Text Classification

Instructor: Yoav Artzi
Overview

• Classification Problems
  – Spam vs. Non-spam, Text Genre, Word Sense, etc.

• Supervised Learning
  – Naïve Bayes
  – Log-linear models (Maximum Entropy Models)
  – Weighted linear models and the Perceptron
  – Neural networks
Supervised Learning: Data

• Learning from annotated data
• Often the biggest problem
• Why?
  – Annotation requires specific expertise
  – Annotation is expensive
  – Data is private and not accessible
  – Often difficult to define and be consistent
• Before fancy models – always think about the data
Held-out Data

• Important tool for estimating generalization:
  – Train on one set, and evaluate during development on another
  – Test data: only use once!
Classification

• Automatically make a decision about inputs
  – Example: document $\rightarrow$ category
  – Example: image of digit $\rightarrow$ digit
  – Example: image of object $\rightarrow$ object type
  – Example: query + webpage $\rightarrow$ best match
  – Example: symptoms $\rightarrow$ diagnosis
  – ...

• Three main ideas:
  – Representation as feature vectors
  – Scoring by linear functions
  – Learning by optimizations
Probabilistic Classifiers

• Two broad approaches to predicting classes $y^*$

• Joint / Generative (e.g., Naïve Bayes)
  – Work with a *joint* probabilistic model of the data
  – Assume functional form for $P(X|Y), P(Y)$
  – Estimate probabilities from data (don’t forget to smooth)
  – Use Bayes rules to calculate $P(Y|X)$
    • E.g., represent $p(y,x)$ as Naïve Bayes model, compute $y^* = \arg\max_y p(y,x) = \arg\max_y p(y)p(x|y)$
  – Advantages: learning weights is easy and well understood

• Conditional / Discriminative (e.g., Logistic Regression)
  – Work with *conditional* probability $p(y|x)$
  – We can then direct compute $y^* = \arg\max_y p(y|x)$
  – Estimate parameters from data (don’t forget to regularize)
  – Advantages: Don’t have to model $p(x)! Can develop feature rich models for $p(y|x)$
Text Categorization

- Want to classify documents into broad semantic topics

Obama is hoping to rally support for his $825 billion stimulus package on the eve of a crucial House vote. Republicans have expressed reservations about the proposal, calling for more tax cuts and less spending. GOP representatives seemed doubtful that any deals would be made.

California will open the 2009 season at home against Maryland Sept. 5 and will play a total of six games in Memorial Stadium in the final football schedule announced by the Pac-10 Conference Friday. The original schedule called for 12 games over 12 weekends.

- Which one is the politics document? (And how much deep processing did that decision take?)
- First approach: bag-of-words and Naïve-Bayes models
- More approaches later…
- Usually begin with a labeled corpus containing examples of each class
Example: Spam Filter

• Input: email
• Output: spam/ham
• Setup:
  – Get a large collection of example emails, each labeled “spam” or “ham”
  – Note: someone has to hand label all this data!
  – Want to learn to predict labels of new, future emails
• Features: The attributes used to make the ham / spam decision
  – Words: FREE!
  – Text Patterns: $dd, CAPS
  – Non-text: SenderInContacts
  – …

Dear Sir.
First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.
General Text Classification

• Input:
  – Document $X$ of length $|X|$ is a sequence of tokens:

  \[ X = \langle x_1, \ldots, x_{|X|} \rangle \]

• Output:
  – One of $k$ labels $y$
Naïve-Bayes Models

• Generative model: pick a topic, then generate a document
• Naïve-Bayes assumption:
  – All words are independent given the topic.

\[ p(y, X) = q(y) \prod_{i=1}^{\mid X \mid} q(x_i \mid y) \]
Using NB for Classification

- We have a joint model of topics and documents

\[ p(y, X) = q(y) \prod_{i=1}^{\left| X \right|} q(x_i \mid y) \]

