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

- 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: $w^T f(x) + b > 0$
  - $w^T f(x) > 0$
- 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!

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

"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

- Doing well at this is going to require structure, but let's start with simple approaches

    *This movie was* great! would watch again  Positive
    *This movie was* not really very enjoyable  Negative

Text Classification: Ham or Spam

- Machine learning is good at this! Lots of data, simple pattern recognition task, hard to write rules by hand

  Hi, I just wanted to send over the latest results from training the LSTM model. In the attachment. What do you think of the performance?

  hi i have very valuable business proposition for you. you make lots of $$$ i just need you to send a small amount of funds

  Surface cues can basically tell you what’s going on here

  Ham  Spam
Naive Bayes

- Data point $x = (x_1, ..., x_n)$, label $y \in \{0, 1\}$
- Formulate a probabilistic model that places a distribution $P(x, y)$
- Compute $P(y|x)$ and then label an example with $\text{argmax}_y P(y|x)$
- $P(y|x) = \frac{P(y)P(x|y)}{P(x)}$ Bayes’ Rule
- $\propto P(y)P(x|y)$ constant: irrelevant for finding the max
- $= P(y) \prod_{i=1}^{n} P(x_i|y)$ “Naive” assumption:
- $\text{argmax}_y P(y|x) = \text{argmax}_y \log P(y|x) = \text{argmax}_y \left[ \log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]$ linear model!

Note that this is not $P(y|x)$ — not the probability of ham given the word spam gets more points in the final posterior
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)
- Equivalent to maximizing logarithm of data likelihood:
  \[
  \sum_{j=1}^{m} \log P(y_j, x_j) = \sum_{j=1}^{m} \log P(y_j) + \sum_{i=1}^{n} \log P(x_{ji}|y_j)
  \]

Maximum Likelihood for Naive Bayes

- Imagine a coin flip which is heads with probability \(p\)
- Observe (H, H, H, T) and maximize log likelihood
  \[
  \sum_{j=1}^{m} \log P(y_j) = 3 \log p + \log (1 - p)
  \]
- Maximum likelihood parameters for multinomial = read counts off of the data

Naive Bayes: Summary

- Model
  \[
  P(x, y) = P(y) \prod_{i=1}^{n} P(x_i|y)
  \]
- Inference
  \[
  \argmax_y \log P(y|x) = \argmax_y \left[ \log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]
  \]
- Alternatively:
  \[
  \log P(y = \text{spam}|x) - \log P(y = \text{ham}|x) > 0
  \leftrightarrow \log \frac{P(y = \text{spam}|x)}{P(y = \text{ham}|x)} + \sum_{i=1}^{n} \log \frac{P(x_i|y = \text{spam})}{P(x_i|y = \text{ham})} > 0
  \]
- Learning: maximize \(P(x, y)\) by reading counts off the data

Hi, I just wanted to send over the latest results from training the LSTM model. In the attachment. What do you think of the performance?

\(P(y = \text{ham}) = 0.5\)
\(P(x_{\text{funds}} = 1|\text{spam}) = 1\)
\(P(x_{\text{funds}} = 0|\text{spam}) = 0\)

Smoothing: add very small counts for each entry to avoid zeroes (bias-variance tradeoff)
\(P(x_{\text{funds}} = 1|\text{spam}) = 0.99\)
\(P(x_{\text{funds}} = 0|\text{spam}) = 0.01\)
Problems with Naive Bayes

- Features are correlated
  \[ P(x_{\text{funds}} = 1 | \text{spam}) = 0.1 \]
  \[ P(x_{\text{funds}} = 1 | \text{ham}) = 0.01 \]
  \[ P(x_{\text{transfer}} = 1 | \text{spam}) = 0.1 \]
  \[ P(x_{\text{transfer}} = 1 | \text{ham}) = 0.01 \]
- This one sentence will make the probability of spam very high!
- Bad independence assumption in NB: these words are not independent!
- Solution: better model, algorithms that explicitly minimize loss rather than maximizing data likelihood

Generative vs. Discriminative Models

- Generative models: \( P(x, y) \)
  - Bayes nets / graphical models
  - Some of the model capacity goes to explaining the distribution of \( x \); prediction uses Bayes rule post-hoc
  - Can sample new instances \((x, y)\)
- Discriminative models: \( P(y|x) \)
  - SVMs, logistic regression, CRFs, most neural networks
  - Model is trained to be good at prediction, but doesn’t model \( x \)
- We’ll come back to this distinction throughout this class

Break!

