Supervised Classification

CMSC 723 / LING 723 / INST 725

Marine Carpuat
marine@cs.umd.edu

Some slides by Graham Neubig, Jacob Eisenstein
Last time

• Text classification problems
  – and their evaluation
• Linear classifiers
  – Features & Weights
  – Bag of words
  – Naïve Bayes
Today

• 3 linear classifiers
  – Naïve Bayes
  – Perceptron
  – (Logistic Regression)

• Bag of words vs. rich feature sets
• Generative vs. discriminative models
• Bias-variance tradeoff
Naïve Bayes Recap

- Define $p(x, y)$ via a generative model
- Prediction: $\hat{y} = \arg \max_y p(x_i, y)$
- Learning:

$$
\theta = \arg \max_{\theta} p(x, y; \theta)
\quad p(x, y; \theta) = \prod_i p(x_i, y_i; \theta) = \prod_i p(x_i | y_i) p(y_i)
$$

$$
\phi_{y,j} = \frac{\sum_{i:Y_i=y} x_{ij}}{\sum_{i:Y_i=y} \sum_j x_{ij}}
\quad \mu_y = \frac{\text{count}(Y = y)}{N}
$$

This gives the maximum likelihood estimator (MLE; same as relative frequency estimator)
The Naivety of Naïve Bayes

\[
\log p(y_i, x_i) = \log p(x_i \mid y_i) + \log p(y_i) \\
= \sum_j \log p(x_{i,j} \mid y_i) + \log p(y_i)
\]
## Naïve Bayes: Example

<table>
<thead>
<tr>
<th>Cat</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>just plain boring</td>
</tr>
<tr>
<td></td>
<td>entirely predictable and lacks energy</td>
</tr>
<tr>
<td></td>
<td>no surprises and very few laughs</td>
</tr>
<tr>
<td></td>
<td>very powerful</td>
</tr>
<tr>
<td></td>
<td>the most fun film of the summer</td>
</tr>
<tr>
<td>Test</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>predictable with no originality</td>
</tr>
</tbody>
</table>
Smoothing

• Goal: assign some probability mass to events that were not seen during training

• One method: “add alpha” smoothing
  – Often, alpha = 1

\[
\phi_{y,j} = \frac{\alpha + \sum_{i: Y_i = y} x_{i,j}}{\sum_{j'=1}^{V} \left( \alpha + \sum_{i: Y_i = y} x_{i,j'} \right)} = \frac{\alpha + \text{count}(y, j)}{V \alpha + \sum_{j'=1}^{V} \text{count}(y, j')} 
\]
Multinomial Naïve Bayes: Learning in Practice

• From training corpus, extract *Vocabulary*

• Calculate $P(y_j)$ terms
  – For each $y_j$ in $Y$ do
    $$docs_j \leftarrow \text{all docs with class } = y_j$$
    $$P(y_j) \leftarrow \frac{|docs_j|}{|\text{total # documents}|}$$

• Calculate $P(w_k \mid y_j)$ terms
  • $Text_j \leftarrow \text{single doc containing all } docs_j$
  • For each word $w_k$ in *Vocabulary*
    $$n_k \leftarrow \text{# of occurrences of } w_k \text{ in } Text_j$$
    $$P(w_k \mid y_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$
Bias Variance trade-off

• **Variance** of a classifier
  – How much its decisions are affected by small changes in training sets
  – Lower variance = smaller changes

• **Bias** of a classifier
  – How accurate it is at modeling different training sets
  – Lower bias = more accurate

• High variance classifiers tend to **overfit**
• High bias classifiers tend to **underfit**
Bias Variance trade-off

• Impact of smoothing
  – Lowers variance
  – Increases bias (toward uniform probabilities)
Naïve Bayes

• A linear classifier whose weights can be interpreted as parameters of a probabilistic model

• Pros
  – parameters are easy to estimate from data: “count and normalize” (and smooth)

• Cons
  – requires making a conditional independence assumption
  – which does not hold in practice
Today

• 3 linear classifiers
  – Naïve Bayes
  – Perceptron
  – Logistic Regression

• Bag of words vs. rich feature sets
• Generative vs. discriminative models
• Bias-variance tradeoff
  – Smoothing, regularization
Beyond Bag of Words for classification tasks

Given an introductory sentence in Wikipedia
predict whether the article is about a person

Given
Gonso was a Sanron sect priest (754-827) in the late Nara and early Heian periods.  

Predict Yes!

Given
Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura, Maizuru City, Kyoto Prefecture.  