- To assign a label \( y^* \) to a new document \( \langle x_1, x_2, \ldots, x_n \rangle \):

\[ y^* = \arg \max_y p(y, X) = \arg \max_y q(y) \prod_{i=1}^{\left| X \right|} q(x_i \mid y) \]
Learning: Maximum Likelihood Estimate (MLE)

\[
p(y, X) = q(y) \prod_{i=1}^{|X|} q(x_i \mid y)
\]

- Parameters to estimate:
  - \(q(y) = \theta_y\) for each topic \(y\)
  - \(q(x \mid y) = \theta_{xy}\) for each topic \(y\) and word \(x\)

- Data:
  \[
  \{(X^{(j)}, y^{(j)})\}_{j=1}^N
  \]

- Objective:
  \[
  \arg \max_{\theta} \prod_{j=1}^N p(y^{(j)}, X^{(j)}) = \arg \max_{\theta} \prod_{j=1}^N q(y^{(j)}) \prod_{i=1}^{|X^{(j)}|} q(x_i \mid y^{(j)})
  \]
MLE

\[ p(y, X) = q(y) \prod_{i=1}^{\left| X \right|} q(x_i \mid y) \]

\[ \arg \max_{\theta} \prod_{j=1}^{N} p(y^{(j)}, X^{(j)}) = \arg \max_{\theta} \prod_{j=1}^{N} q(y^{(j)}) \prod_{i=1}^{\left| X^{(j)} \right|} q(x_i \mid y^{(j)}) \]

- How do we do learning? We count!

\[ q(y) = \theta_y = \frac{C(y)}{N} \quad q(x \mid y) = \theta_{xy} = \frac{C(x, y)}{C(y)} \]
Word Sparsity

\[
q(y) = \theta_y = \frac{C(y)}{N} \quad q(x \mid y) = \theta_{xy} = \frac{C(x, y)}{C(y)}
\]
Using NB for Classification

• We have a joint model of topics and documents

\[ p(y, X) = q(y) \prod_{i=1}^{\vert X \vert} q(x_i \mid y) \]

• To assign a label \( y^* \) to a new document \( \langle x_1, x_2, \ldots, x_n \rangle \):

\[ y^* = \arg \max_y p(y, X) = \arg \max_y q(y) \prod_{i=1}^{\vert X \vert} q(x_i \mid y) \]

• We get \( q(x_i \mid y) = 0 \) when \( C(x_i, y) = 0 \)
• Solution: smoothing + accounting for unknowns
  – More when we discuss language models
Using NB for Classification

• We have a joint model of topics and documents

\[ p(y, x_1, x_2 \ldots x_n) = q(y) \prod_i q(x_i \mid y) \]

• To assign a label \( y^* \) to a new document \(<x_1 x_2 \ldots x_n>\):

\[ y^* = \arg \max_y p(y, x_1, x_2 \ldots x_n) = \arg \max_y q(y) \prod_i q(x_i \mid y) \]

• How do we do learning?
  – We count!
• Smoothing? What about totally unknown words?
• Can work shockingly well for text categorization (especially in the wild)
• How can unigram models be so terrible for language modeling, but class-
  conditional unigram models work for text categorization?
• Numerical / speed issues?
Example: Word-sense Disambiguation

• Example:
  – living **plant** vs. manufacturing **plant**

• How do we tell these senses apart?
  – “context”
  
  The **plant** which had previously sustained the town’s economy shut down after an extended labor strike. The **plants** at the entrance, dry and wilted, the first victims of …

  – It’s just text categorization! (at the word level)
  – Each word sense represents a topic
Case Study: Word Senses

• Words have multiple distinct meanings, or senses:
  – Plant: living plant, manufacturing plant, …
  – Title: name of a work, ownership document, form of address, material at the start of a film, …

• Many levels of sense distinctions
  – Homonymy: totally unrelated meanings
    • river bank, money bank
  – Polysemy: related meanings
    • star in sky, star on TV
  – Systematic polysemy: productive meaning extensions or metaphor
    • metonymy such as organizations to their buildings
  – Sense distinctions can be extremely subtle (or not)

