Today

- A quick flashback from last week:
  - Linear classifiers
  - Logistic regression
- Multinomial logistic regression
  = Maximum entropy modeling
- Maximum entropy tagging
- Combining classifiers
- A glimpse on non-linear classifiers and SVM
- Comparing classifiers
Last week

- Some points from last week
- An example
- Linear classifiers
- Logistic regression
- Maximum entropy modeling
- Maximum entropy tagging
Linear classifiers — general case

- The classes can be separated by a hyperplane

\[ \sum_{i=1}^{M} w_i x_i = \theta \]

- (equivalently)

\[ \vec{w} \cdot \vec{x} = \sum_{i=0}^{M} w_i x_i = 0 \]

  - taking \( w_0 = -\theta \) and \( x_0 = 1 \)

- The object represented by \((x_1, x_2, \ldots, x_n)\)

  - is in \( C \) if and only if \( \sum_{i=1}^{M} w_i x_i > \theta \)
  - and in \(-C\) if \( \sum_{i=1}^{M} w_i x_i < \theta \)

- Or the other way around: Check \( \gg \) in each case!
Linear classifiers

- Rocchio
- Naive Bayes
- Logistic regression
- (SVM — with linear kernel)
- Perceptron

- Non-linear:
  - $k$NN
Naive Bayes is a linear classifier

\[ \hat{c} = \arg \max_{c \in \{c_1, c_2\}} P(c) \prod_{j=1}^{n} P(f_j \mid c) \]

\[ P(c_1) \prod_{j=1}^{n} P(f_j \mid c_1) > P(c_2) \prod_{j=1}^{n} P(f_j \mid c_2) \]

\[ \frac{P(c_1) \prod_{j=1}^{n} P(f_j \mid c_1)}{P(c_2) \prod_{j=1}^{n} P(f_j \mid c_2)} > 1 \]

\[ \sum_{i=1}^{M} w_i x_i = \theta \quad w_j = \log \left( \frac{P(f_j \mid c_1)}{P(f_j \mid c_2)} \right) \]

\[ \theta = -w_0 = -\log \left( \frac{P(c_1)}{P(c_2)} \right) \]
**NB and logistic regression**

- The NB uses a linear expression to decide
  \[
  \log \left( \frac{P(c_1 \mid f)}{P(c_2 \mid f)} \right) = \log \left( \frac{P(c_1 \mid f)}{1 - P(c_1 \mid f)} \right) = \mathbf{w} \cdot \mathbf{f} = \sum_{i=0}^{M} w_i x_i = w_0 x_0 + \sum_{i=1}^{M} w_i x_i > 0
  \]
- where
  \[
  w_j = \log \left( \frac{P(f_j \mid c_1)}{P(f_j \mid c_2)} \right)
  \]
- Are these the best choices for the \( w_j \)s?
  - **Logistic regression** instead faces the question directly:
  - Which \( w_j \)s make the best classifier of the form
  \[
  \text{logit} \left( P(c_1 \mid f) \right) = \ln \left( \frac{P(c_1 \mid f)}{1 - P(c_1 \mid f)} \right) = \mathbf{w} \cdot \mathbf{f} = \sum_{i=0}^{M} w_i x_i = w_0 x_0 + \sum_{i=1}^{M} w_i x_i > 0
  \]
Logistic regression – learning

- Conditional maximum likelihood estimation:
  Choose the model that fits the training data best!

\[
\hat{w} = \arg \max_w \prod_{i=1}^{m} P(c^i | \vec{f}^i)
\]

\[
P(c_1 | \vec{f}) = \frac{1}{1 + e^{-\vec{w} \cdot \vec{f}}}
\]

- where:
  - There are \( m \) many training data
  - \( c^i \) is the class of observation \( i \), i.e. \( c_1 \) or \( c_2 \).
  - The feature vector for observation \( i \) is: \( \vec{f}^i = (f_1^i, f_2^i, \ldots, f_n^i) \)
Learning: Gradient ascent
Today

