Overview

1 Recap
2 Intro vector space classification
3 Rocchio
4 kNN
5 Linear classifiers
6 > two classes
Outline

1. Recap

2. Intro vector space classification

3. Rocchio

4. kNN

5. Linear classifiers

6. > two classes
Feature selection: MI for poultry/EXPORT

Goal of feature selection: eliminate noise and useless features for better effectiveness and efficiency

\[
e_t = e_{\text{EXPORT}} = 1 \\
\begin{array}{|c|c|}
\hline
N_{11} & N_{10} \\
\hline
49 & 27,652 \\
\hline
\end{array}
\quad e_c = e_{\text{poultry}} = 1 \\
\begin{array}{|c|c|}
\hline
N_{01} & N_{00} \\
\hline
141 & 774,106 \\
\hline
\end{array}
\quad e_c = e_{\text{poultry}} = 0
\]

Plug these values into formula:

\[
I(U; C) = \frac{49}{801,948} \log_2 \frac{801,948 \cdot 49}{(49+27,652)(49+141)} \\
\quad + \frac{141}{801,948} \log_2 \frac{801,948 \cdot 141}{(141+774,106)(49+141)} \\
\quad + \frac{27,652}{801,948} \log_2 \frac{801,948 \cdot 27,652}{(49+27,652)(27,652+774,106)} \\
\quad + \frac{774,106}{801,948} \log_2 \frac{801,948 \cdot 774,106}{(141+774,106)(27,652+774,106)}
\]

\[
\approx 0.000105
\]
## Feature selection for Reuters classes coffee and sports

### Class: coffee

<table>
<thead>
<tr>
<th>term</th>
<th>MI</th>
</tr>
</thead>
<tbody>
<tr>
<td>COFFEE</td>
<td>0.0111</td>
</tr>
<tr>
<td>BAGS</td>
<td>0.0042</td>
</tr>
<tr>
<td>GROWERS</td>
<td>0.0025</td>
</tr>
<tr>
<td>KG</td>
<td>0.0019</td>
</tr>
<tr>
<td>COLOMBIA</td>
<td>0.0018</td>
</tr>
<tr>
<td>BRAZIL</td>
<td>0.0016</td>
</tr>
<tr>
<td>EXPORT</td>
<td>0.0014</td>
</tr>
<tr>
<td>EXPORTERS</td>
<td>0.0013</td>
</tr>
<tr>
<td>EXPORTS</td>
<td>0.0013</td>
</tr>
<tr>
<td>CROP</td>
<td>0.0012</td>
</tr>
</tbody>
</table>

### Class: sports

<table>
<thead>
<tr>
<th>term</th>
<th>MI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCCER</td>
<td>0.0681</td>
</tr>
<tr>
<td>CUP</td>
<td>0.0515</td>
</tr>
<tr>
<td>MATCH</td>
<td>0.0441</td>
</tr>
<tr>
<td>MATCHES</td>
<td>0.0408</td>
</tr>
<tr>
<td>PLAYED</td>
<td>0.0388</td>
</tr>
<tr>
<td>LEAGUE</td>
<td>0.0386</td>
</tr>
<tr>
<td>BEAT</td>
<td>0.0301</td>
</tr>
<tr>
<td>GAME</td>
<td>0.0299</td>
</tr>
<tr>
<td>GAMES</td>
<td>0.0284</td>
</tr>
<tr>
<td>TEAM</td>
<td>0.0264</td>
</tr>
</tbody>
</table>
Using language models (LMs) for IR

- LM = language model
- We view the document as a generative model that generates the query.
- What we need to do:
  - Define the precise generative model we want to use
  - Estimate parameters (different parameters for each document’s model)
  - Smooth to avoid zeros
  - Apply to query and find document most likely to have generated the query
  - Present most likely document(s) to user
Jelinek-Mercer smoothing

\[ P(t|d) = \lambda P(t|M_d) + (1 - \lambda) P(t|M_c) \]

- Mixes the probability from the document with the general collection frequency of the word.
- High value of \( \lambda \): “conjunctive-like” search – tends to retrieve documents containing all query words.
- Low value of \( \lambda \): more disjunctive, suitable for long queries
- Correctly setting \( \lambda \) is very important for good performance.
Vector space classification: Basic idea of doing text classification for documents that are represented as vectors

Rocchio classifier: Rocchio relevance feedback idea applied to text classification