• Granularity of senses needed depends a lot on the task
• Why is it important to model word senses?
  – Translation, parsing, information retrieval?
Word Sense Disambiguation

- Example: living plant vs. manufacturing plant
- How do we tell these senses apart?
  - “context”
    
    The plant which had previously sustained the town’s economy shut down after an extended labor strike. The plants at the entrance, dry and wilted, the first victims of …

- Maybe it’s just text categorization
- Each word sense represents a topic
- Run a Naïve-Bayes classifier?

- Bag-of-words classification works OK for noun senses
  - 90% on classic, shockingly easy examples (line, interest, star)
  - 80% on senseval-1 nouns
  - 70% on senseval-1 verbs
Verb WSD

• Why are verbs harder?
  – Verbal senses less topical
  – More sensitive to structure, argument choice

• Verb Example: “Serve”
  – [function] The tree stump serves as a table
  – [enable] The scandal served to increase his popularity
  – [dish] We serve meals for the homeless
  – [enlist] She served her country
  – [jail] He served six years for embezzlement
  – [tennis] It was Agassi’s turn to serve
  – [legal] He was served by the sheriff
Better Features

• There are smarter features:
  – Argument selectional preference:
    • serve NP[meals] vs. serve NP[papers] vs. serve NP[country]
  – Sub-categorization:
    • [function] serve PP[as]
    • [enable] serve VP[to]
    • [tennis] serve <intransitive>
    • [food] serve NP {PP[to]}
  – Can be captured poorly (but robustly) with modified Naïve Bayes approach

• Other constraints (Yarowsky 95)
  – One-sense-per-discourse (only true for broad topical distinctions)
  – One-sense-per-collocation (pretty reliable when it kicks in: manufacturing plant, flowering plant)
Complex Features with NB

- Example:
  
  ```
  Washington County jail served 11,166 meals last month - a figure that translates to feeding some 120 people three times daily for 31 days.
  ```

- So we have a decision to make based on a set of cues:
  - context:jail, context:county, context:feeding, ...
  - local-context:jail, local-context:meals
  - subcat:NP, direct-object-head:meals

- Not clear how build a generative derivation for these:
  - Choose topic, then decide on having a transitive usage, then pick “meals” to be the object’s head, then generate other words?
  - How about the words that appear in multiple features?
  - Hard to make this work (though maybe possible)
  - No real reason to try
Where we are?

• So far: Naïve Bayes models for classification
  – Generative models, estimating $P(X \mid y)$ and $P(y)$
  – Assumption: features are independent given the label (often violated in practice)
  – Easy to estimate (just count!)

• Next: Discriminative models
  – Estimating $P(y \mid X)$ directly
  – Very flexible feature handling
  – Require numerical optimization methods
A Discriminative Approach

- View WSD as a discrimination task, directly estimate:
  \[ P(\text{sense} \mid \text{context:jail, context:county, context:feeding, …, local-context:jail, local-context:meals, subcat:NP, direct-object-head:meals, …}) \]

- Have to estimate multinomial (over senses) where there are a huge number of things to condition on
  - History is too complex to think about this as a smoothing / back-off problem

- Many feature-based classification techniques out there
  - Discriminative models extremely popular in the NLP community!
Feature Representations

Washington County jail served 11,166 meals last month - a figure that translates to feeding some 120 people three times daily for 31 days.

- Features are indicator functions which count the occurrences of certain patterns in the input
- Initially: we will have different feature values for every pair of input $X$ and class $y$

{ 
  context:jail = 1
  context:county = 1
  context:feeding = 1
  context:game = 0
  ...
  local-context:jail = 1
  local-context:meals = 1
  ...
  object-head:meals = 1
  object-head:ball = 0
}
Example: Text Classification

- Goal: classify document to categories
  
  ... win the election ...  POLITICS
  
  ... win the game ...  SPORTS
  
  ... see a movie ...  OTHER

- Classically: based on words in the document
- But other information sources are potentially relevant:
  - Document length
  - Average word length
  - Document’s source
  - Document layout
Some Notation

INPUT $X^{(j)}$ ... win the election ...