- A quick flashback from last week:
  - Linear classifiers
  - Logistic regression
- Multinomial logistic regression
  = Maximum entropy modeling
- Maximum entropy tagging
- Combining classifiers
- A glimpse on non-linear classifiers and SVM
- Comparing classifiers
Remember NB

- We saw that for NB
  \[ P(c_1 \mid \vec{f}) > P(c_2 \mid \vec{f}) \]

- iff
  \[ \log \left( \frac{P(c_1)}{P(c_2)} \right) + \sum_{j=1}^{n} \log \left( \frac{P(f_j \mid c_1)}{P(f_j \mid c_2)} \right) > 0 \]

- This could also be written
  \[ (\log P(c_1) - \log P(c_2)) + \sum_{j=1}^{n} (\log P(f_j \mid c_1) - \log P(f_j \mid c_2)) > 0 \]

2. \[ \log P(c_1) + \sum_{j=1}^{n} \log P(f_j \mid c_1) > \log P(c_2) + \sum_{j=1}^{n} \log P(f_j \mid c_2) \]
Multinomial logistic regression

Multinomial regression \( \approx \) Naive Bayes (Bernoulli)
Logistic regression \( \approx \) Binary NB as linear classifier

- (J&M:) A linear expression for
  - each class \( c_i \) for \( i = 1,..,k \)
  - Choose the largest, cf.

\[
P(c^i \mid \vec{f}) = \frac{1}{Z} \exp (\vec{w}^i \bullet \vec{f}) = \frac{1}{Z} e^{\sum_j w^i_j f_j} = \frac{1}{Z} \prod_j (e^{w^i_j})^{f_j} = \frac{1}{Z} \prod_j a_j^{f_j}
\]

and

\[
Z = \sum_{i=1}^{k} \exp (\vec{w}^i \bullet \vec{f})
\]

where \( a_j = e^{w^j} \)
(In case you read other presentations, like Mitchell or Hastie et. al.):

They use a slightly different formulation, corresponding to where for \( i = 1, 2, \ldots, k-1 \):

\[
P(c^i | \vec{f}) = \frac{1}{Z} \exp(\vec{w}^i \cdot \vec{f}) = \frac{1}{Z} e^{\sum_j w_j f_j} = \frac{1}{Z} \prod_j (e^{w_j})^{f_j} = \frac{1}{Z} \prod_j a_j^{f_j}
\]

But \( Z = 1 + \sum_{i=1}^{k-1} \exp(\vec{w}^i \cdot \vec{f}) \) and \( P(c^k | \vec{f}) = \frac{1}{1 + \sum_{i=1}^{k-1} \exp(\vec{w}^i \cdot \vec{f})} \)

The two formulations are equivalent though:

In the J&M formulation, divide the numerator and denominator in each \( P(c^i | f) \) with \( P(c^k | f) \) and you get this formulation (with adjustments to \( Z \) and \( \mathbf{w} \)).
Indicator variables

- As we saw for NBB: features are binary
- Earlier notation
  - There are $m$ many training data: $o^1, o^2, \ldots, o^i, \ldots o^m$.
  - The feature vector for observation $i$ is: $\tilde{f}^i = (f_1^i, f_2^i, \ldots f_n^i)$
- Alternative notation $f_s(o^i)$ instead of $f_s^i$ or simply $f_s(o)$ when $o$ is obvious from the context.
- In addition usual to include class in indicator variables!
  - $f_j(c, o)$ for the indicator variable.
We would like to know whether to assign the class *VB* to *race* (or instead assign some other class like *NN*). One useful feature, we’ll call it $f_1$, would be the fact that the current word is *race*. We can thus add a binary feature which is true if this is the case:

$$f_1(c,x) = \begin{cases} 1 & \text{if } \text{word}_i = \text{“race”} \text{ & } c = \text{NN} \\ 0 & \text{otherwise} \end{cases}$$

Another feature would be whether the previous word has the tag *TO*:

$$f_2(c,x) = \begin{cases} 1 & \text{if } t_{i-1} = \text{TO} \text{ & } c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

Two more part-of-speech tagging features might focus on aspects of a word’s spelling and case:

$$f_3(c,x) = \begin{cases} 1 & \text{if } \text{suffix(}\text{word}_i\text{)} = \text{“ing”} \text{ & } c = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$
Why called "maximum entropy"?