Predict No!
Designing features

Gonso was a Sanron sect priest (754 – 827) in the late Nara and early Heian periods.

Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura, Maizuru City, Kyoto Prefecture.
Predicting requires combining information

• Given features and weights

\[
\begin{align*}
 w_{\text{contains "priest"}} &= 2 \\
 w_{\text{contains "site"}} &= -3 \\
 w_{\text{contains "(<>-<>)"}} &= 1 \\
 w_{\text{contains "Kyoto Prefecture"}} &= -1
\end{align*}
\]

• Predicting for a **new example**:  
  - If (sum of weights > 0), “yes”; otherwise “no”

Kuya (903-972) was a priest born in Kyoto Prefecture.  

\[2 + (-1) + 1 = 2\]
Formalizing binary classification with linear models

\[ y = \text{sign}(w \cdot \varphi(x)) \]
\[ = \text{sign}(\sum_{i=1}^{I} w_i \cdot \varphi_i(x)) \]

- \( x \): the input
- \( \varphi(x) \): vector of feature functions \( \{\varphi_1(x), \varphi_2(x), \ldots, \varphi_I(x)\} \)
- \( w \): the weight vector \( \{w_1, w_2, \ldots, w_I\} \)
- \( y \): the prediction, +1 if “yes”, -1 if “no”
  - (\( \text{sign}(v) \) is +1 if \( v \geq 0 \), -1 otherwise)
Example feature functions: Unigram features

- Number of times a particular word appears
  - i.e. bag of words

\[ \mathbf{x} = \text{A site, located in Maizuru, Kyoto} \]

\[
\begin{align*}
\varphi_{\text{unigram "A"}}(\mathbf{x}) &= 1 \\
\varphi_{\text{unigram "site"}}(\mathbf{x}) &= 1 \\
\varphi_{\text{unigram ","}}(\mathbf{x}) &= 2 \\
\varphi_{\text{unigram "located"}}(\mathbf{x}) &= 1 \\
\varphi_{\text{unigram "in"}}(\mathbf{x}) &= 1 \\
\varphi_{\text{unigram "Maizuru"}}(\mathbf{x}) &= 1 \\
\varphi_{\text{unigram "Kyoto"}}(\mathbf{x}) &= 1 \\
\varphi_{\text{unigram "the"}}(\mathbf{x}) &= 0 \\
\varphi_{\text{unigram "temple"}}(\mathbf{x}) &= 0 \\
\varphi_{\text{unigram "..."}}(\mathbf{x}) &= 0
\end{align*}
\]

The rest are all 0
An **online** learning algorithm

```python
create map w
for / iterations
    for each labeled pair x, y in the data
        phi = CREATE_FEATURES(x)
        y' = PREDICT_ONE(w, phi)
        if y' != y
            UPDATE_WEIGHTS(w, phi, y)
```
Perceptron weight update

\[ w \leftarrow w + y \varphi(x) \]

- If \( y = 1 \), increase the weights for features in \( \varphi(x) \)
- If \( y = -1 \), decrease the weights for features in \( \varphi(x) \)
Example: initial update

- Initialize $\mathbf{w} = 0$

$x = \text{A site, located in Maizuru, Kyoto}$  \hspace{1cm} y = -1

\[ \mathbf{w} \cdot \varphi(x) = 0 \quad y' = \text{sign}(\mathbf{w} \cdot \varphi(x)) = 1 \]

\[ y' \neq y \]  

\[ \mathbf{w} \leftarrow \mathbf{w} + y \varphi(x) \]

\[ W_{\text{unigram "Maizuru"}} = -1 \]
\[ W_{\text{unigram ","}} = -2 \]
\[ W_{\text{unigram "in"}} = -1 \]
\[ W_{\text{unigram "Kyoto"}} = -1 \]
\[ W_{\text{unigram "A"}} = -1 \]
\[ W_{\text{unigram "site"}} = -1 \]
\[ W_{\text{unigram "located"}} = -1 \]
Example: second update

\[ \mathbf{x} = \text{Shoken, monk born in Kyoto} \quad y = 1 \]

\[ \mathbf{w} \cdot \varphi(x) = -4 \quad y' = \text{sign}(\mathbf{w} \cdot \varphi(x)) = -1 \]

\[ y' \neq y \]

\[ \mathbf{w} \leftarrow \mathbf{w} + y \varphi(x) \]

\[ \mathbf{w} \text{ unigram "Maizuru" } = -1 \]
\[ \mathbf{w} \text{ unigram "," } = -1 \]
\[ \mathbf{w} \text{ unigram "in" } = 0 \]
\[ \mathbf{w} \text{ unigram "Kyoto" } = 0 \]
\[ \mathbf{w} \text{ unigram "A" } = -1 \]
\[ \mathbf{w} \text{ unigram "site" } = -1 \]
\[ \mathbf{w} \text{ unigram "located" } = -1 \]
\[ \mathbf{w} \text{ unigram "Shoken" } = 1 \]
\[ \mathbf{w} \text{ unigram "monk" } = 1 \]
\[ \mathbf{w} \text{ unigram "born" } = 1 \]
Perceptron

