Binary Classification

Wei Xu

(many slides from Greg Durrett and Vivek Srikumar)
Readings on course website

Homework 1 is due January 17.

Please look at the assignment, if you haven’t.

If this seems like it’ll be challenging for you, come and talk to me (this is smaller-scale than the later assignments, which are smaller-scale than the final project)
Alternatives

- LING 5801 or LING 5802 — covers similar topics as 5525 at a more moderate difficulty level.

- CSE 5522 or 5524 — CSE 5525 could be challenging for students who haven't taken 5522 or 5523. For undergraduates and non-CS major, it is recommended to take 5522 first (though not required).

- CSE 5539s (2-cr hrs) — taught by tenure-track faculty members.

- LING 7890.08 (1- or 2- cr hrs) — Clippers Seminar
This Lecture

- Linear classification fundamentals
- Naive Bayes, maximum likelihood in generative models
- Three discriminative models: logistic regression, perceptron, SVM
  - Different motivations but very similar update rules / inference!
Classification
Classification

- Datapoint $x$ with label $y \in \{0, 1\}$

- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$
  but in this lecture $f(x)$ and $x$ are interchangeable

- Linear decision rule: $w^\top f(x) + b > 0$
  
  $w^\top f(x) > 0$

- Can delete bias if we augment feature space:
  
  $f(x) = [0.5, 1.6, 0.3]$

  ↓

  $[0.5, 1.6, 0.3, 1]$
Linear functions are powerful!

\[ f(x) = [x_1, x_2] \]

\[ f(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2] \]

- “Kernel trick” does this for “free,” but is too expensive to use in NLP applications, training is \( O(n^2) \) instead of \( O(n \cdot (\text{num feats})) \)

Classification: Sentiment Analysis

- **this movie was great! would watch again**  Positive
- **that film was awful, I’ll never watch again**  Negative

- Surface cues can basically tell you what’s going on here: presence or absence of certain words (*great*, *awful*)
- Steps to classification:
  - Turn examples like this into feature vectors
  - Pick a model / learning algorithm
  - Train weights on data to get our classifier
Feature Representation

- Convert this example to a vector using *bag-of-words features*

  \[
  f(x) = [0 \quad 0 \quad 1 \quad 1 \quad 0 \quad \ldots]
  \]

- Very large vector space (size of vocabulary), sparse features
- Requires *indexing* the features (mapping them to axes)
- More sophisticated feature mappings possible (tf-idf), as well as lots of other features: character n-grams, parts of speech, lemmas, \ldots
Naive Bayes
Naive Bayes

- Data point \( x = (x_1, \ldots, x_n) \), label \( y \in \{0, 1\} \)
- Formulate a probabilistic model that places a distribution \( P(x, y) \)
- Compute \( P(y|x) \), predict \( \text{argmax}_y P(y|x) \) to classify

\[
P(y|x) = \frac{P(y)P(x|y)}{P(x)}
\]

Bayes’ Rule

constant: irrelevant

for finding the max

“Naive” assumption:

\[
P(y|x) = P(y) \prod_{i=1}^{n} P(x_i|y)
\]

\[
\text{argmax}_y P(y|x) = \text{argmax}_y \log P(y|x) = \text{argmax}_y \left[ \log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]
\]

linear model!
Naive Bayes Example

\[ P(y|x) \propto \prod_{i=1}^{n} P(x_i|y) \]

\[ \arg\max_y \log P(y|x) = \arg\max_y \left[ \log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right] \]
Maximum Likelihood Estimation

- Data points \((x_j, y_j)\) provided \((j\) indexes over examples\)

- Find values of \(P(y)\), \(P(x_i|y)\) that maximize data likelihood (generative):

\[
\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \prod_{i=1}^{n} P(x_{ji}|y_j)
\]

- Data points \((j)\)
- Features \((i)\)
- \(i\)th feature of \(j\)th example
Imagine a coin flip which is heads with probability $p$