<table>
<thead>
<tr>
<th>NN</th>
<th>JJ</th>
<th>NNS</th>
<th>VB</th>
<th>NNP</th>
<th>IN</th>
<th>MD</th>
<th>UH</th>
<th>SYM</th>
<th>VBG</th>
<th>POS</th>
<th>PRP</th>
<th>CC</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
<td>$\frac{1}{45}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NN</th>
<th>JJ</th>
<th>NNS</th>
<th>VB</th>
<th>NNP</th>
<th>IN</th>
<th>MD</th>
<th>UH</th>
<th>SYM</th>
<th>VBG</th>
<th>POS</th>
<th>PRP</th>
<th>CC</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NN</th>
<th>JJ</th>
<th>NNS</th>
<th>VB</th>
<th>NNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{4}{10}$</td>
<td>$\frac{1}{10}$</td>
<td>$\frac{4}{10}$</td>
<td>$\frac{1}{10}$</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NN</th>
<th>JJ</th>
<th>NNS</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{4}{10}$</td>
<td>$\frac{3}{20}$</td>
<td>$\frac{4}{10}$</td>
<td>$\frac{1}{20}$</td>
</tr>
</tbody>
</table>
Why called ”maximum entropy”? 

- The multinomial logistic regression yields the probability distribution which 
  - Gives the maximum entropy 
  - Given our training data
Similarly to the binary logistic regression,

- Regularization

---

**NLTK:** Some iterative optimization techniques are much faster than others.

- When training Maximum Entropy models, avoid the use of
  - Generalized Iterative Scaling (GIS) or
  - Improved Iterative Scaling (IIS),
- which are both considerably slower than the
  - Conjugate Gradient (CG) and
  - the BFGS optimization methods.

Hence, we happily skip FSNLP, sec 16.2.1
Today

- A quick flashback from last week:
  - Linear classifiers
  - Logistic regression
- Multinomial logistic regression
  = Maximum entropy modeling
- **Maximum entropy tagging**
- Combining classifiers
- A glimpse on non-linear classifiers and SVM
- Comparing classifiers
Given a sequence of words $w_1^n = w_1 w_2 \cdots w_n$.

Find the corresponding tagsequence $t_1^n$ which satisfies

$$\arg \max_{t_1^n} P(t_1^n | w_1^n)$$
HMM tagging

\[
\arg \max_{t_1^n} P(t_1^n \mid w_1^n) = \arg \max_{t_1^n} \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)} = \arg \max_{t_1^n} P(w_1^n \mid t_1^n)P(t_1^n)
\]

- **HMM**: simplifying assumptions:
  - Markov assumption for tags
    - (Bigram tagger)
  - Local dependency between \(w\) and \(t\):
    \[
P(t_1^n) = \prod_{i=1}^{n} P(t_i \mid t_{i-1}) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-1})
\]
  - Resulting expression
    \[
    \arg \max_{t_1^n} P(t_1^n \mid w_1^n) = \arg \max_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i)P(t_i \mid t_{i-1})
    \]
Different strategies

Directly attacking: \[ \arg \max_{t_i^n} P(t_i^n | w_i^n) \]
MaxEnt tagging

\[ P(t_1^n | w_1^n) = \prod_{i=1}^{n} P(t_i | t_1^{i-1}, w_1^n) \]

- At stage \( i \)
  - the history is an observation \( h_i = t_1^{i-1}, w_1^n \)
  - the tag \( t_i \) is a class

- Example feature:
  - \( f_k(h_i, t_i) \) 1 iff suffix \( (w_i) = "ing" \) and \( t_i = \text{VBG} \), otherwise 0