$k$ nearest neighbor classification

Linear classifiers

More than two classes
Outline

1 Recap

2 Intro vector space classification

3 Rocchio

4 kNN

5 Linear classifiers

6 > two classes
Recall vector space representation

- Each document is a vector, one component for each term.
- Terms are axes.
- High dimensionality: 100,000s of dimensions
- Normalize vectors (documents) to unit length
- How can we do classification in this space?
Basic text classification setup

classes:

training set:

test set:

regions

industries

subject areas

UK

China

poultry

coffee

elections

sports

CONGESTION

Olympics

feed chicken

roasting beans

recount votes

diamond baseball

LONDON

Beijing

London

China

Beijing

China

China

First private Chinese airline

Olympics

congestion

Big Ben

Great Wall

Parliament

tourism

pate ducks

arabica robusta

seat run-off

forward soccer

Windsor

Mao

Bird flu turkey

Kenya harvest

TV ads campaign

Team captain

The Queen

Communist

Chicken feed

Beans

Elections

Run-off seat

Campaign

Team captain
As before, the training set is a set of documents, each labeled with its class.

In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space.

Premise 1: Documents in the same class form a contiguous region.

Premise 2: Documents from different classes don't overlap.

We define lines, surfaces, hypersurfaces to divide regions.
Should the document $\star$ be assigned to China, UK or Kenya? Find separators between the classes. Based on these separators: $\star$ should be assigned to China. How do we find separators that do a good job at classifying new documents like $\star$? – Main topic of today
Aside: 2D/3D graphs can be misleading

Left: A projection of the 2D semicircle to 1D. For the points $x_1, x_2, x_3, x_4, x_5$ at $x$ coordinates $-0.9, -0.2, 0, 0.2, 0.9$ the distance $|x_2x_3| \approx 0.201$ only differs by 0.5% from $|x'_2x'_3| = 0.2$; but $|x_1x_3|/|x'_1x'_3| = d_{true}/d_{projected} \approx 1.06/0.9 \approx 1.18$ is an example of a large distortion (18%) when projecting a large area. Right: The corresponding projection of the 3D hemisphere to 2D.
Outline

1. Recap

2. Intro vector space classification

3. Rocchio

4. kNN

5. Linear classifiers

6. > two classes
Relevance feedback

- In relevance feedback, the user marks documents as relevant/nonrelevant.
- Relevant/nonrelevant can be viewed as classes or categories.
- For each document, the user decides which of these two classes is correct.
- The IR system then uses these class assignments to build a better query (“model”) of the information need . . .
- . . . and returns better documents.
- Relevance feedback is a form of text classification.
Using Rocchio for vector space classification

- The principal difference between relevance feedback and text classification:
  - The training set is given as part of the input in text classification.
  - It is interactively created in relevance feedback.
Rocchio classification: Basic idea

- Compute a centroid for each class
  - The centroid is the average of all documents in the class.
- Assign each test document to the class of its closest centroid.
Recall definition of centroid

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

where \( D_c \) is the set of all documents that belong to class \( c \) and \( \vec{v}(d) \) is the vector space representation of \( d \).
Rocchio illustrated: $a_1 = a_2, b_1 = b_2, c_1 = c_2$
Rocchio algorithm

\textsc{TrainRocchio}(\mathbb{C}, \mathbb{D})

1. \textbf{for each} \( c_j \in \mathbb{C} \)
2. \textbf{do} \( D_j \leftarrow \{ d : \langle d, c_j \rangle \in \mathbb{D} \} \)
3. \( \mu_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \tilde{v}(d) \)
4. \textbf{return} \( \{ \mu_1, \ldots, \mu_J \} \)

\textsc{ApplyRocchio}(\{\mu_1, \ldots, \mu_J\}, d)

1. \textbf{return} \( \arg \min_j |\mu_j - \tilde{v}(d)| \)
Rocchio properties

- Rocchio forms a simple representation for each class: the **centroid**
  - We can interpret the centroid as the **prototype** of the class.
- Classification is based on similarity to / distance from centroid/prototype.
- Does not guarantee that classifications are consistent with the training data!
# Time complexity of Rocchio

<table>
<thead>
<tr>
<th>mode</th>
<th>time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>training</td>
<td>$\Theta(</td>
</tr>
<tr>
<td>testing</td>
<td>$\Theta( L_a +</td>
</tr>
</tbody>
</table>
Rocchio vs. Naive Bayes

- In many cases, Rocchio performs worse than Naive Bayes.
- One reason: Rocchio does not handle nonconvex, multimodal classes correctly.
Rocchio cannot handle nonconvex, multimodal classes

Exercise: Why is Rocchio not expected to do well for the classification task a vs. b here?