OUTPUT SPACE $\mathcal{Y}$ SPORTS, POLITICS, OTHER

OUTPUT $y$ SPORTS

TRUE OUTPUT $y^{(j)}$ POLITICS

FEATURE VECTOR $\phi(X^{(j)}, y) [1 0 1 0 0 0 0 0 0 0 0 0 0 0 0]$

SPORTS+”win” POLITICS+”win”
Block Feature Vectors

- Sometimes, we think of the input as having features, which are multiplied by outputs to form the candidates

\[ \ldots \text{win the election} \ldots \]

\[
\begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[ \phi(X,\text{SPORTS}) = [1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \]

\[ \phi(X,\text{POLITICS}) = [0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \]

\[ \phi(X,\text{OTHER}) = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0] \]
Non-block Feature Vectors

- Sometimes the features of candidates cannot be decomposed in this regular way.
- **Example:** A parse tree’s features may be the rules used for sentence $X$.

\[
\phi(X, \text{NP}_N \text{NP}_N \text{VP}_V) = [1 \ 0 \ 1 \ 0 \ 1]
\]

\[
\phi(X, \text{NP}_N \text{NP}_N \text{VP}_V) = [1 \ 1 \ 0 \ 1 \ 0]
\]

- Different candidates will often share features.
- We’ll return to the non-block case later.
Linear Models: Scoring

• In a linear model, each feature gets a weight in $w$

$$\phi(X, SPORTS) = [1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$$

$$\phi(X, POLITICS) = [0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$$

$$w = [1 \ 1 \ -1 \ -2 \ 1 \ -1 \ 1 \ -2 \ -2 \ -1 \ -1 \ 1]$$

• We compare $y$’s on the basis of their linear scores:

$$score(X, y; w) = w^\top \cdot \phi(X, y)$$

$$score(X, POLITICS; w) = 1 \times 1 + 1 \times 1 = 2$$
Linear Models: Prediction Rule

\[ w = [ \ 1 \ 1 \ -1 \ -2 \ 1 \ -1 \ 1 \ -2 \ -2 \ -1 \ -1 \ 1 ] \]

- The linear prediction rule:

\[
\text{prediction}(X, w) = \arg \max_{y \in \mathcal{Y}} w^\top \phi(X, y)
\]

\[
\begin{align*}
\phi(X, \text{SPORTS}) &= [1 \ 0 \ 1 \ 0 \ldots] \quad \text{score}(X, \text{SPORTS}, w) = 1 \times 1 + (-1) \times 1 = 0 \\
\phi(X, \text{POLITICS}) &= [\ldots 1 \ 0 \ 1 \ 0 \ldots] \quad \text{score}(X, \text{POLITICS}, w) = 1 \times 1 + 1 \times 1 = 2 \\
\phi(X, \text{OTHER}) &= [\ldots 1 \ 0 \ 1 \ 0] \quad \text{score}(X, \text{OTHER}, w) = (-2) \times 1 + (-1) \times 1 = -3
\end{align*}
\]

\[
\text{prediction}(X, w) = \text{POLITICS}
\]

- How do we get the weights?
How to Pick Weights?

