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
  - \( k = \text{number of neighbours to consider} \)

![Diagram of k-NN concept]

**Linear vs. non-linear classification**

<table>
<thead>
<tr>
<th>Function</th>
<th>Linear classifier</th>
<th>Non-linear classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear combination of features: ( y = f(w^T x) )</td>
<td>Arbitrary non-linear function</td>
<td></td>
</tr>
<tr>
<td>Decision boundary</td>
<td>Hyperplane</td>
<td>Non-linear, possibly discontinuous</td>
</tr>
<tr>
<td>Examples</td>
<td>Naive Bayes, Rocchio, logistic regression, linear SVMs</td>
<td>k-NN, multilayer neural networks, non-linear SVMs</td>
</tr>
<tr>
<td>Pros</td>
<td>Often robust, fast</td>
<td>Can express complex dependencies</td>
</tr>
<tr>
<td>Cons</td>
<td>Can fail if problem is not linearly separable</td>
<td>Prone to overfitting</td>
</tr>
</tbody>
</table>
Bias-variance trade-off

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

\[
\text{learning-error}(\Gamma) = \mathbb{E}_D \left[ \mathbb{E}_x [\gamma(x) - P(c|x)]^2 \right] \\
\quad \ldots \\
\quad = \mathbb{E}_x [\text{bias}(\Gamma, x) + \text{variance}(\Gamma, x)]
\]

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

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:

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

Classification output: \{+1,-1\}

Bias term

Input vector

Weight vector

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

• Training data \( \mathcal{D} \) is composed of \( n \) examples \( \{(x_i,y_i) : 1 \leq i \leq n\} \), 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:
- \( w^Tw/2 \) is minimized
- for all \( \{(x_i,y_i)\} \), \( y_i(w^T x_i + b) \geq 1 \)

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:

\[ w = \sum_{i} \alpha_i y_i x_i \]

Lagrange multipliers (determined during the optimisation)

Most \( \alpha_i \) will be zero. The points \((x_i, y)\) which have a non-zero \( \alpha_i \) 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:

\[ f(x) = \text{sign} \left( \sum_{i} \alpha_i y_i x_i^T x + b \right) \]

• Combination of dot products between the vector \( x \) to classify and the vectors \( x_i \) from the training data

• Multipliers \( \alpha_i \) 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!

Idea: introduce slack variables $\xi_i$

• The slack variable $\xi_i$ measures the degree of misclassification of the point $x_i$

Corresponding optimisation problem:

Find $w$ and $b$ such that:

- $w^Tw/2 + C \sum \xi_i$ is minimized
- for all $\{(x_i, y_i)\}$, $y_i(w^Tx_i + b) \geq 1 - \xi_i$

$C$ 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

![Diagram showing mapping from 2D to 3D space]

Non-linear SVMs

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

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

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

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

- **Classifier becomes:**
  
  \[
  f(x) = \text{sign} \left( \sum_i \alpha y_i K(x_i, x) + b \right)
  \]

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

Non-linear SVMs

- The kernel function must satisfy some properties
  (be continuous, symmetric, and positive definite)

- **Popular kernels:**
  - Polynomial kernels: $K(x_i, x_j) = (x_i^T x_j + 1)^d$
  - Gaussian kernels: $K(x_i, x_j) = \exp(-(x_i - x_j)^2 / 2\sigma^2)$
  - 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”)
• 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

• 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 \( \omega \), 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 $\Psi$ 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:
    $$w^T \Phi(d_i, d_j, q) > 0 \text{ iff } d_i \text{ precedes } d_j$$

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