- A linear model for classification
- An algorithm to learn feature weights given labeled data
  - online algorithm
  - error-driven
  - Does it converge?
    - See “A Course In Machine Learning” Ch.3
Multiclass perceptron

\[ \hat{y} = \arg \max_y \theta^T f(x, y) \]

**Algorithm 1** Perceptron learning algorithm

1: **procedure** PERCEPTRON \((x_{1:N}, y_{1:N})\)
2:    repeat
3:        Select an instance \(i\)
4:        \(\hat{y} \leftarrow \arg \max_y \theta_t^T f(x_i, y)\)
5:        if \(\hat{y} \neq y_i\) then
6:            \(\theta_{t+1} \leftarrow \theta_t + f(x_i, y_i) - f(x_i, \hat{y})\)
5:        else
7:            do nothing
8:    until tired
Bias Variance trade off

• How do we decide when to stop?
  – Accuracy on held out data
  – Early stopping

• Averaged perceptron
  – Improves generalization
Averaged perceptron

Algorithm 2 Averaged perceptron learning algorithm

1: procedure \textsc{Avg-Perceptron}(x_{1:N}, y_{1:N})
2: repeat
3: Select an instance $i$
4: $\hat{y} \leftarrow \arg \max_y \theta_t^\top f(x_i, y)$
5: if $\hat{y} \neq y_i$ then
6: $\theta_{t+1} \leftarrow \theta_t + f(x_i, y_i) - f(x_i, \hat{y})$
7: $m \leftarrow m + \theta_{t+1}$
8: else
9: do nothing
10: until tired
11: $\overline{\theta} \leftarrow \frac{1}{t} m$
Learning as optimization: Loss functions

- Naïve Bayes chooses weights to maximize the joint likelihood of the training data (or log likelihood)

\[
\log p(x, y; \theta) = \sum_{i=1}^{N} \log p(x_i, y_i; \theta)
\]

\[
\ell_{NB}(\theta; x_i, y_i) = -\log p(x_i, y_i; \theta)
\]

\[
\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} \ell_{NB}(\theta, x_i, y_i)
\]
Perceptron Loss function

\[ \ell_{\text{perceptron}}(\theta; x_i, y_i) = \begin{cases} 
0, & y_i = \arg \max_y \theta^\top f(x_i, y) \\
1, & \text{otherwise}
\end{cases} \]

- “0-1” loss
- Treats all errors equally
- Does not care about confidence of classification decision
Today

• 3 linear classifiers
  – Naïve Bayes
  – Perceptron
  – (Logistic Regression)

• Bag of words vs. rich feature sets
• Generative vs. discriminative models
• Bias-variance tradeoff
Perceptron & Probabilities

• What if we want a probability \( p(y|x) \)?

• The perceptron gives us a prediction \( y \)

In other words:

\[
\begin{align*}
P(y = 1|x) &= 1 \text{ if } w \cdot \varphi(x) \geq 0 \\
P(y = 1|x) &= 0 \text{ if } w \cdot \varphi(x) < 0
\end{align*}
\]
The logistic function

\[ P(y = 1 | x) = \frac{e^{w \cdot \varphi(x)}}{1 + e^{w \cdot \varphi(x)}} \]

- “Softer” function than in perceptron
- Can account for uncertainty
- Differentiable
Logistic regression: how to train?

