CS 371N Lecture 2
Classification 1: Features, Perceptron

Announcements
- AI
- Reading notation != lecture notation

Today
- Classification (linear, binary)
- Feature extraction
- ML basics + Perceptron

Classification Points \( \vec{x} \) (for us: strings)

Label \( y \in \{-1, +1\} \)

Classifier maps \( \vec{x} \rightarrow y \)
Linear classifier: represented by a weight vector \( \overrightarrow{w} \in \mathbb{R}^n \)

Decision rule: \[
\overrightarrow{w}^T f(\overrightarrow{x}) > 0
\]

\( n=2 \) \[
\overrightarrow{w} \cdot f(\overrightarrow{x}) \quad \text{if} \quad D = +1
\]

\( \text{else} : -1 \)

\( \overrightarrow{w} = (2, 3) \)
Sentiment Analysis

\( \overline{x} = \) the movie was great!

1) Feature extraction
\( \overline{x} \rightarrow f(x) \in \mathbb{R}^n \)

2) Learning algorithm

Training set \( \{ (f(\overline{x}^{(i)}), y^{(i)}) \}_{i=1}^{D} \rightarrow \overline{w} \) learned weight vector

Today: cover #1
see Perceptron for #2
Feature Extraction

\( \bar{x} = \text{the movie was great} \)

What do we want? – reflect word order

- word frequency \( \checkmark \)
- parts of speech
- word meaning/proximity \( \times \times \)
- negation (not good) \( \times \)

Bag-of-words featurization

\[ [1 \ 0 \ 0 \ 1 \ 1 \ 1] \]

- Vocabulary \( \sim 10,000 \) words
- 1 if present
- sparse feature representation
weight vector: \( \mathbf{w} \in \mathbb{R}^n \) 

\[
\begin{bmatrix}
-0.1 & +0.2 & \ldots & +0.3 & \ldots & +10 & \ldots
\end{bmatrix}
\]

the a movie great

\[ w^T f(x) = w_{\text{the}} + w_{\text{movie}} + w_{\text{was}} + \]

"weighted voting" \( w_{\text{great}} \)

awesome and great have independent weights

Preprocessing:

1. Vocab selection:
   - Vector space is a fixed set of words
     - replace unseen words w/ \text{UNK}
     - have a weight for \text{UNK}
That movie... really it wasn't great!

was not great!

Typical tokenization

- break out punc.
- break out contractions
② Remove stopwords (the, of, be, etc.)
③ Lowercasing / stemming

D = \{ +3 \}

punc \bar{W} = \{ -7 \}

split punc \bar{W} = \{ +3 \}

18,000

15,000
Is lower casing always good? No!
- text messages
- capture names

Revisit “not awesome”

{ not awesome movie was }

bigram bag of words

sentence $\Rightarrow$ adjacent pairs

vocab is now huge (old vocab^2)

we can mix unigrams + bigrams in our feature space
"custom" feature space

\[
\begin{bmatrix}
\text{movie} \\
\text{was} \\
\text{awesome} \\
\text{great} \\
\text{not awesome} \\
\text{movie was }
\end{bmatrix}
\]

Machine Learning

Optimize parameters \( \overline{w} \) to fit some training data (labeled)

We want \( \overline{w} \) to make good predictions

\[
\text{loss} = \sum_{i=1}^{D} \text{loss}(\overline{x}^{(i)}, \overline{y}^{(i)}, \overline{w})
\]

"if we use \( \overline{w} \) as our weights, how badly do we mess up?"
Stochastic Gradient Descent

for t in range (0, epochs)
  for i in range (0, D)
      \( \overline{w} \leftarrow w - \alpha \frac{2}{2\overline{w}} \text{loss}(\overline{x}, (\overline{y}, \overline{w})) \)

step size
\( \approx 1 \)

Subtracting gradient of the loss =
finding a \( \overline{w} \) with lower loss

\[ \text{loss}(w) = w^2 \]
\[ \Rightarrow \quad \begin{array}{c}
\text{w=1} \\
\text{gradient}=2w \\
\text{w=-1} \\
\text{gradient}=-2
\end{array} \]
Perception (instance of SGD)

Initialize $\overrightarrow{w} = 0$

for $t$ in range (0, epochs)

for $i$ in range (0, D) (shuffle ends each epoch)

$y_{pred} \leftarrow \begin{cases} 
1 & \overrightarrow{w}^T f(\overrightarrow{x}^{(i)}) > 0 \\
-1 & \text{else}
\end{cases}$

$\overrightarrow{w} \leftarrow \begin{cases} 
\overrightarrow{w} & \text{if } y_{pred} = y^{(i)} \\
\overrightarrow{w} + \alpha f(\overrightarrow{x}^{(i)}) & \text{if } y^{(i)} = +1 \\
\overrightarrow{w} - \alpha f(\overrightarrow{x}^{(i)}) & \text{if } y^{(i)} = -1
\end{cases}$

Let $\alpha = 1$ for now
Our update rule is sparse

$$\|U\|\text{ vocab} = 10k \text{ words}$$

$$f(\bar{x}^{(i)}) \text{ has 4 words}$$

we only update 4 weights
Step size

$\text{loss} = w^2$

$\frac{\partial l}{\partial w} = 2w$

$w = 1$

$\Rightarrow \text{grad} = 2$

$\Rightarrow \text{update} = -2\alpha \times \text{step size}$

$\alpha = 1$

$\Rightarrow 1 - 2 = -1$

$w = -1$

$\Rightarrow \ldots 1$

$\alpha = 0.5 \checkmark$

For AI:

Schedule

Start with $\alpha = 1$

Then reduce

$\alpha = \frac{1}{t} + \text{epoch}$