Logistic Regression

Chapter 5

Classification

• **Learn**: $f: X \rightarrow Y$
  - $X$ – features
  - $Y$ – target classes
Generative vs. Discriminative Models

Generative

- Learn a model of the joint probability $p(d, c)$
- Use Bayes’ Rule to calculate $p(c|d)$
- Build a model of each class; given example, return the model most likely to have generated that example
- Examples: Naive Bayes, HMM

Discriminative

Naive Bayes Review

- Features = \{I hate love this book\}
- Training
  - I hate this book
  - Love this book
- What is $P(Y|X)$?
- Prior $p(Y)$
- Testing
  - hate book
- Different conditions
  - $a = 0$ (no smoothing)
  - $a = 1$ (smoothing)

$$P(Y) = \begin{bmatrix} 1/2 & 1/2 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$P(X|Y) = \begin{bmatrix} 1/4 & 1/4 & 0 & 1/4 & 1/4 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$P(Y|X) \propto \begin{bmatrix} 1/2 \times 1/4 \times 1/4 & 1/2 \times 0 \times 1/3 \end{bmatrix} = [1 \\ 0]$$

$$M = \begin{bmatrix} 2 & 2 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 \end{bmatrix}$$

$$P(X|Y) = \begin{bmatrix} 2/9 & 2/9 & 1/9 & 2/9 & 2/9 \\ 1/8 & 1/8 & 2/8 & 2/8 & 2/8 \end{bmatrix}$$

$$P(Y|X) \propto \begin{bmatrix} 1/2 \times 2/9 \times 2/9 & 1/2 \times 1/8 \times 2/8 \end{bmatrix} = [0.613 \ 0.387]$$
Generative vs. Discriminative Models

Generative

- Learn a model of the joint probability $p(d, c)$
- Use Bayes' Rule to calculate $p(c|d)$
- Build a model of each class; given example, return the model most likely to have generated that example
- Examples: Naive Bayes, HMM

Discriminative

- Model $p(c|d)$ directly
- Class is a function of document vector
- Find the exact function that minimizes classification errors on the training data
- Learn boundaries between classes
- Example: Logistic regression

Linear boundary

Slide from Drago Radev
Discriminative vs. Generative Classifiers

- Discriminative classifiers are generally more effective, since they directly optimize the classification accuracy. But
  - They are sensitive to the choice of features
    - Plus: easy to incorporate linguistic information
    - Minus: until neural networks, features extracted heuristically
  - Also, overfitting can happen if data is sparse
- Generative classifiers are the “opposite”
  - They directly model text, an unnecessarily harder problem than classification

Review

- Multiclass NB and Evaluation
- NB tailored to sentiment
- Generative vs discriminative classifiers
Assumptions of Discriminative Classifiers

- Data examples (documents) are represented as vectors of features (words, phrases, ngrams, etc).
- Looking for a function that maps each vector into a class.
- This function can be found by minimizing the errors on the training data (plus other various criteria).
- Different classifiers vary on what the function looks like, and how they find the function.

Linear Separators

\[ f(x) = \Theta X + b \]

where
\( \Theta \) is a vector of weights: \( w_1, \ldots, w_n \)
\( X \) is the input vector
\( b \) is a constant

Two dimensional space:
\[ w_1x_1 + w_2x_2 = b \]
In n-dimensional spaces:
\[ \Theta X = \sum_{i=1}^{n} w_i x_i = b \]
One can also add \( w_0 = -1, x_0 = b \) to account for bias
Pass output of \( f(x) \) to the sign function, mapping negative values to -1 and positive values to 1.
How to find the weights?

- Logistic regression is one method
- Training using optimization
  - Select values for w
  - Compute f(x)
  - Compare f(x) output to gold labels and compute loss
    - Cross-Entropy (Section 5.3)
  - Adjust w

What does a logistic regression model look like?

- Given document instance x and sentiment label y
- We can propose various features that we think will tell us whether y is + or -:
  - f1(x): Is the word “excellent” used in x?
  - f2(x): How many adjectives are used in x?
  - f3(x): How many words in x are from the positive list in our sentiment lexicon?
  - ...
- We then need some way to combine these features to help us predict y
A Feature Representation of the Input

![Diagram of feature representation]

<table>
<thead>
<tr>
<th>Var</th>
<th>Definition</th>
<th>Value in Fig. 5.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>count(positive lexicon) ∈ doc</td>
<td>3</td>
</tr>
<tr>
<td>$x_2$</td>
<td>count(negative lexicon) ∈ doc</td>
<td>2</td>
</tr>
<tr>
<td>$x_3$</td>
<td>( \begin{cases} 1 &amp; \text{if “no” ∈ doc} \ 0 &amp; \text{otherwise} \end{cases} )</td>
<td>1</td>
</tr>
<tr>
<td>$x_4$</td>
<td>count(1st and 2nd pronouns ∈ doc)</td>
<td>3</td>
</tr>
<tr>
<td>$x_5$</td>
<td>( \begin{cases} 1 &amp; \text{if “!” ∈ doc} \ 0 &amp; \text{otherwise} \end{cases} )</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>log(word count of doc)</td>
<td>( \ln(64) = 4.15 )</td>
</tr>
</tbody>
</table>

But where did the feature representation (and interactions) come from?

