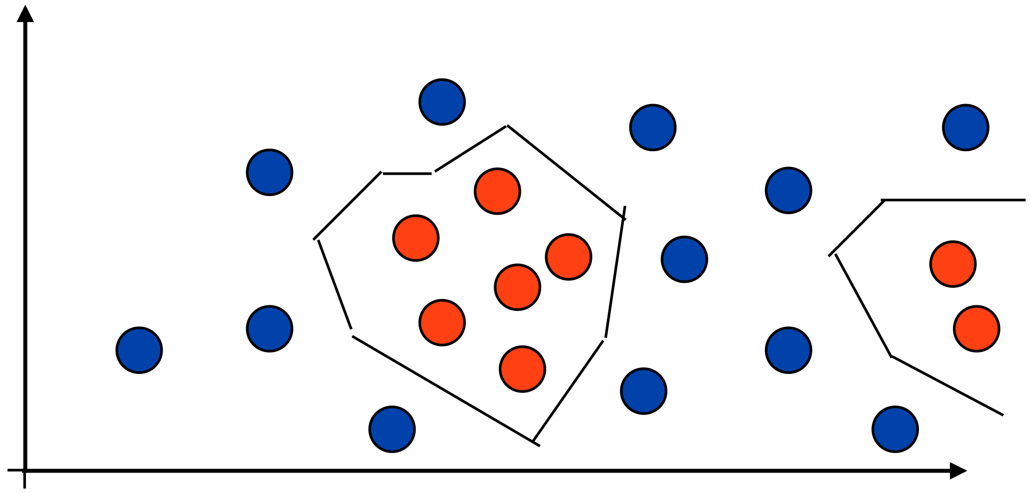


k-nearest neighbour

- Decision steps (for a input *x*)
  - Find the set S_k of *k* points closest to x
  - The probability P(c|x) is then the proportion of neighbours in S_k that belong to c
- We can also weight the votes of each neighbour according to their cosine similarity (or other measures)

k-nearest neighbour

- Complex, non-linear decision boundaries
  - Technically speaking: k-NN defines a *Voronoi tessellation*



k-nearest neighbour

- Lazy learning: no actual training required!
  - (apart from extracting the features for each point)
  - "Instance-based" learning
  - But need to store all data points
- Testing can be expensive:
  - Need to consider a large number of points
  - But we can use an inverted index to filter the points to consider as possible neighbours

Outline of the lecture

- Recap' of last week
- Classification in vector space
  - Rocchio
  - k-nearest neighbours
- Analysis of classifiers
- Conclusion

Analysis of classifiers

- We already covered 3 classification algorithms: Naive Bayes, Rocchio, k-NN
  - Next week: Support Vector Machines
- How to decide which algorithm is "better" for a particular task?
  - Ultimately, only a empirical evaluation will give us the answer
  - But we can gain some insight into the *theoretical properties* of each algorithm (what they can and cannot do)

Dimension of analysis

- What is the form of the decision boundary (linear, non-linear)?
- How is the classification function encoded (parametric or non-parametric form?)
- Is the model generative or discriminative?
    - **Generative**: estimate P(c|x) via P(x|c) and P(c)
    - **Discriminative**: estimate P(c|x) directly

Linear vs. non-linear classification

- A linear classifier defines a hyperplane in the input space $\mathbb{X}$
    - The hyperplane should separate the classes
    - If a hyperplane can be found with a perfect separation of classes, we say the problem is *linearly separable*
- See textbook for proofs that NB and Rocchio are linear classifiers

Linear vs. non-linear classification

- Non-linear not necessarily more "powerful" than linear
  - Ex: a linear model with millions of features may be more powerful than a non-linear with a few dozens
- What is best in practice?
  - If problem is (more or less) linearly separable, linear classifiers often scale better
  - If problem is highly non-linear, non-linear classifiers are often more accurate

Linear vs. non-linear classification

	Linear classifier	Non-linear classifier
Function	Linear combination of features: $y = f(\mathbf{w}^{\mathrm{T}}\mathbf{x})$.	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

Parametric vs. non-parametric

- One can also analyse classifiers in terms of the form of their classification function
  - *Parametric* classifiers are represented by a fixed set of parameters θ, and use the training data to estimate the best values for these parameters θ
  - *Non-parametric* classifiers do no rely on any parametric form, but directly operate on the training data
- Non-parametric classifiers make less assumptions about the problem