Observe (H, H, H, T) and maximize likelihood:

$$\prod_{j=1}^{m} P(y_j) = p^3 (1 - p)$$

Easier: maximize log likelihood

$$\sum_{j=1}^{m} \log P(y_j) = 3 \log p + \log(1 - p)$$

Maximum likelihood parameters for binomial/multinomial = read counts off of the data + normalize

http://fooplot.com/
Data points \((x_j, y_j)\) provided \((j\) indexes over examples\)

Find values of \(P(y), P(x_i | y)\) that maximize data likelihood (generative):

\[
\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[ \prod_{i=1}^{n} P(x_{ji} | y_j) \right]
\]

- Data points \((j)\)
- Features \((i)\)
- \(i\)th feature of \(j\)th example

Equivalent to maximizing logarithm of data likelihood:

\[
\sum_{j=1}^{m} \log P(y_j, x_j) = \sum_{j=1}^{m} \left[ \log P(y_j) + \sum_{i=1}^{n} \log P(x_{ji} | y_j) \right]
\]
Maximum Likelihood for Naive Bayes

\[
P(+|y) \propto \left[ \begin{array}{c} P(+)P(\text{great}|+) \\ P(-)P(\text{great}|-) \end{array} \right] = \left[ \begin{array}{c} 1/4 \\ 1/8 \end{array} \right] = \left[ \begin{array}{c} 2/3 \\ 1/3 \end{array} \right]
\]

this movie was great! would watch again

\(P(+|y) = 1/2\)

I liked it well enough for an action flick

\(P(+) = 1/2\)

I expected a great film and left happy

\(P(\text{great}|+) = 1/2\)

brilliant directing and stunning visuals

that film was awful, I’ll never watch again

\(P(-) = 1/2\)

I didn’t really like that movie

\(P(\text{great}|-) = 1/4\)

dry and a bit distasteful, it misses the mark

\(P(+) = 1/2\)
great potential but ended up being a flop

\(P(-) = 1/2\)

it was great

\(P(+) = 1/2\)
Naive Bayes: Summary

- **Model**

\[ P(x, y) = P(y) \prod_{i=1}^{n} P(x_i | y) \]

- **Inference**

\[ \arg \max_y \log P(y \mid x) = \arg \max_y \left[ \log P(y) + \sum_{i=1}^{n} \log P(x_i \mid y) \right] \]

- **Alternatively:**

\[ \log P(y = + \mid x) - \log P(y = - \mid x) > 0 \]

\[ \iff \log \frac{P(y = +)}{P(y = -)} + \sum_{i=1}^{n} \log \frac{P(x_i \mid y = +)}{P(x_i \mid y = -)} > 0 \]

- **Learning:** maximize \( P(x, y) \) by reading counts off the data
Problems with Naive Bayes

- The film was beautiful, stunning cinematography and gorgeous sets, but boring.

\[
\begin{align*}
    P(x_{\text{beautiful}}|+) &= 0.1 & P(x_{\text{beautiful}}|-) &= 0.01 \\
    P(x_{\text{stunning}}|+) &= 0.1 & P(x_{\text{stunning}}|-) &= 0.01 \\
    P(x_{\text{gorgeous}}|+) &= 0.1 & P(x_{\text{gorgeous}}|-) &= 0.01 \\
    P(x_{\text{boring}}|+) &= 0.01 & P(x_{\text{boring}}|-) &= 0.1
\end{align*}
\]

- Correlated features compound: beautiful and gorgeous are not independent!
- Naive Bayes is naive, but another problem is that it’s generative: spends capacity modeling \(P(x,y)\), when what we care about is \(P(y|x)\).
- Discriminative models model \(P(y|x)\) directly (SVMs, most neural networks, ...).
Homework 1 Demo (Numpy)