Classification Decision

\[ \hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases} \]

\[ p(+|x) = P(Y = 1|x) = \sigma(w \cdot x + b) = \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.15] + 0.1) = \sigma(0.805) = 0.69 \]  \( (5.6) \)

\[ p(-|x) = P(Y = 0|x) = 1 - \sigma(w \cdot x + b) = 0.31 \]

<table>
<thead>
<tr>
<th>Var</th>
<th>Definition</th>
<th>Value in Fig. 5.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>count(positive lexicon) ∈ doc</td>
<td>3</td>
</tr>
<tr>
<td>$x_2$</td>
<td>count(negative lexicon) ∈ doc</td>
<td>2</td>
</tr>
<tr>
<td>$x_3$</td>
<td>( \begin{cases} 1 &amp; \text{if “no” ∈ doc} \ 0 &amp; \text{otherwise} \end{cases} )</td>
<td>1</td>
</tr>
<tr>
<td>$x_4$</td>
<td>count(1st and 2nd pronouns ∈ doc)</td>
<td>3</td>
</tr>
<tr>
<td>$x_5$</td>
<td>( \begin{cases} 1 &amp; \text{if “!” ∈ doc} \ 0 &amp; \text{otherwise} \end{cases} )</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>log(word count of doc)</td>
<td>( \ln(64) = 4.15 )</td>
</tr>
</tbody>
</table>

But where did the weights and bias come from?
Motivating Logistic Regression, continued

- if \( f_1(x) + f_2(x) + \ldots + f_n(x) > \text{thresh} \): return + else return -
  - Problem: not all features are equally important

- if \( w_0 + w_1 f_1(x) + \ldots + w_n f_n(x) > 0 \): return + else return -
  - Problem: not probabilistic

- Apply sigmoid function

\[
 f(x) = \frac{1}{1 + e^{-x}}
\]

Using a loss function

- Training data
  - \( x_1 x_2 \ldots x_n \) (input)
  - \( y_1 y_2 \ldots y_n \) (labels)
- Algorithm that returns \( f(x) \) with predictions \( \hat{y} \)
- Loss function \( L(\hat{y}, y) \)
- Parameters of the learned function \((\Theta, b)\) set to minimize \( L \)
Logistic Regression

- An example of a discriminative classifier
- Input:
  - Training example pairs of $(\tilde{x}, y)$ where $\tilde{x}$ is the feature vector and $y$ is the label
- Goal:
  - Build a model that predicts the probability of the label
- Output:
  - Set of weights $\tilde{w}$ that maximizes likelihood of correct labels on training examples

Logistic Regression

- Similar to Naive Bayes (but discriminative!)
  - Features don’t have to be independent
- Examples of features
  - Anything of use
  - Linguistic and non-linguistic
  - Count of “good”
  - Count of “not good”
  - Sentence length
Classification using LR

- Compute the feature vector \( x \)
- Multiply with weight vector \( w \)
  \[
  z = \sum w_i x_i
  \]
- Compute the logistic sigmoid function
  \[
  f(z) = \frac{1}{1 + e^{-z}}
  \]

Examples

- Example 1
  \[
  x = (2, 1, 1, 1) \\
  w = (1, -1, -2, 3) \\
  z = 2 - 1 - 2 + 3 = 2 \\
  f(z) = 1/(1 + e^{-2})
  \]
- Example 2
  \[
  x = (2, 1, 0, 1) \\
  w = (0, 0, -3, 0) \\
  z = 0 \\
  f(z) = 1/(1 + e^{0}) = 1/2
  \]
Why Sigmoid?  
First, Linear Regression

- Regression used to fit a linear model to data where the dependent variable is continuous:

\[ Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon \]

- Given a set of points \((X_i, Y_i)\), we wish to find a linear function (or line in 2 dimensions) that “goes through” these points.