• Goal: choose “best” vector $w$ given training data
  – For now, we mean “best for classification”

• The ideal: the weights which have greatest test set accuracy / F1 / whatever
  – But, don’t have the test set
  – Must compute weights from training set

• Maybe we want weights which give best training set accuracy?
  – Hard discontinuous optimization problem
  – May not (does not) generalize to test set
    • Easy to overfit
Naïve-Bayes as a Linear Model

• (Multinomial) Naïve-Bayes is a linear model:

\[ x^i = d_1, d_2, d_3, \ldots, d_n \]

\[
\begin{align*}
\phi(X, y) &= \begin{bmatrix} \ldots 0 \ldots, 1, \#v_1, \#v_2, \ldots, \#v_n, \ldots \end{bmatrix} \\
w &= \begin{bmatrix} \ldots, \log P(y), \log P(v_1|y), \log P(v_2|y), \ldots, \log P(v_n|y), \ldots \end{bmatrix}
\end{align*}
\]

\[
\text{score}(X, y, w) = w^\top \phi(X, y)
\]

\[
= \log P(y) + \sum_k \#v_k \log P(v_k|y)
\]

\[
= \log(P(y) \prod_k P(v_k|y)^{\#v_k})
\]

\[
= \log(P(y) \prod_{d \in x^i} P(d|y))
\]

\[
= \log P(X, y)
\]
Maximum Entropy Models (MaxEnt)

- Maximum entropy (logistic regression)
  - Model: use the scores as probabilities:
    \[
    p(y|X; w) = \frac{\exp \left( w \cdot \phi(X, y) \right)}{\sum_{y'} \exp \left( w \cdot \phi(X, y') \right)}
    \]
  - Learning: maximize the (log) conditional likelihood of training data \( \{(X^{(i)}, y^{(i)})\}_{i=1}^{N} \)
    \[
    L(w) = \log \prod_{i=1}^{N} p(y^{(i)}|X^{(i)}; w) = \sum_{i=1}^{N} \log p(y^{(i)}|X^{(i)}; w)
    \]
    \[
    w^* = \arg\max_w L(w)
    \]
  - Prediction:
    \[
    y^* = \arg\max_y p(y \mid X; w)
    \]
Unconstrained Optimization

\[
L(w) = \sum_{i=1}^{N} \log P(y^{(i)} \mid X^{(i)}; w) \quad w^* = \arg\max_w L(w)
\]

- Unfortunately, \(\arg\max_w L(w)\) doesn’t have a close formed solution
- The MaxEnt objective is an unconstrained optimization problem

\[L(w)\]

- Basic idea: move uphill from current guess
- Gradient ascent / descent follows the gradient incrementally
- At local optimum, derivative vector is zero
- Will converge if step sizes are small enough, but not efficient
- All we need is to be able to evaluate the function and its derivative
Unconstrained Optimization

• Once we have a function $f$, we can find a local optimum by iteratively following the gradient

• For convex functions:
  – A local optimum will be global
  – Does this mean that all is good?

• Basic gradient ascent isn’t very efficient, but there are simple enhancements which take into account previous gradients: conjugate gradient, L-BFGs

• There are special-purpose optimization techniques for MaxEnt, like iterative scaling, but they aren’t better
Derivative of the MaxEnt Objective

\[ L(w) = \sum_{i=1}^{N} \log p(y^{(i)} | X^{(i)}; w) \]

\[ p(y | X; w) = \frac{e^{w \cdot \phi(X,y)}}{\sum_{y'} e^{w \cdot \phi(X,y')}} \]

- Some necessities:

\[ w \cdot \phi(x, y) = w_1 \times \phi_1(x, y) + w_2 \times \phi_2(x, y) + \cdots + w_n \times \phi_n(x, y) \]

\[ \frac{\partial}{\partial x} \log_a u = \frac{1}{u \log_e a} \frac{\partial}{\partial x} u \]

\[ \frac{\partial}{\partial x} \log_e u = \frac{1}{u \log_e e} \frac{\partial}{\partial x} u = \frac{1}{u} \frac{\partial}{\partial x} u \]

\[ \frac{\partial}{\partial x} e^u = e^u \frac{\partial}{\partial x} u \]
Derivative of the MaxEnt Objective

\[ L(w) = \sum_{i=1}^{N} \log p(y^{(i)} | X^{(i)}; w) \]

\[ p(y | X ; w) = \frac{e^{w \cdot \phi(X,y)}}{\sum_{y'} e^{w \cdot \phi(X,y')}} \]
Derivative of the MaxEnt Objective