Logistic Regression

- \[ P(y = \text{spam}|x) = \text{logistic}(w^T x) \]
- \[ P(y = \text{spam}|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1 + \exp(\sum_{i=1}^{n} w_i x_i)} \]
- How to set the weights \( w \)?
- (Stochastic) gradient ascent to maximize log likelihood

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

Logistic Regression

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

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

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

- Final gradient:
  \[ x_j(y_j - P(y_j = 1|x_j)) \]

**Regularization**

- Can end up making extreme updates to fit the training data.
  
  \[ w_{\text{funds}} = +1000, \quad w_{\text{transfer}} = -900, \quad w_{\text{send}} = +742, \quad w_{\text{the}} = +203 \]

- All examples have $P(\text{correct}) > 0.999$, but classifier does crazy things on new examples.

**Regularization**

- Can end up making extreme updates to fit the training data.

- Rather than optimizing likelihood alone, impose a penalty on the norm of the weight vector (can also view as a Gaussian prior).

- Maximize:
  \[
  \sum_{j=1}^{m} \mathcal{L}(x_j, y_j) - \lambda \|w\|^2
  \]

**Logistic Regression: Summary**

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

- Inference:
  \[
  \arg\max_y P(y|x) \quad \text{similar to Naive Bayes, but different model/learning}
  \]
  \[
  P(y = 1|x) \geq 0.5 \Leftrightarrow w^\top x \geq 0
  \]

- Learning: gradient ascent on the (regularized) discriminative log-likelihood.
Perceptron

- Simple error-driven learning approach similar to logistic regression
- Decision rule: $w^T f(x) > 0$
  - 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, but does it learn a good boundary?

Logistic Regression

$$w \leftarrow w + x(1 - P(y = 1|x))$$

$$w \leftarrow w - xP(y = 1|x)$$

Support Vector Machines

- Many separating hyperplanes — is there a best one?

Support Vector Machines

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

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

As a single constraint:

$$\forall j \quad y_j(w^T x_j) + (1 - y_j)(-w^T x_j) \geq 1$$

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

- What’s wrong with this quadratic program for real data?
- Data is generally non-separable — need slack!
N-Slack SVMs

\[
\text{Minimize } \lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j \\
\text{s.t. } \forall j \ (2y_j - 1)(w^T x_j) \geq 1 - \xi_j \\
\forall j \ \xi_j \geq 0
\]

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

Optimization — next time...

- Haven’t talked about optimization at all
- Range of techniques from simple gradient descent (works pretty well) to more complex methods (can work better)

Sentiment Analysis

- Classify sentence as positive or negative sentiment
  - 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>16696</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>76.1</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)

---

**Sentiment Analysis**

<table>
<thead>
<tr>
<th>Method</th>
<th>RT+</th>
<th>MPQA</th>
<th>CR</th>
<th>Subj.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB-uni</td>
<td>77.9</td>
<td>85.3</td>
<td>79.8</td>
<td>92.6</td>
</tr>
<tr>
<td>MNB-bi</td>
<td><strong>79.0</strong></td>
<td><strong>86.3</strong></td>
<td><strong>80.0</strong></td>
<td><strong>93.6</strong></td>
</tr>
<tr>
<td>SVM-uni</td>
<td>76.2</td>
<td>86.1</td>
<td>79.0</td>
<td>90.8</td>
</tr>
<tr>
<td>SVM-bi</td>
<td>77.7</td>
<td><strong>86.7</strong></td>
<td>80.8</td>
<td>91.7</td>
</tr>
<tr>
<td>NBSVM-uni</td>
<td><strong>78.1</strong></td>
<td>85.3</td>
<td>80.5</td>
<td>92.4</td>
</tr>
<tr>
<td>NBSVM-bi</td>
<td><strong>79.4</strong></td>
<td><strong>86.3</strong></td>
<td><strong>81.8</strong></td>
<td><strong>93.2</strong></td>
</tr>
<tr>
<td>RAE</td>
<td>76.8</td>
<td>85.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RAE-pretrain</td>
<td>77.7</td>
<td><strong>86.4</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Voting-w/Rev.</td>
<td>63.1</td>
<td>81.7</td>
<td>74.3</td>
<td>-</td>
</tr>
<tr>
<td>Rule</td>
<td>62.9</td>
<td>81.8</td>
<td>74.3</td>
<td>-</td>
</tr>
<tr>
<td>BoF-noDic.</td>
<td>75.7</td>
<td>81.8</td>
<td>79.3</td>
<td>-</td>
</tr>
<tr>
<td>BoF-w/Rev.</td>
<td>76.4</td>
<td>84.1</td>
<td><strong>81.4</strong></td>
<td>-</td>
</tr>
<tr>
<td>Tree-CRF</td>
<td>77.3</td>
<td>86.1</td>
<td><strong>81.4</strong></td>
<td>-</td>
</tr>
<tr>
<td>BoWSVM</td>
<td>-</td>
<td>-</td>
<td>90.0</td>
<td>-</td>
</tr>
</tbody>
</table>

Wang and Manning (2012)

Naive Bayes is doing well!

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

Before neural nets had taken off — results weren’t that great

Two years later Kim (2014) with neural networks

---

**Recap**

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

  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”