Linear Models: Learning Goals

1. Regression
2. Regularization
3. Ridge and Lasso
4. Elastic Net
5. Logistic Regression

- Russell 18.6, Tibshirani 3.2-3.43, 4.4

Many Classification Approaches

- Decision trees
- Linear models, e.g., regression, ridge, lasso
- Neural Nets
- Naïve Bayes
- kNN
- SVM
- Ensembles
- Random forests
- Hidden Markov Models
- Conditional Random Fields

Why Linear Models?

- "Linear" = "fitting the data with a straight line/ surface(in higher dimensions)"
- A quadratic or high-order polynomial may lead to overfitting, particularly if training data are lacking
- Experience also shows that linear models show good approximation and easier to understand
- They have solid foundations in statistics and matrix algebra

Univariate Linear Regression

- Let us start with a single attribute with a label, both of which *numeric*: $(x_1,y_1), \ldots, (x_N,y_N)$
  - In decision trees, we deal with k-attributes, X_i, and Y_i categorical
  - Will return to those cases later
- Goal: find the "best" line $y = \beta_0 + \beta_1 x$
- Using least squares to define goodness

$$\text{optimum} = \text{argmin}_{\beta_0, \beta_1} \sum_{i=1}^{N} (y_i - (\beta_0 + \beta_1 x_i))^2$$

Univariate Linear Regression (2)

- Least squares has nice geometric interpretation: distance between y_i and $\hat{y}_i$, which is y_i projected onto the line
- It also has nice computational properties
  - Closed form solutions for the optimal β_0, β_1
  - The book gives details on the gradient descent method if the loss function is more general and has no closed form solutions (will return to this when neural net is discussed)

Multivariate Linear Regression

- Let us return to the k-attribute general situation $(X_1,y_1), \ldots, (X_N,y_N)$, where $X_i = [x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,k}]$
- Goal: find the "best" hyperplane $y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$
- Using least squares to define goodness

  $\text{argmin}_{\beta s} \sum_{i=1}^{N} (y_i - (\beta_0 + \sum_{j=1}^{k} \beta_j x_{i,j}))^2$
- Elegant closed-form matrix algebraic solution

  $\beta^* = (X^T X)^{-1} X^T y$

  $\hat{y} = X \beta^*$

where X the input data matrix, N rows, k columns (Tib Fig 3.1, 3.2)

Clicker Question

- Where did we see/use $X^T X$ before?

a) Not so far.

b) Clustering

c) PCA

d) Decision Trees

Shrinkage and Regularization

- This brings up an intricate connection with principal components
- Recall that the PCs are in descending order of variances
- Higher PCs have less "signals" anyways
- Why not drop them to make simpler models and minimizing overfitting?
- This leads to a method called principal component regression

Shrinkage and Regularization (2)

- But not so obvious how many PCs to keep
- Why not explicitly solve this problem mathematically:
  - Explicitly penalize for the size of the model, which is called regularization
  - Smoothly shrinks the contributions of those PCs with smaller variances

Ridge Regression

- Penalizing model size

  $$\text{argmin}_{\beta s} \{ \sum^{N}_{i=1} (y_i - (\beta_0 + \sum^{k}_{j=1} \beta_j x_{i,j}))^2 + \lambda \sum^{k}_{j=1} \beta_j^2 \}$$

- Which is equivalent to

  $$\text{argmin}_{\beta s} \{ \sum^{N}_{i=1} (y_i - (\beta_0 + \sum^{k}_{j=1} \beta_j x_{i,j}))^2 \}$$

  subject to $\sum^{k}_{j=1} \beta_j^2 \leq t$

- $\lambda \geq 0$, is the shrinkage parameter; the larger it is, the more shrinkage occurs

Ridge Regression (2)

- It also has an elegant algebraic solution
  $\beta^*_{ridge} = (X^T X + \lambda I)^{-1} X^T y$
- Essentially, ridge regression computes the coordinates of y but shrinks those coordinates by a factor of $d_j^2/(d_j^2 + \lambda)$, where the d's are singular values of X
- A smaller d_j is shrunk more because of λ
- Corresponding to the goal of shrinking higher PCs due to a lack of "signals"