- Multivariate Bernoulli or Multinominal Naive Bayes
- Use log probabilities
- Use Numpy vector/matrix operations. Avoid using for loops.
- Smoothing (ALPHA)
Logistic Regression
Logistic Regression

\[ P(y = + | x) = \text{logistic}(w^\top x) \]

\[ P(y = + | x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1 + \exp(\sum_{i=1}^{n} w_i x_i)} \]

- To learn weights: maximize discriminative log likelihood of data \( P(y | x) \)

\[ \mathcal{L}(x_j, y_j = +) = \log P(y_j = + | x_j) \]

\[ = \sum_{i=1}^{n} w_i x_{ji} - \log \left( 1 + \exp \left( \sum_{i=1}^{n} w_i x_{ji} \right) \right) \]

sum over features
Logistic Regression

\[ \mathcal{L}(x_j, y_j = +) = \log P(y_j = + | x_j) = \sum_{i=1}^{n} w_i x_{ji} - \log \left( 1 + \exp \left( \sum_{i=1}^{n} w_i x_{ji} \right) \right) \]

\[
\frac{\partial \mathcal{L}(x_j, y_j)}{\partial w_i} = x_{ji} - \frac{1}{1 + \exp \left( \sum_{i=1}^{n} w_i x_{ji} \right)} \frac{\partial}{\partial w_i} \left( 1 + \exp \left( \sum_{i=1}^{n} w_i x_{ji} \right) \right)
\]

\[
= x_{ji} - \frac{1}{1 + \exp \left( \sum_{i=1}^{n} w_i x_{ji} \right)} \left[ x_{ji} \exp \left( \sum_{i=1}^{n} w_i x_{ji} \right) \right]
\]

\[
= x_{ji} - \frac{\exp \left( \sum_{i=1}^{n} w_i x_{ji} \right)}{1 + \exp \left( \sum_{i=1}^{n} w_i x_{ji} \right)} x_{ji} = x_{ji} \left( 1 - P(y_j = + | x_j) \right)
\]
Logistic Regression

- Recall that $y_j = 1$ for positive instances, $y_j = 0$ for negative instances.

- Gradient of $w_i$ on positive example $= x_{ji}(y_j - P(y_j = + | x_j))$
  
  If $P(+) \text{ is close to } 1$, make very little update
  Otherwise make $w_i$ look more like $x_{ji}$, which will increase $P(+) $

- Gradient of $w_i$ on negative example $= x_{ji}(-P(y_j = + | x_j))$
  
  If $P(+) \text{ is close to } 0$, make very little update
  Otherwise make $w_i$ look less like $x_{ji}$, which will decrease $P(+) $

- Can combine these gradients as $x_j(y_j - P(y_j = 1 | x_j))$
Regularization

- Regularizing an objective can mean many things, including an L2-norm penalty to the weights:
  \[
  \sum_{j=1}^{m} \mathcal{L}(x_j, y_j) - \lambda \|w\|_2^2
  \]

- Keeping weights small can prevent overfitting

- For most of the NLP models we build, explicit regularization isn’t necessary
  - Early stopping
  - Large numbers of sparse features are hard to overfit in a really bad way
  - For neural networks: dropout and gradient clipping
Logistic Regression: Summary

- **Model**
  
  \[
P(y = +|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1 + \exp(\sum_{i=1}^{n} w_i x_i)}
  \]

- **Inference**
  
  \[
  \arg\max_y P(y|x) \quad \text{fundamentally same as Naive Bayes}
  \]
  
  \[
  P(y = 1|x) \geq 0.5 \iff w^\top x \geq 0
  \]

- **Learning:** gradient ascent on the (regularized) discriminative log-likelihood
Perceptron/SVM
Perceptron

- Simple error-driven learning approach similar to logistic regression

- Decision rule: \( w^\top x > 0 \)
  - If incorrect: if positive, \( w \leftarrow w + x \)
  - if negative, \( w \leftarrow w - x \)