- Ratnaparkhi restricts histories to
  \[ h_i = \{w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, t_{i-2}, t_{i-1}\} \]

- Consider features from p.135
Maxent tagging decoding

- Ratnaparkhi: Beam search:
  - Tag from left to right
  - At stage $j$ have a list of the $N$ best hypotheses so far
  - At stage $j+1$,
    - for each $(k = 1, \ldots, N)$ hypothesis $(t_k)_j$ consider all possible tags $t_{j+1}$ and calculate the probability of $(t_k)_j t_{j+1}$
    - keep the $N$ best

- J&M: Maximum Entropy Markov Models
  - If the tags included in the history are restricted
    - E.g. Ratnaparkhi’s histories yield trigram
  - Use Viterbi

$$h_i = \{w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, t_{i-2}, t_{i-1}\}$$
Tagging: Maxent vs HMM

- Maxent has the possibility for
  - Different types of features
  - Specialized features that apply to some cases only
Today

- A quick flashback from last week:
  - Linear classifiers
  - Logistic regression
- Multinomial logistic regression
  = Maximum entropy modeling
- Maximum entropy tagging
- Combining classifiers
- A glimpse on non-linear classifiers and SVM
- Comparing classifiers
More than two classes (in general)

- **Any of** or **multivalue** classification
  - An item may belong to 1, 0 or more than 1 classes
  - Classes are independent
  - Use $n$ binary classifiers
  - Example: Documents

- **One-of** or **multinomial** classification
  - Each item belongs to one class
  - Classes are mutually exclusive
  - Example: POS-tagging
Many classifiers are built for binary problems.

Simply combining several binary quantifiers do not result in a one-of-classifier.
Combining binary classifiers

- Build a classifier for each class compared to its complement
- For a test document, evaluate it for membership in each class
- Assign document to class with either:
  - maximum probability
  - maximum score
  - maximum confidence
- Multinomial logistic regression is a good example
- Sometimes one postpones the decision and proceed with the probabilities (soft classification),
  - E.g. Maxent tagging
Today

- A quick flashback from last week:
  - Linear classifiers
  - Logistic regression

- Multinomial logistic regression
  = Maximum entropy modeling

- Maximum entropy tagging

- Combining classifiers

- A glimpse on non-linear classifiers and SVM

- Comparing classifiers
A nonlinear problem

- A linear classifier like Naïve Bayes does badly on this task
- kNN will do very well (assuming enough training data)
Selecting hyperplanes

- If the training set is linearly separable, there are infinitely many separating hyperplanes.
- They all separate the training set
- But are not equally good on general test data
  - Perceptron – not so good
  - Naïve Bayes and Rocchio better
- Support Vector Machine (SVM) finds an optimal solution.
  - Maximizes the distance between the hyperplane and the “difficult points” close to decision boundary
Support Vector Machine (SVM)

- SVMs maximize the margin around the separating hyperplane.
- The points in the training set closest to the separating planes are called support vectors.
- The decision function is specified by the support vectors.
- Currently widely seen as the best text classification method.
Non-linear SVMs

- Datasets that are linearly separable (with some noise) work out great:

- What to do if the datasets are not linearly separable?

- Map data to a higher-dimensional space using some suitable mapping.
  - Suitable: the resulting data are linearly separable
Principle of Support Vector Machines (SVM)
SVMs – main ideas

- Maximize the distance between training data and a separating plane.
- Mapping a non-linear problem to a linear problem in higher dimensions using a kernel function.
Today

- A quick flashback from last week:
  - Linear classifiers
  - Logistic regression
- Multinomial logistic regression
  = Maximum entropy modeling
- Maximum entropy tagging
- Combining classifiers
- A glimpse on non-linear classifiers and SVM
- Comparing classifiers
Maxent vs Naive Bayes

- If the Naive Bayes assumption is warranted — i.e. the features are independent — the two yield the same result in the limit.

- Otherwise, Maxent cope better with dependencies between features:
  - What happens in the two strategies if a feature gets repeated twice?