Lasso

- Ridge regression shrinks non-essential dimensions towards 0 – but not yet to 0!
- Lasso goes the "full" distance but shrinking those to 0:
  - Replacing the $\lambda \sum^{k}_{j=1} \beta_j^2$ term to $\lambda \sum^{k}_{j=1} | \beta_j |$
  - Deliver the intended goal of forcing small non-zero coefficients to 0 (Tib Fig. 3.11) – making for simpler models
- However, there is no closed form solution, requiring quadratic programming

A Closer look on Correlated Attributes

- Lasso can pick "strong" (i.e., good fit to training data) but correlated attributes
- "Covering the bases": experience indicates that generalization accuracy may improve if non-correlated attributes are chosen
  - Particularly if there is a constraint on how many attributes can be selected

Ridge vs Lasso on Correlated Attributes

- Recall the penalty terms of Ridge and Lasso:

$$\lambda \sum_{j=1}^{k} \beta_j^2 \leq t_1 \quad \text{vs} \quad \lambda \sum_{j=1}^{k} |\beta_j| \leq t_2$$

- In the case of Ridge, the square term has the effect of letting the correlated attributes shrink towards each other
- In the case of Lasso, correlated attributes are arbitrarily chosen
  - This may hurt interpretation
  - Admission that Lasso may have gone too far in dropping attributes

Elastic Net: Blending Ridge + Lasso

- So why not combine the two, as in Elastic Net

$$\sum_{j=1}^{k} [\ \alpha\ |\ \beta_j\ |\ +\ (1 - \alpha)\ \beta_j^2\ ]$$

- i.e., a linear combination between the L_1 and L_2 penalty terms
- The first term encourages a simpler model, and the second term encourages all correlated attributes to be selected and *averaged*

Elastic Net (2)

- Thus, it has two parameters: λ controlling the level of penalties, and α controlling the degree to which all correlated attributes are selected
- CV can be used to tune the two parameters
- Shown to be quite effective in dealing with problems with a large number of correlated attributes, e.g., genomics, text

Clicker Question

- For what we have seen so far, which family of prediction problems are linear models good at?

a) Numeric attributes, discrete label
b) Numeric attributes, numeric label
c) Categorical attributes, discrete label
d) Categorical attributes, numeric label

Linear **Classification**

- So far we have discussed linear models for regression; let us return to classification
  - Our focus is two binary labels/discrete classes
  - Everything we discuss below generalizes to multiple classes (≥ 3)
- Imagine all the training examples in the attribute space
- We seek to partition the space into regions of a constant label (remember decision trees?)

Linearly Separable Classes

- A decision boundary is a curve that separates the two classes
- The boundary is called a linear separator if the boundary is a straight line or a surface in higher dimensions
- If the two classes can be separated by only linear separators, then the two classes are called linearly separable

Linearly Separable Classes (2)

- Let us focus on linear separable classes for now
- The textbook shows that using gradient descent, we can find *a* linear separator with the simple perceptron learning rule
- Later on, we try to find *the optimal* linear separator that maximizes the margin of separation