- Guaranteed to eventually separate the data if the data are separable

Logistic Regression

\[
\begin{align*}
w & \leftarrow w + x(1 - P(y = 1|x)) \\
w & \leftarrow w - xP(y = 1|x)
\end{align*}
\]
Many separating hyperplanes — is there a best one?
Support Vector Machines

- Many separating hyperplanes — is there a best one?
Support Vector Machines

- Constraint formulation: find $w$ via following quadratic program:

$$\begin{align*}
\text{Minimize} & \quad \|w\|_2^2 \\
\text{s.t.} & \quad \forall j \quad w^\top x_j \geq 1 \text{ if } y_j = 1 \\
& \quad w^\top x_j \leq -1 \text{ if } y_j = 0
\end{align*}$$

minimizing norm with fixed margin $\iff$ maximizing margin

As a single constraint:

$$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1$$

- Generally no solution (data is generally non-separable) — need slack!
N-Slack SVMs

Minimize \( \lambda \|w\|^2_2 + \sum_{j=1}^{m} \xi_j \)

s.t. \( \forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j \quad \forall j \quad \xi_j \geq 0 \)

- The \( \xi_j \) are a “fudge factor” to make all constraints satisfied
- Take the gradient of the objective:
  \[
  \frac{\partial}{\partial w_i} \xi_j = 0 \text{ if } \xi_j = 0 \quad \frac{\partial}{\partial w_i} \xi_j = (2y_j - 1)x_{ji} \text{ if } \xi_j > 0 \\
  \quad \quad \quad \quad = x_{ji} \text{ if } y_j = 1, \quad -x_{ji} \text{ if } y_j = 0
  \]
- Looks like the perceptron! But updates more frequently
Gradients on Positive Examples

Logistic regression
\[ x(1 - \text{logistic}(w^\top x)) \]

Perceptron
\[ x \text{ if } w^\top x < 0, \text{ else } 0 \]

SVM (ignoring regularizer)
\[ x \text{ if } w^\top x < 1, \text{ else } 0 \]

*gradients are for maximizing things, which is why they are flipped*
## Comparing Gradient Updates (Reference)

<table>
<thead>
<tr>
<th>Method</th>
<th>Update Formula</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression (unregularized)</td>
<td>$x(y - P(y = 1</td>
<td>x)) = x(y - \text{logistic}(w^\top x))$</td>
</tr>
<tr>
<td>Perceptron</td>
<td>$(2y - 1)x$ if classified incorrectly</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0$ else</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>$(2y - 1)x$ if not classified correctly with margin of $1$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0$ else</td>
<td></td>
</tr>
</tbody>
</table>
Range of techniques from simple gradient descent (works pretty well) to more complex methods (can work better)

Most methods boil down to: take a gradient and a step size, apply the gradient update times step size, incorporate estimated curvature information to make the update more effective
Sentiment Analysis

- **this movie was** great! would **watch again** +
- **the movie was** gross and overwrought, but **I liked it** +
- **this movie was** not really very **enjoyable** —

- Bag-of-words doesn’t seem sufficient (discourse structure, negation)
- There are some ways around this: extract bigram feature for “not X” for all X following the *not*

