Logistic Regression

Chapter 5
(5.1-5.2)

1

Classification

- **Learn**: f:X->Y
  - X – features
  - Y – target classes

2

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)

$$P(Y) = [1/2 \quad 1/2]$$

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

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

$$P(Y|X) \propto [1/2 \times 1/4 \times 1/4 \quad 1/2 \times 0 \times 1/3] = [1 \; 0]$$

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

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

$$P(Y|X) \propto [1/2 \times 2/9 \times 2/9 \quad 1/2 \times 1/8 \times 2/8] = [0.613 \; 0.387]$$

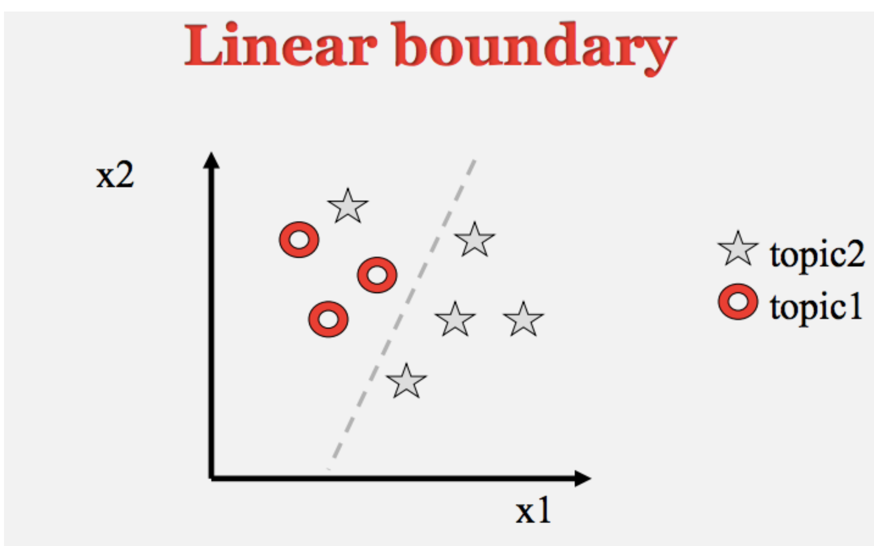
Generative vs. Discriminative Models

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



Slide from Drago Radev 6

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

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
Θ is a vector of weights: $w_1....w_n$
X is the input vector
b is a constant

Two dimensional space:
$w_1x_1+w_2x_2=b$
In n-dimensional spaces:
$\Theta X=\Sigma_{i=1}^{n}=w_1x_1+w_2x_2+....+w_nx_n$
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
  - Adjust w

10

Using a loss function

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

11

Logistic Regression

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

Logistic Regression

- Similar to Naive Bayes (but discriminative!)
  - Log-linear model
  - 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

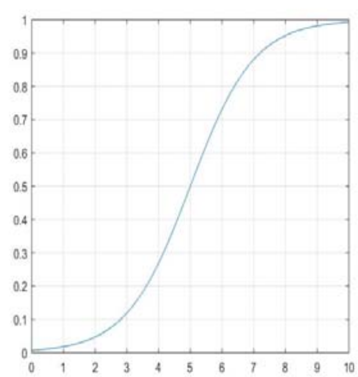
Classification using LR

- 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

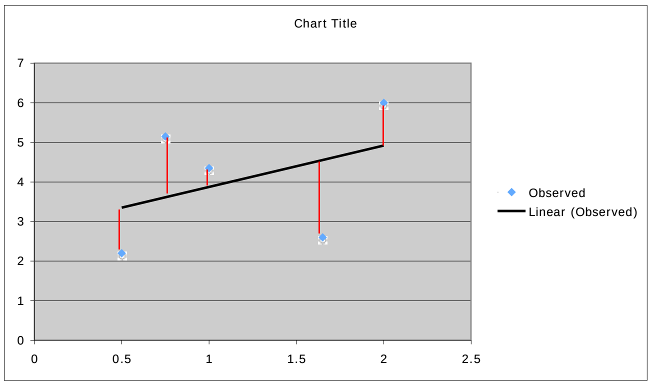
- Regression used to fit a linear model to data where the dependent variable is continuous:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon$$

- Given a set of points (Xi,Yi), 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

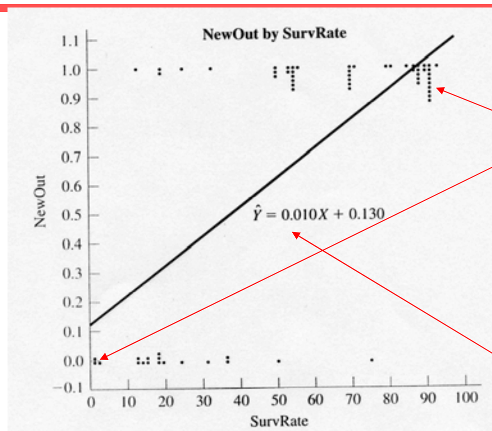


FIGURE 15.7 *Outcome as a function of SurvRate*

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

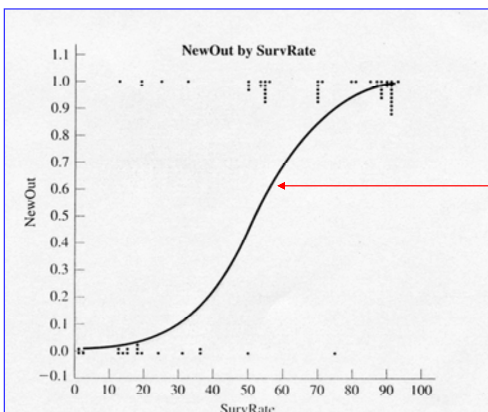
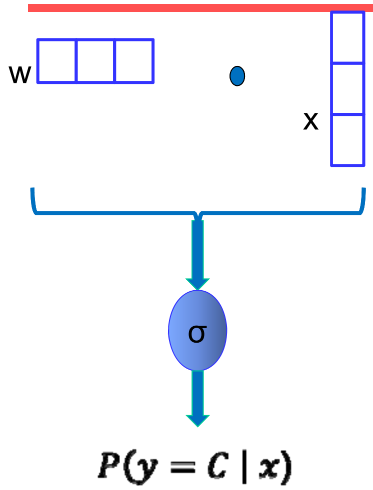


FIGURE 15.8 *More appropriate regression line for predicting outcome*

Regression Curve:
Sigmoid function!

(bounded by asymptotes *y*=0 and *y*=1)

Logistic Regression



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 w that satisfy

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

- where $\mathbf{w} = <w_0, w_1, \ldots, w_n>$ 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

•22

Constructing a Learning Algorithm

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

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

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

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

- 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

•23

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

•24

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.

•25

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

•26