\[ L(w) = \sum_{i=1}^{N} \log p(y^{(i)} \mid X^{(i)}; w) \]

\[ p(y \mid X; w) = \frac{e^{w \cdot \phi(X, y)}}{\sum_{y'} e^{w \cdot \phi(X, y')}} \]

\[ \frac{\partial}{\partial w_j} L(w) = \frac{\partial}{\partial w_j} \sum_{i=1}^{N} \log P(y^{(i)} \mid X^{(i)}; w) \]

\[ = \frac{\partial}{\partial w_j} \sum_{i=1}^{N} \log \left( \frac{e^{w \cdot \phi(X^{(i)}, y^{(i)})}}{\sum_{y'} e^{w \cdot \phi(X^{(i)}, y')}} \right) \]

\[ = \frac{\partial}{\partial w_j} \sum_{i=1}^{N} \left( \log e^{w \cdot \phi(X^{(i)}, y^{(i)})} - \log \sum_{y'} e^{w \cdot \phi(X^{(i)}, y')} \right) \]

\[ = \frac{\partial}{\partial w_j} \sum_{i=1}^{N} \left( w \cdot \phi(X^{(i)}, y^{(i)}) - \log \sum_{y'} e^{w \cdot \phi(X^{(i)}, y')} \right) \]

\[ = \sum_{i=1}^{N} \left( \phi_j(X^{(i)}, y^{(i)}) - \frac{1}{\sum_{y'} e^{w \cdot \phi(X^{(i)}, y')}} \sum_{y'} e^{w \cdot \phi(X^{(i)}, y')} \phi_j(X^{(i)}, y') \right) \]

\[ = \sum_{i=1}^{N} \left( \phi_j(X^{(i)}, y^{(i)}) - \sum_{y'} \frac{e^{w \cdot \phi(X^{(i)}, y')}}{\sum_{y''} e^{w \cdot \phi(X^{(i)}, y'')}} \phi_j(X^{(i)}, y') \right) \]

\[ = \sum_{i=1}^{N} \left( \phi_j(X^{(i)}, y^{(i)}) - \sum_{y'} P(y' \mid X^{(i)}; w) \phi_j(X^{(i)}, y') \right) \]
Derivative of the MaxEnt Objective

\[ L(w) = \sum_{i=1}^{N} \log p(y^{(i)} | X^{(i)}; w) \]

\[ p(y | X; w) = \frac{e^{w \cdot \phi(X,y)}}{\sum_{y'} e^{w \cdot \phi(X,y')}} \]

\[ \frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{N} \left( \phi_j(X^{(i)}, y^{(i)}) - \sum_{y'} P(y' | X^{(i)}; w) \phi_j(X^{(i)}, y') \right) \]

Total count of feature \( j \) in correct candidates

Expected count of feature \( j \) in predicted candidates
Expected Counts

- The optimum parameters are the ones for which each feature’s predicted expectation equals its empirical expectation.

\[
\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{N} \left( \phi_j(X^{(i)}, y^{(i)}) - \sum_{y'} P(y'|X^{(i)}; w) \phi_j(X^{(i)}, y') \right)
\]
What About Overfitting?

• For Naïve Bayes, we were worried about zero counts in MLE estimates
  – Can that happen here?

• Regularization (smoothing) for Log-linear models
  – Instead, we worry about large feature weights
  – Add a regularization term to the likelihood to push weights towards zero

\[
L(w) = \sum_{i=1}^{N} \log p(y^{(i)}|X^{(i)}; w) - \frac{\lambda}{2} ||w||^2
\]
Derivative of the Regularized MaxEnt Objective