Bo Pang, Lillian Lee, Shivakumar Vaithyanathan (2002)
### Sentiment Analysis

#### Simple feature sets can do pretty well!

<table>
<thead>
<tr>
<th>Features</th>
<th># of features</th>
<th>frequency or presence?</th>
<th>NB</th>
<th>ME</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) unigrams</td>
<td>16165</td>
<td>freq.</td>
<td>78.7</td>
<td>N/A</td>
<td>72.8</td>
</tr>
<tr>
<td>(2) unigrams</td>
<td></td>
<td>pres.</td>
<td>81.0</td>
<td>80.4</td>
<td>82.9</td>
</tr>
<tr>
<td>(3) unigrams+bigrams</td>
<td>32330</td>
<td>pres.</td>
<td>80.6</td>
<td>80.8</td>
<td>82.7</td>
</tr>
<tr>
<td>(4) bigrams</td>
<td>16165</td>
<td>pres.</td>
<td>77.3</td>
<td>77.4</td>
<td>77.1</td>
</tr>
<tr>
<td>(5) unigrams+POS</td>
<td>16695</td>
<td>pres.</td>
<td>81.5</td>
<td>80.4</td>
<td>81.9</td>
</tr>
<tr>
<td>(6) adjectives</td>
<td>2633</td>
<td>pres.</td>
<td>77.0</td>
<td>77.7</td>
<td>75.1</td>
</tr>
<tr>
<td>(7) top 2633 unigrams</td>
<td>2633</td>
<td>pres.</td>
<td>80.3</td>
<td>81.0</td>
<td>81.4</td>
</tr>
<tr>
<td>(8) unigrams+position</td>
<td>22430</td>
<td>pres.</td>
<td>81.0</td>
<td>80.1</td>
<td>81.6</td>
</tr>
</tbody>
</table>

Bo Pang, Lillian Lee, Shivakumar Vaithyanathan (2002)
Before neural nets had taken off — results weren’t that great

Ng and Jordan (2002) — NB can be better for small data

Naive Bayes is doing well!

<table>
<thead>
<tr>
<th>Method</th>
<th>RT-s</th>
<th>MPQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB.uni</td>
<td>77.9</td>
<td>85.3</td>
</tr>
<tr>
<td>MNB-bi</td>
<td>79.0</td>
<td>86.3</td>
</tr>
<tr>
<td>SVM-uni</td>
<td>76.2</td>
<td>86.1</td>
</tr>
<tr>
<td>SVM-bi</td>
<td>77.7</td>
<td>86.7</td>
</tr>
<tr>
<td>NBSVM-uni</td>
<td>78.1</td>
<td>85.3</td>
</tr>
<tr>
<td>NBSVM-bi</td>
<td>79.4</td>
<td>86.3</td>
</tr>
<tr>
<td>RAE</td>
<td>76.8</td>
<td>85.7</td>
</tr>
<tr>
<td>RAE-pretrain</td>
<td>77.7</td>
<td>86.4</td>
</tr>
<tr>
<td>Voting-w/Rev.</td>
<td>63.1</td>
<td>81.7</td>
</tr>
<tr>
<td>Rule</td>
<td>62.9</td>
<td>81.8</td>
</tr>
<tr>
<td>BoF-noDic.</td>
<td>75.7</td>
<td>81.8</td>
</tr>
<tr>
<td>BoF-w/Rev.</td>
<td>76.4</td>
<td>84.1</td>
</tr>
<tr>
<td>Tree-CRF</td>
<td>77.3</td>
<td>86.1</td>
</tr>
<tr>
<td>BoWSVM</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td><strong>Kim (2014) CNNs</strong></td>
<td><strong>81.5</strong></td>
<td><strong>89.5</strong></td>
</tr>
</tbody>
</table>
Recap

Logistic regression:  
\[ P(y = 1|x) = \frac{\exp\left(\sum_{i=1}^{n} w_i x_i\right)}{1 + \exp\left(\sum_{i=1}^{n} w_i x_i\right)} \]

Decision rule:  
\[ P(y = 1|x) \geq 0.5 \iff w^\top x \geq 0 \]

Gradient (unregularized):  
\[ x(y - P(y = 1|x)) \]

SVM:

Decision rule:  
\[ w^\top x \geq 0 \]

(Sub)gradient (unregularized): 0 if correct with margin of 1, else  
\[ x(2y - 1) \]
Recap

- Logistic regression, SVM, and perceptron are closely related.

- SVM and perceptron inference require taking maxes, logistic regression has a similar update but is “softer” due to its probabilistic nature.

- All gradient updates: “make it look more like the right thing and less like the wrong thing.”