- A is centroid of the a’s, B is centroid of the b’s.
- The point o is closer to A than to B.
- But o is a better fit for the b class.
- A is a multimodal class with two prototypes.
- But in Rocchio we only have one prototype.
Outline

1 Recap
2 Intro vector space classification
3 Rocchio
4 kNN
5 Linear classifiers
6 > two classes
kNN classification

- kNN classification is another vector space classification method.
- It also is very simple and easy to implement.
- kNN is more accurate (in most cases) than Naive Bayes and Rocchio.
- If you need to get a pretty accurate classifier up and running in a short time . . .
- . . . and you don’t care about efficiency that much . . .
- . . . use kNN.
kNN classification

- kNN = \( k \) nearest neighbors
- **kNN classification rule for** \( k = 1 \) (1NN): Assign each test document to the class of its nearest neighbor in the training set.
- 1NN is not very robust – one document can be mislabeled or atypical.
- **kNN classification rule for** \( k > 1 \) (kNN): Assign each test document to the majority class of its \( k \) nearest neighbors in the training set.
- **Rationale of kNN:** contiguity hypothesis
  - We expect a test document \( d \) to have the same label as the training documents located in the local region surrounding \( d \).
Probabilistic kNN

- Probabilistic version of kNN: $P(c|d) = \text{fraction of } k \text{ neighbors of } d \text{ that are in } c$

- kNN classification rule for probabilistic kNN: Assign $d$ to class $c$ with highest $P(c|d)$
kNN is based on Voronoi tessellation
kNN algorithm

**Train-kNN(\(C, D\))**
1. \(D' \leftarrow \text{Preprocess}(D)\)
2. \(k \leftarrow \text{Select-k}(C, D')\)
3. return \(D', k\)

**Apply-kNN(\(D', k, d\))**
1. \(S_k \leftarrow \text{ComputeNearestNeighbors}(D', k, d)\)
2. for each \(c_j \in C(D')\)
3. do \(p_j \leftarrow |S_k \cap c_j|/k\)
4. return arg max\(_j\) \(p_j\)
Exercise

How is star classified by:

(i) 1-NN (ii) 3-NN (iii) 9-NN (iv) 15-NN (v) Rocchio?
Time complexity of kNN

**kNN with preprocessing of training set**

training $\Theta(|D| L_{ave})$

testing $\Theta(L_a + |D| M_{ave} M_a) = \Theta(|D| M_{ave} M_a)$

- kNN test time proportional to the size of the training set!
- The larger the training set, the longer it takes to classify a test document.
- kNN is inefficient for very large training sets.
- Question: Can we divide up the training set into regions, so that we only have to search in one region to do kNN classification for a given test document? (which perhaps would give us better than linear time complexity)
Curse of dimensionality

- Our intuitions about space are based on the 3D world we live in.
- Intuition 1: some things are close by, some things are distant.
- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- These two intuitions don’t necessarily hold for high dimensions.
- In particular: for a set of $k$ uniformly distributed points, let $d_{\text{min}}$ be the smallest distance between any two points and $d_{\text{max}}$ be the largest distance between any two points.
- Then
  \[
  \lim_{d \to \infty} \frac{d_{\text{max}} - d_{\text{min}}}{d_{\text{min}}} = 0
  \]
Curse of dimensionality: Simulation

- Simulate
  \[
  \lim_{d \to \infty} \frac{d_{\text{max}} - d_{\text{min}}}{d_{\text{min}}} = 0
  \]

- Pick a dimensionality \( d \)
- Generate 10 random points in the \( d \)-dimensional hypercube (uniform distribution)
- Compute all 45 distances
- Compute \( \frac{d_{\text{max}} - d_{\text{min}}}{d_{\text{min}}} \)
- We see that intuition 1 (some things are close, others are distant) is not true for high dimensions.
Intuition 2: Space can be carved up

- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- If this is true, then we have a simple and efficient algorithm for kNN.
- To find the $k$ closest neighbors of data point $<x_1, x_2, \ldots, x_d>$ do the following.
- Using binary search find all data points whose first dimension is in $[x_1 - \epsilon, x_1 + \epsilon]$. This is $O(\log n)$ where $n$ is the number of data points.
- Do this for each dimension, then intersect the $d$ subsets.
Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
  - for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
  - for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$
- In $d = 3$ dimensions?
  - for $d = 3$: $1 - (1 - 0.1^3)^{100} \approx 0.095$
- In $d = 4$ dimensions?
  - for $d = 4$: $1 - (1 - 0.1^4)^{100} \approx 0.0095$
- In $d = 5$ dimensions?
  - for $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$
Intuition 2: Space can be carved up

- In $d = 5$ dimensions?
- For $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$
- In other words: with enough dimensions, there is only one “local” region that will contain the nearest neighbor with high certainty: the entire search space.
- We cannot carve up high-dimensional space into neat neighborhoods . . .
- . . . unless the “true” dimensionality is much lower than $d$. 

kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
  - We always preprocess the training set, so in reality training time of kNN is linear.

- kNN is very accurate if training set is large.

- Optimality result: asymptotically zero error if Bayes rate is zero.

- But kNN can be very inaccurate if training set is small.
Outline

1. Recap
2. Intro vector space classification
3. Rocchio
4. kNN
5. Linear classifiers
6. > two classes
Linear classifiers

- **Definition:**
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - ...where $\theta$ (the threshold) is a parameter.

(First, we only consider binary classifiers.)

- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.

- We find this separator based on training set.

- Methods for finding separator: Perceptron, Rocchio, Naive Bayes – as we will explain on the next slides.

- Assumption: The classes are linearly separable.
A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$. The point at $\theta / w_1$.

Points $(d_1)$ with $w_1 d_1 \geq \theta$ are in the class $c$.

Points $(d_1)$ with $w_1 d_1 < \theta$ are in the complement class $\overline{c}$. 
A linear classifier in 2D

A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$

Example for a 2D linear classifier

Points $(d_1, d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class $c$.

Points $(d_1, d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$. 
A linear classifier in 3D

- A linear classifier in 3D is a plane described by the equation
  \[ w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta \]
- Example for a 3D linear classifier
- Points \((d_1 \ d_2 \ d_3)\) with \(w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta\) are in the class \(c\).
- Points \((d_1 \ d_2 \ d_3)\) with \(w_1 d_1 + w_2 d_2 + w_3 d_3 < \theta\) are in the complement class \(\overline{c}\).
Rocchio as a linear classifier

Rocchio is a linear classifier defined by:

$$\sum_{i=1}^{M} w_i d_i = \vec{w} \vec{d} = \theta$$

where $\vec{w}$ is the normal vector $\mu(c_1) - \mu(c_2)$ and $\theta = 0.5 \times (|\mu(c_1)|^2 - |\mu(c_2)|^2)$. 
Naive Bayes as a linear classifier

Multinomial Naive Bayes is a linear classifier (in log space) defined by:

$$\sum_{i=1}^{M} w_i d_i = \theta$$

where $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, $d_i$ = number of occurrences of $t_i$ in $d$, and $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$. Here, the index $i$, $1 \leq i \leq M$, refers to terms of the vocabulary (not to positions in $d$ as $k$ did in our original definition of Naive Bayes).
Classification decision based on majority of $k$ nearest neighbors.

The decision boundaries between classes are piecewise linear . . .

. . . but they are in general not linear classifiers that can be described as

$$\sum_{i=1}^{M} w_id_i = \theta.$$
Example of a linear two-class classifier

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$w_i$</th>
<th>$d_{1i}$</th>
<th>$d_{2i}$</th>
<th>$t_i$</th>
<th>$w_i$</th>
<th>$d_{1i}$</th>
<th>$d_{2i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>prime</td>
<td>0.70</td>
<td>0</td>
<td>1</td>
<td>dlr</td>
<td>-0.71</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>rate</td>
<td>0.67</td>
<td>1</td>
<td>0</td>
<td>world</td>
<td>-0.35</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>interest</td>
<td>0.63</td>
<td>0</td>
<td>0</td>
<td>sees</td>
<td>-0.33</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rates</td>
<td>0.60</td>
<td>0</td>
<td>0</td>
<td>year</td>
<td>-0.25</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>discount</td>
<td>0.46</td>
<td>1</td>
<td>0</td>
<td>group</td>
<td>-0.24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bundesbank</td>
<td>0.43</td>
<td>0</td>
<td>0</td>
<td>dlr</td>
<td>-0.24</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- This is for the class *interest* in Reuters-21578.
- For simplicity: assume a simple 0/1 vector representation
- $d_1$: “rate discount dlr world”
- $d_2$: “prime dlr”
- $\theta = 0$
- Exercise: Which class is $d_1$ assigned to? Which class is $d_2$ assigned to?
- We assign document $\vec{d}_1$ “rate discount dlr world” to *interest* since
  \[ \vec{w}^T \vec{d}_1 = 0.67 \cdot 1 + 0.46 \cdot 1 + (-0.71) \cdot 1 + (-0.35) \cdot 1 = 0.07 > 0 = \theta. \]
- We assign $\vec{d}_2$ “prime dlr” to the complement class (not in *interest*) since
  \[ \vec{w}^T \vec{d}_2 = -0.01 \leq \theta. \]
Which hyperplane?
In terms of actual computation, there are two types of learning algorithms.

