Supervised Classifiers: Naïve Bayes, Perceptron, Logistic Regression

CMSC 723 / LING 723 / INST 725

MARINE CARPUAT
marine@cs.umd.edu

Some slides by Graham Neubig, Jacob Eisenstein
Today

- 3 linear classifiers
  - Naïve Bayes
  - Perceptron
  - Logistic Regression

- Bag of words vs. rich feature sets
- Generative vs. discriminative models
- Bias-variance tradeoff
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 \( \varphi(x) \)
The 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
When to stop?

• One technique
  – When the accuracy on held out data starts to decrease
  – Early stopping
**Multiclass perceptron**

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

---

**Algorithm 1** Perceptron learning algorithm

1: **procedure** \textsc{Perceptron}(\(x_{1:N}, y_{1:N}\))
2: \hspace{1em} \textbf{repeat}
3: \hspace{2em} Select an instance \(i\)
4: \hspace{2em} \(\hat{y} \leftarrow \arg \max_y \theta_t^T f(x_i, y)\)
5: \hspace{2em} \textbf{if} \(\hat{y} \neq y_i\) \textbf{then}
6: \hspace{3em} \(\theta_{t+1} \leftarrow \theta_t + f(x_i, y_i) - f(x_i, \hat{y})\)
7: \hspace{2em} \textbf{else}
8: \hspace{3em} do nothing
9: \hspace{2em} \textbf{until} tired
Algorithm 2 Averaged perceptron learning algorithm

1: procedure AVG-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: $\theta_{t+1} \leftarrow \theta_t + f(x_i, y_i) - f(x_i, \hat{y})$
7: $m \leftarrow m + \theta_{t+1}$
8: else
9: do nothing
10: until tired
11: $\overline{\theta} \leftarrow \frac{1}{t} m$
What objective/loss does the perceptron optimize?

- Zero-one loss function

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

- What are the pros and cons compared to Naïve Bayes loss?
Perceptron & Probabilities

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

• The perceptron gives us a prediction $y$

In other words:

$$P(y = 1|x) = 1 \text{ if } w \cdot \varphi(x) \geq 0$$

$$P(y = 1|x) = 0 \text{ if } w \cdot \varphi(x) < 0$$
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

```c
create map w
for / iterations
    for each labeled pair x, y in the data
        w += \alpha \cdot \frac{dP(y|x)}{dw}
```

- Update weights for every training example
- Move in direction given by gradient
- Size of update step scaled by learning rate
Gradient of the logistic function

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

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

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

$x = \text{A site, located in Maizuru, Kyoto}$

$y = -1$

\[
\begin{align*}
    w \cdot \varphi(x) &= 0 \\
    \frac{d}{dw} P(y = -1 | x) &= -\frac{e^0}{(1+e^0)^2} \varphi(x) \\
    &= -0.25 \varphi(x)
\end{align*}
\]

$w \leftarrow w + -0.25 \varphi(x)$

$W_{\text{unigram "Maizuru"}} = -0.25$  
$W_{\text{unigram "A"}} = -0.25$  
$W_{\text{unigram ","}} = -0.5$  
$W_{\text{unigram "in"}} = -0.25$  
$W_{\text{unigram "located"}} = -0.25$  
$W_{\text{unigram "Kyoto"}} = -0.25$
Example: second update

\[ x = \text{Shoken, monk born in Kyoto} \]

\[ y = 1 \]

\[ w \cdot \phi(x) = -1 \]

\[ \frac{d}{dw} P(y = 1|x) = \frac{e^1}{(1+e^1)^2} \phi(x) = 0.196 \phi(x) \]

\[ w \leftarrow w + 0.196 \phi(x) \]

\[ w_{\text{unigram "Maizuru"}} = -0.25 \]

\[ w_{\text{unigram ","}} = -0.304 \]

\[ w_{\text{unigram "in"}} = -0.054 \]

\[ w_{\text{unigram "Kyoto"}} = -0.054 \]

\[ w_{\text{unigram "A"}} = -0.25 \]

\[ w_{\text{unigram "site"}} = -0.25 \]

\[ w_{\text{unigram "located"}} = -0.25 \]

\[ w_{\text{unigram "Shoken"}} = 0.196 \]

\[ w_{\text{unigram "monk"}} = 0.196 \]

\[ w_{\text{unigram "born"}} = 0.196 \]
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
Multiclass version

\[ p(y \mid x) = \frac{\exp \left( \theta^\top f(x, y) \right)}{\sum_{y' \in \mathcal{Y}} \exp \left( \theta^\top f(x, y') \right)} . \]
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

```python
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
What you should know

• Standard supervised learning set-up for text classification
  – Difference between train vs. test data
  – How to evaluate

• 3 examples of supervised linear classifiers
  – Naïve Bayes, Perceptron, Logistic Regression
  – Learning as optimization: what is the objective function optimized?
  – Difference between generative vs. discriminative classifiers
  – Bias-Variance trade-off, smoothing, regularization