Parametric vs. non-parametric

	Parametric classifier	**Non-parametric classifier**
Type of model	classifier with fixed set of parameters, use data to estimate their values	Use training data itself as "model"
Examples	Naive Bayes, Rocchio, logistic regression	k-NN, SVMs
Pros	Compact encoding, excellent results if model is appropriate	Flexible (naturally adapts itself to the data)
Cons	Rigidity (must stick to the given model structure)	Prior knowledge difficult to include; expensive in memory & CPU

Generative vs. discriminative

- Finally, classifiers may be analysed in terms of how they estimate P(c|x)
  - **Generative** classifiers estimate P(c|x) indirectly:

$$P(c|x) \propto \underbrace{P(x|c)}_{\textit{likelihood}\text{ of the input given the class}} \overbrace{P(c)}^{\textit{prior}\text{ of the class}}$$

"generative" because they can generate new data points

  - **Discriminative** classifiers estimate P(c|x) directly

Generative vs. discriminative

	Generative model	Discriminative model
Type of model	Estimates both the likelihood P(x|c) and the prior P(c)	Directly estimates the posterior P(c|x)
Examples	Naive Bayes, many graphical models	Logistic regression, SVMs, k-NNs
Pros	Explanatory power	Usually more accurate when lots of data is available
Cons	Tries to model "more" than is necessary for the task	Problems with missing data

Bias-variance trade-off

- The *bias-variance trade-off* provides useful insights on the theoretical properties of classification algorithms
- One important quality metric in classification is the *mean square error*:

$$\underbrace{\mathrm{MSE}(\gamma)}_{\text{mean square error for classifier}} = \underbrace{E_x}_{\text{expectation for input x}}[\underbrace{\gamma(x)}_{\text{classification output}} - \underbrace{P(c|x)}_{\text{“true” prob. of class c}}]^2$$

NB: other error metrics can be devised

Bias-variance trade-off

- MSE measures the error of a particular classifier
  - But we are interested in evaluating l*earning methods*
- The learning error is the expectation (averaged) over the possible training sets:

$$\begin{aligned}\text{learning-error}(\Gamma) &= E_{\mathbb{D}}\left[E_x[\gamma(x) - P(c|x)]^2\right] \\ &\dots \\ &= E_x[\text{bias}(\Gamma, x) + \text{variance}(\Gamma, x)]\end{aligned}$$

with

$$\begin{aligned}\text{bias}(\Gamma, x) &= [E_{\mathbb{D}}[\Gamma_{\mathbb{D}}(x)] - P(c|x)]^2 \\ \text{variance}(\Gamma, x) &= E_{\mathbb{D}}[\Gamma_{\mathbb{D}}(x) - E_{\mathbb{D}}[\Gamma_{\mathbb{D}}(x)]]^2\end{aligned}$$

(details in the textbook, but you don't need to memorise them)

Bias-variance trade-off

- The **bias** represents how much the classifier prediction deviates (on average) from the “true class probability
  - Bias is large = the classifiers are consistently wrong
- The **variance** represents the amount of *variation* in the classifier prediction depending on the training data
  - Variance is large = distinct training sets may lead to very different classifiers

Bias-variance trade-off

- Bias encodes the domain knowledge (assumptions) prior to learning
  - All learning algorithms necessarily have a bias (else they would not be able to generalize to new points)
- Variance represents the sensitivity of the algorithm to variations in the data
  - High variance = prone to overfitting!

Bias-variance trade-off

- Examples:
  - Naive Bayes has a *high bias* (can only encode linear problems) but a *low variance* (small variations of the training set will not move the boundary by much)
  - k-nearest neighbour has a *low bias* (can encode complex, non-linear problems) but a *high variance* (very sensitive to noise in the training data)
- Minimising the learning error = finding the right trade-off between bias and variance

Bias-variance trade-off

- How to practically control the bias-variance trade-off
  - Reduce dimensionality (e.g. feature selection)
  - Reduce model complexity (e.g. regularisation, Bayesian priors on parameters)
  - Ensemble learning (combine several classifiers)
  - Cross-validation to test models empirically

Outline of the lecture

- Recap' of last week
- Classification in vector space
  - Rocchio
  - k-nearest neighbours
- Analysis of classifiers
- Conclusion

Conclusion

- Classification in vector space:
  - *Rocchio*: a linear classifier that maps an input x to the class whose center of mass is closest to the point
  - *k-nearest neighbour*: a non-linear classifier that maps x to the majority class of its *k* closest neighbours
- Analysis of classification algorithms:
  - Various dimensions: *Linear* vs. *non-linear*, *parametric* vs. *non-parametric*, *generative* vs. *discriminative*
  - *Bias-variance trade-off*: finding the right balance between accuracy and robustness to overfitting