- Unfortunately, $\text{argmax}_w L(w)$ still doesn’t have a close formed solution
- We will have to differentiate and use gradient ascent

$$L(w) = \sum_{i=1}^{N} \left( w \cdot \phi(X^{(i)}, y^{(i)}) - \log \sum_y \exp(w \cdot \phi(X^{(i)}, y)) \right) - \frac{\lambda}{2} ||w||^2$$

$$\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{N} \left( \phi_j(X^{(i)}, y^{(i)}) - \sum_y p(y|X^{(i)}; w) \phi_j(X^{(i)}, y) \right) - \lambda w_j$$

- Total count of feature $j$ in correct candidates
- Expected count of feature $j$ in predicted candidates
- Big weights are bad
Example: NER Regularization

Because of regularization, the more common prefixes have larger weights even though entire-word features are more specific.

Local Context

<table>
<thead>
<tr>
<th>Word</th>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>at</td>
<td>IN</td>
<td>Grace</td>
<td>Road</td>
</tr>
<tr>
<td>x</td>
<td>Xx</td>
<td>Xx</td>
<td></td>
</tr>
</tbody>
</table>

**Feature Weights**

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>at</td>
<td>-0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>Current word</td>
<td>Grace</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>Beginning bigram</td>
<td>Gr</td>
<td>0.45</td>
<td>-0.04</td>
</tr>
<tr>
<td>Current POS tag</td>
<td>NNP</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>Prev and cur tags</td>
<td>IN NNP</td>
<td>-0.10</td>
<td>0.14</td>
</tr>
<tr>
<td>Current signature</td>
<td>Xx</td>
<td>0.80</td>
<td>0.46</td>
</tr>
<tr>
<td>Prev-cur-next sig</td>
<td>x-Xx-Xx</td>
<td>-0.69</td>
<td>0.37</td>
</tr>
<tr>
<td>P. state - p-cur sig</td>
<td>O-x-Xx</td>
<td>-0.20</td>
<td>0.82</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td>-0.58</td>
<td>2.68</td>
</tr>
</tbody>
</table>
A Very Nice Objective

- The MaxEnt objective behaves nicely:
  - Differentiable (so many ways to optimize)
  - Convex (so no local optima)

\[
f(\lambda a + (1 - \lambda)b) \geq \lambda f(a) + (1 - \lambda)f(b)
\]

Convexity guarantees a **single, global maximum value** because any higher points are greedily reachable.
Learning Classifiers

- Two probabilistic approaches to predicting classes $y^*$
  - **Joint:** work with a *joint* probabilistic model of the data, weights are (often) local conditional probabilities
    - E.g., represent $p(y,x)$ as Naïve Bayes model, compute $y^* = \arg\max_y p(y,X)$
  - **Conditional:** work with *conditional* probability $p(y \mid X)$
    - We can then direct compute $y^* = \arg\max_y p(y \mid X)$ Can develop feature rich models for $p(y \mid X)$.

- But, why estimate a distribution at all?
  - Linear predictor: $y^* = \arg\max_y w \cdot \phi(X,y)$
  - Perceptron algorithm
    - Online (or batch)
    - Error driven
    - Simple, additive updates
Perceptron Learning

• The perceptron algorithm
  – Iteratively processes the training set, reacting to training errors
  – Can be thought of as trying to drive down training error
• The online (binary $\rightarrow y = \pm 1$) perceptron algorithm:
  – Start with zero weights
  – Visit training instances $(X^{(i)}, y^{(i)})$ one by one, until all correct
    • Make a prediction
      \[
      y^* = \text{sign}(w \cdot \phi(X^{(i)}))
      \]
    • If correct ($y^* = y^{(i)}$): no change, goto next example!
    • If wrong: adjust weights
      \[
      w = w - y^* \phi(X^{(i)})
      \]
Two Simple Examples

Data set I:

\[ X^{(1)} = [1, 1], \quad y^{(1)} = 1 \]
\[ X^{(2)} = [1, -1], \quad y^{(2)} = 1 \]
\[ X^{(3)} = [-1, -1], \quad y^{(3)} = -1 \]