- Train based on **conditional likelihood**
- Find parameters \( w \) that maximize conditional likelihood of all answers \( y_i \) given examples \( x_i \)

\[
\hat{w} = \arg \max_w \prod_i P(y_i | x_i ; w)
\]
Stochastic gradient ascent (or descent)

- Online training algorithm for logistic regression
  - and other probabilistic models

```
create map w
for / iterations
  for each labeled pair x, y in the data
    w += α * dP(y|x)/dw
```
Gradient of the logistic function

\[
\frac{d}{d w} P(y=1|x) = \frac{d}{d w} \frac{e^{w \cdot \varphi(x)}}{1+e^{w \cdot \varphi(x)}} = \varphi(x) \frac{e^{w \cdot \varphi(x)}}{(1+e^{w \cdot \varphi(x)})^2}
\]

\[
\frac{d}{d w} P(y=-1|x) = \frac{d}{d w} \left(1 - \frac{e^{w \cdot \varphi(x)}}{1+e^{w \cdot \varphi(x)}}\right) = -\varphi(x) \frac{e^{w \cdot \varphi(x)}}{(1+e^{w \cdot \varphi(x)})^2}
\]
Example: initial update

- Set $\alpha=1$, initialize $w=0$

$x = \text{A site, located in Maizuru, Kyoto}$ \hspace{1cm} y = -1$

\[ w \cdot \varphi(x) = 0 \]

\[
\frac{d}{dw} P(y = -1|x) = -\frac{e^0}{(1+e^0)^2} \varphi(x)
\]

\[
= -0.25 \varphi(x)
\]

\[ w \leftarrow w + -0.25 \varphi(x) \]

\[
\begin{align*}
W_{\text{unigram “Maizuru”}} &= -0.25 \\
W_{\text{unigram “,”}} &= -0.5 \\
W_{\text{unigram “in”}} &= -0.25 \\
W_{\text{unigram “Kyoto”}} &= -0.25 \\
W_{\text{unigram “A”}} &= -0.25 \\
W_{\text{unigram “site”}} &= -0.25 \\
W_{\text{unigram “located”}} &= -0.25
\end{align*}
\]
Example: second update

\[ \mathbf{x} = \text{Shoken, monk born in Kyoto} \quad y = 1 \]

\[ \mathbf{w} \cdot \varphi(x) = -1 \]

\[ \frac{d}{d\mathbf{w}} P(y = 1|x) = \frac{e^1}{(1+e^1)^2} \varphi(x) \]

\[ = 0.196 \varphi(x) \]

\[ \mathbf{w} \leftarrow \mathbf{w} + 0.196 \varphi(x) \]

- \( W_{\text{unigram "Maizuru"}} = -0.25 \)
- \( W_{\text{unigram ","}} = -0.304 \)
- \( W_{\text{unigram "in"}} = -0.054 \)
- \( W_{\text{unigram "Kyoto"}} = -0.054 \)
- \( W_{\text{unigram "A"}} = -0.25 \)
- \( W_{\text{unigram "site"}} = -0.25 \)
- \( W_{\text{unigram "located"}} = -0.25 \)
- \( W_{\text{unigram "Shoken"}} = 0.196 \)
- \( W_{\text{unigram "monk"}} = 0.196 \)
- \( W_{\text{unigram "born"}} = 0.196 \)
How to set the learning rate?

• Various strategies
  • decay over time

\[ \alpha = \frac{1}{C + t} \]

• Use held-out test set, increase learning rate when likelihood increases
Some models are better than others...

- Consider these 2 examples

  -1  he saw a bird in the park
  +1  he saw a robbery in the park

- Which of the 2 models below is better?

<table>
<thead>
<tr>
<th>Classifier 1</th>
<th>Classifier 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>he +3</td>
<td>bird -1</td>
</tr>
<tr>
<td>saw -5</td>
<td>robbery +1</td>
</tr>
<tr>
<td>a +0.5</td>
<td>in +5</td>
</tr>
<tr>
<td>bird -1</td>
<td>the -3</td>
</tr>
<tr>
<td>robbery +1</td>
<td>park -2</td>
</tr>
</tbody>
</table>

Classifier 2 will probably generalize better! It does not include irrelevant information => Smaller model is better
Regularization

• A penalty on adding extra weights

• L2 regularization: $\|w\|_2$
  – big penalty on large weights
  – small penalty on small weights

• L1 regularization: $\|w\|_1$
  – Uniform increase when large or small
  – Will cause many weights to become zero
L1 regularization in online learning

```
update_weights(w, phi, y, c)
    for name, value in w:
        if abs(value) < c:
            w[name] = 0
        else:
            w[name] -= sign(value) * c
    for name, value in phi:
        w[name] += value * y
```

- If abs. value < c, set weight to zero
- If value > 0, decrease by c
- If value < 0, increase by c
Today

• 3 linear classifiers
  – Naïve Bayes
  – Perceptron
  – (Logistic Regression)

• Bag of words vs. rich feature sets
• Generative vs. discriminative models
• Bias-variance tradeoff