(i) **Simple** learning algorithms that estimate the parameters of the classifier directly from the training data, often **in one linear pass**.
   - Naive Bayes, Rocchio, kNN are all examples of this.

(ii) **Iterative** algorithms
   - Support vector machines
   - Perceptron (example available as PDF on website: http://cislmu.org)

**The best performing learning algorithms usually require iterative learning.**
Perceptron update rule

- Randomly initialize linear separator \( \vec{w} \)
- Do until convergence:
  - Pick data point \( \vec{x} \)
  - If \( \text{sign}(\vec{w}^T \vec{x}) \) is correct class (1 or -1): do nothing
  - Otherwise: \( \vec{w} = \vec{w} - \text{sign}(\vec{w}^T \vec{x}) \vec{x} \)
Perceptron (class of $\vec{x}$ is YES)
Perceptron (class of $\vec{x}$ is YES)
Perceptron (class of $\vec{x}$ is YES)
Perceptron (class of $\vec{x}$ is YES)
Which hyperplane?
Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly . . .
- . . . but they behave differently on test data.
- Error rates on new data are low for some, high for others.
- How do we find a low-error separator?
- Perceptron: generally bad; Naive Bayes, Rocchio: ok; linear SVM: good
Linear classifiers: Discussion

- Many common text classifiers are linear classifiers: Naive Bayes, Rocchio, logistic regression, linear support vector machines etc.
- Each method has a different way of selecting the separating hyperplane
  - Huge differences in performance on test documents
- Can we get better performance with more powerful nonlinear classifiers?
- Not in general: A given amount of training data may suffice for estimating a linear boundary, but not for estimating a more complex nonlinear boundary.
A nonlinear problem

- Linear classifier like Rocchio does badly on this task.
- kNN will do well (assuming enough training data)
Which classifier do I use for a given TC problem?

- Is there a learning method that is optimal for all text classification problems?
- No, because there is a tradeoff between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy is the problem?
  - How stable is the problem over time?
    - For an unstable problem, it’s better to use a simple and robust classifier.
Outline

1 Recap

2 Intro vector space classification

3 Rocchio

4 kNN

5 Linear classifiers

6 > two classes
How to combine hyperplanes for $> 2$ classes?
One-of problems

- One-of or multiclass classification
  - Classes are mutually exclusive.
  - Each document belongs to exactly one class.
  - Example: language of a document (assumption: no document contains multiple languages)
One-of classification with linear classifiers

- Combine two-class linear classifiers as follows for one-of classification:
  - Run each classifier separately
  - Rank classifiers (e.g., according to score)
  - Pick the class with the highest score
Any-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves decisions open on all other classes.
  - A type of “independence” (but not statistical independence)
  - Example: topic classification
  - Usually: make decisions on the region, on the subject area, on the industry and so on “independently”
Any-of classification with linear classifiers

- Combine two-class linear classifiers as follows for any-of classification:
  - Simply run each two-class classifier separately on the test document and assign document accordingly
Take-away today

- **Vector space classification**: Basic idea of doing text classification for documents that are represented as vectors
- **Rocchio classifier**: Rocchio relevance feedback idea applied to text classification
- $k$ nearest neighbor classification
- Linear classifiers
- More than two classes
Resources

- Chapter 13 of IIR (feature selection)
- Chapter 14 of IIR
- Resources at http://cislmu.org
  - Perceptron example
  - Text classification chapter on decision tress and perceptrons: Manning & Schütze (1999)