Data set II:

\[ X^{(1)} = [1, 1], \quad y^{(1)} = 1 \]
\[ X^{(2)} = [1, -1], \quad y^{(2)} = 1 \]
\[ X^{(3)} = [-1, -1], \quad y^{(3)} = -1 \]
\[ X^{(4)} = [0.25, 0.25], \quad y^{(4)} = -1 \]
Geometric Interpretation

• The perceptron finds a separating hyperplane

\[ X^{(1)} = [1, 1], \quad y^{(1)} = 1 \]
\[ X^{(2)} = [1, -1], \quad y^{(2)} = 1 \]
\[ X^{(3)} = [-1, -1], \quad y^{(3)} = -1 \]

\[ w = [1, 1] \]

Finding the hyperplane:

\[ w \cdot [x, y] = 1 \times x + 1 \times y = 0 \]
Geometric Interpretation II

- Start with zero weights
- Visit training instances \((x_i, y_i)\) one by one, until all correct
  - Make a prediction
    \[ y^* = \text{sign}(w \cdot \phi(X^{(i)})) \]
  - If correct \((y^* = y_i)\): no change, goto next example!
  - If wrong: adjust weights
    \[ w = w - y^* \phi(X^{(i)}) \]

\[ y^* = 1, y_i = -1 \]
Geometric Interpretation

• The perceptron finds a separating hyperplane

\[ X^{(1)} = [1, 1], \quad y^{(1)} = 1 \]
\[ X^{(2)} = [1, -1], \quad y^{(2)} = 1 \]
\[ X^{(3)} = [-1, -1], \quad y^{(3)} = -1 \]
\[ X^{(4)} = [0.25, 0.25], \quad y^{(4)} = -1 \]
\[ w = [0, 0] \]
\[ w = [1, 1] \]
\[ w = [0.75, 0.75] \]
\[ w = [0.5, 0.5] \]
\[ w = [0.25, 0.25] \]
Adding Bias

• Decision rule:

\[ y^* = \text{sign}(w \cdot \phi(X^{(i)}) + b) \]

• Algorithm stays the same!

• Only difference: dummy always-on feature

\[
X^{(1)} = [1, 1], \quad y^{(1)} = 1 \\
X^{(2)} = [1, -1], \quad y^{(2)} = 1 \\
X^{(3)} = [-1, -1], \quad y^{(3)} = -1 \\
w = [0, 0] \in \mathbb{R}^2
\]

\[
X^{(1)} = [1, 1, 1], \quad y^{(1)} = 1 \\
X^{(2)} = [1, 1, -1], \quad y^{(2)} = 1 \\
X^{(3)} = [1, -1, -1], \quad y^{(3)} = -1 \\
w = [0, 0, 0] \in \mathbb{R}^3
\]
Simple Example with Bias

Data set:

\[ X^{(1)} = [1, 1], \quad y^{(1)} = 1 \]
\[ X^{(2)} = [1, -1], \quad y^{(2)} = 1 \]
\[ X^{(3)} = [-1, -1], \quad y^{(3)} = -1 \]
\[ X^{(4)} = [0.25, 0.25], \quad y^{(4)} = -1 \]
Separable Case
Multiclass Perceptron

- If we have multiple classes:
  - A weight vector for each class: \( w_y \)
  - Score (activation) of a class \( y \):
    \[ w_y \cdot \phi(X) \]
  - Prediction highest score wins
    \[ y^* = \text{arg max}_{y} w_y \cdot \phi(X) \]
Multiclass Perceptron

- Start with zero weights
- Visit training instances \((X^{(i)}, y^{(i)})\) one by one
  - Make a prediction
    \[ y^* = \arg \max_y w_y \cdot \phi(X^{(i)}) \]
  - If correct \((y^* = y^{(i)})\): no change, continue
  - If wrong: adjust weights
    \[ w_{y_i} = w_{y^{(i)}} + \phi(X^{(i)}) \]
    \[ w_{y^*} = w_{y^*} - \phi(X^{(i)}) \]