- With Maxent you may throw in features and let the model decide whether they are useful
- Maxent training is slower
Repeating a feature

<table>
<thead>
<tr>
<th>Example</th>
<th>P(c1)=0.5</th>
<th>P(c2)=0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(a</td>
<td>c1)=0.6</td>
<td>P(a</td>
</tr>
<tr>
<td>P(b</td>
<td>c1)=0.2</td>
<td>P(b</td>
</tr>
</tbody>
</table>

**Naive Bayes:**
- consider an observation containing $a$ and $c$:
  - Which class is assigned if each feature is counted once?
  - Which class if $a$ is counted twice and $b$ once?
Generative vs discriminative model

**Generative (e.g. NB)**
- $P(o, c)$
- $P(c | o)$
- $\arg\max_c P(c | o)$
- $P(o)$
- $\arg\max_o P(o)$
- $P(o | c)$

**Discriminative (e.g. Maxent)**
- $\ldots$
- $P(c | o)$
- $\arg\max_c P(c | o)$

See NLTK book
Which classifier do I use for a given classification problem?

- There is no learning method that is optimal for all classification problems.
  - 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 data?
  - How stable is the problem over time?
    - For an unstable problem, it’s better to use a simple and robust classifier.
Learning algorithms

- In terms of actual computation, there are two types of learning algorithms.
  
  1. Simple learning algorithms that estimate the parameters of the classifier directly from the training data,
    - Examples: Naive Bayes, Rocchio, kNN
  2. Iterative algorithms
    - Maxent
    - Support vector machines
    - Perceptron

- The best performing learning algorithms usually require iterative learning.
Naive Bayes vs. other methods

<table>
<thead>
<tr>
<th>(a)</th>
<th>NB</th>
<th>Rocchio</th>
<th>kNN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>micro-avg-L (90 classes)</td>
<td>80</td>
<td>85</td>
<td>86</td>
<td>89</td>
</tr>
<tr>
<td>macro-avg (90 classes)</td>
<td>47</td>
<td>59</td>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(b)</th>
<th>NB</th>
<th>Rocchio</th>
<th>kNN</th>
<th>trees</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>earn</td>
<td>96</td>
<td>93</td>
<td>97</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>acq</td>
<td>88</td>
<td>65</td>
<td>92</td>
<td>90</td>
<td>94</td>
</tr>
<tr>
<td>money-fx</td>
<td>57</td>
<td>47</td>
<td>78</td>
<td>66</td>
<td>75</td>
</tr>
<tr>
<td>grain</td>
<td>79</td>
<td>68</td>
<td>82</td>
<td>85</td>
<td>95</td>
</tr>
<tr>
<td>crude</td>
<td>80</td>
<td>70</td>
<td>86</td>
<td>85</td>
<td>89</td>
</tr>
<tr>
<td>trade</td>
<td>64</td>
<td>65</td>
<td>77</td>
<td>73</td>
<td>76</td>
</tr>
<tr>
<td>interest</td>
<td>65</td>
<td>63</td>
<td>74</td>
<td>67</td>
<td>78</td>
</tr>
<tr>
<td>ship</td>
<td>85</td>
<td>49</td>
<td>79</td>
<td>74</td>
<td>86</td>
</tr>
<tr>
<td>wheat</td>
<td>70</td>
<td>69</td>
<td>77</td>
<td>93</td>
<td>92</td>
</tr>
<tr>
<td>corn</td>
<td>65</td>
<td>48</td>
<td>78</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>micro-avg (top 10)</td>
<td>82</td>
<td>65</td>
<td>82</td>
<td>88</td>
<td>92</td>
</tr>
<tr>
<td>micro-avg-D (118 classes)</td>
<td>75</td>
<td>62</td>
<td>n/a</td>
<td>n/a</td>
<td>87</td>
</tr>
</tbody>
</table>

Evaluation measure: $F_1$
Naive Bayes does pretty well, but some methods beat it consistently (e.g., SVM).