- In general, the points are not exactly aligned:
  - Find line that best fits the points

Error

- Error:
  - Observed value - Predicted value
Logistic Regression

- Regression used to fit a curve to data in which the dependent variable is binary, or dichotomous
- Example application: Medicine
  - We might want to predict response to treatment, where we might code survivors as 1 and those who don’t survive as 0

Example

Observations: For each value of SurvRate, the number of dots is the number of patients with that value of NewOut

Regression: Standard linear regression

Problem: extending the regression line a few units left or right along the X axis produces predicted probabilities that fall outside of [0,1]
A Better Solution

Regression Curve:
Sigmoid function!
(bounded by asymptotes $y=0$ and $y=1$)

FIGURE 15.8  More appropriate regression line for predicting outcome

Logistic Regression

$P(y = C | x)$
Constructing a Learning Algorithm

- The conditional data likelihood is the probability of the observed $Y$ values in the training data, conditioned on their corresponding $X$ values. We choose parameters $\mathbf{w}$ that satisfy

$$\mathbf{w} = \arg \max_{\mathbf{w}} \prod_i P(y^l | \mathbf{x}^l, \mathbf{w})$$

- where $\mathbf{w} = \langle w_0, w_1, \ldots, w_n \rangle$ is the vector of parameters to be estimated, $y^l$ denotes the observed value of $Y$ in the $l$th training example, and $\mathbf{x}^l$ denotes the observed value of $X$ in the $l$th training example.

Constructing a Learning Algorithm

- Equivalently, we can work with the log of the conditional likelihood:

$$\mathbf{w} = \arg \max_{\mathbf{w}} \sum_l \ln P(y^l | \mathbf{x}^l, \mathbf{w})$$

- This conditional data log likelihood, which we will denote $I(W)$ can be written as

$$I(\mathbf{w}) = \sum_l y^l \ln P(y^l = 1 | \mathbf{x}^l, \mathbf{w}) + (1 - y^l) \ln P(y^l = 0 | \mathbf{x}^l, \mathbf{w})$$

- Note here we are utilizing the fact that $Y$ can take only values 0 or 1, so only one of the two terms in the expression will be non-zero for any given $y^l$. 

\[ \cdot \quad 28 \]
Fitting LR by Gradient Descent

• Unfortunately, there is no closed form solution to maximizing $l(w)$ with respect to $w$. Therefore, one common approach is to use gradient descent
  – Beginning with initial weights of zero, we repeatedly update the weights
  – Details optional, see text, but should understand following concepts
    • Loss function
    • Gradient descent
      – Gradient, learning rate, mini-batch training
    • Regularization
      – Overfitting

Gradient Descent

Learning Rate:
• the magnitude of the amount to move is the slope (more generally, the gradient) weighted by the learning rate
• if too high, overshoot minimum
• if too low, take too long to learn
• common to begin high, then decrease
Some Practical Issues

- Feature representation
  - want all features to have similar value ranges
  - too many features? feature selection
- Efficiency
  - Stochastic Gradient Descent / Batching
- Over-fitting
  - Regularization
- Classifying more than two categories

Mini-batch training

- Stochastic gradient descent chooses a random example at a time
- To make movements less choppy, compute gradient over batches of training instances from training set of size m
  - If batch size is m, batch training
  - If batch size is 1, stochastic gradient descent
  - Otherwise, mini batch training (for efficiency)
Regularization

- Weight training can yield models that don’t generalize well to test data (i.e., that overfit to training data)
- To avoid overfitting, a regularization term (various options) is used to penalize large weights
  - L2 quadratic function of the weight values
  - L1 linear function of the weight values

Multinomial Logistic Regression

- More than two classes
- AKA softmax regression, maxent classifier
  - Instead of sigmoid, use “softmax function”
  - Instead of having just one set of weights and one set of features, different set of weights and feature vectors for each class label
  - Loss function changes too
Summary of Logistic Regression

- Learns the Conditional Probability Distribution $P(y|x)$
- Local Search.
  - Begins with initial weight vector.
  - Modifies it iteratively to maximize an objective function.
  - The objective function is the conditional log likelihood of the data – so the algorithm seeks the probability distribution $P(y|x)$ that is most likely given the data.

---

Two Phases

- Training
  - we train the system (specifically the weights $w$ and $b$), e.g., using stochastic gradient descent and the cross-entropy loss
- Test
  - Given a test example $x$ we compute $p(y | x)$ and return the higher probability label $y = 1$ or $y = 0$
Final Comments

• In general, NB and LR make different assumptions
  – NB: Features independent given class -> assumption on P(X|Y)
  – LR: Functional form of P(Y|X), no assumption on P(X|Y)

• LR is optimized
  – no closed-form solution

• LR is interpretable

Summary

• Logistic regression is a supervised machine learning classifier (discriminative)

• Use: LR extracts real-valued features from the input, multiplies each by a weight, sums them, and passes the sum through a sigmoid function to generate a probability. A threshold is used to make a decision

• Learning: The weights (vector w and bias b) are learned from a labeled training set via a loss function that must be minimized, e.g., by using (iterative) gradient descent to find the optimal weights, and regularization to avoid overfitting