Logistic Regression: Motivation

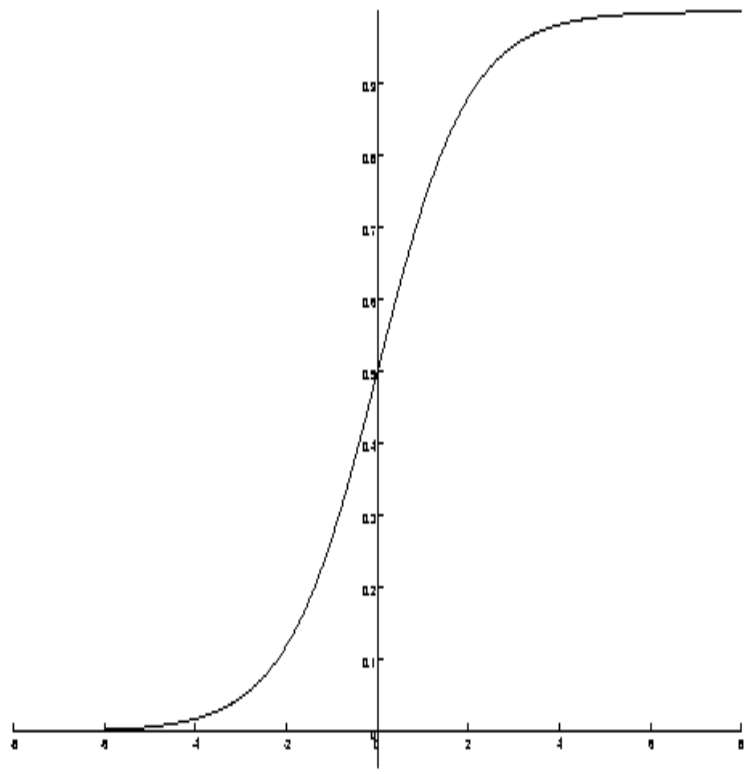
- However, finding a linear separator corresponds to using a *hard* threshold between the classes
- Examples close to the separator not stable
- We may want to "soften" the decision boundary with graduated predictions
- This leads us to probabilistic predictions

Logit Function

- "Softening" is achieved through

  $logit(z) = 1/(1 + exp(-z))$

  aka the *sigmoid* function
- Later on when we discuss fitting a line, the *signed* distance of an example from the line is translated into a *class probability*



The logistic function, 1/(1+exp(-x))

Logistic Regression (1)

- Consider optimal classification as the problem of finding class posterior probability Pr(L|X), i.e., given the attributes, predict the probabilities of the labels
- In 2-class logistic regression, we consider Pr (L=1 | X) and Pr(L=0|X)
- As these two posterior probabilities sum to 1, we fit their log odds ratio with a linear regression

  $\log [(\Pr(L=1|X) / \Pr(L=0|X)] = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$

  (the line refers to the situation when Pr = 0.5)

Logistic Regression: Model Fitting

- Optimal β's obtained by maximizing the *log conditional likelihood of L given X* over training data
- For N training examples: $(x_1,g_1), \ldots, (x_N,g_N)$ where x_i is the vector of features and g_i the label (i.e., 0 or 1) of the i-th example, the conditional likelihood for each example is $\Pr(L=g_i \mid X=x_i)$
- Assuming independent examples, the combined conditional likelihood is the product

  $\prod_{i=1}^{N} \Pr(L=g_i \mid X=x_i)$
- Log conditional likelihood is given by

  $\sum_{i=1}^{N} \log (\Pr(L=g_i \mid X=x_i))$

Logistic Regression: Model Fitting (2)

- For binary classification, let us simplify notation by focusing on the class L = 1, call Pr(L=1 | X) just p
- When gi =1, log Pr($L=g_i$ | $X=x_i$)) = log p = g_i log p
- When gi =0, log Pr($L=g_i$ | $X=x_i$)) = log (1- p) = (1 – g_i) * log (1- p)
- Thus, finding optimal β's to maximize conditional log likelihoods on training data becomes

  $\Sigma_{i=1}^{N}$ log (Pr($L=g_i$ | $X=x_i$))

  = $\Sigma_{i=1}^{N}$ [g_i log p + (1 – g_i) * log (1- p)]
- In essence, using the logit function to maximize the number of examples on the correct side of the boundary

Logistic Regression (3)

- Unfortunately, there is no closed form formula
- Gradient descent computation converges
- In fact, it works quite well even for non-linearly separable classes as well
- Quite popular as it generalizes to multiple classes
- We will return to logistic regression when we later discuss conditional random fields

Feature Selection

- The coefficients of a logistic regression indicate the "impact" of attributes
- But we also need to take the standard deviation of the attribute into account
- The Z-score is the coefficient divided by the standard deviation
- Rule of thumb: any attribute whose absolute Z-score is below 2 can be considered insignificant

Feature Selection (2)

- Following the spirit of regularization, we can drop one insignificant attribute at a time and re-fit a logistic regression model
- Until we run out of insignificant attributes
- There are more sophisticated techniques where, for example, we explicitly add the lasso penalty term to the logistic regression and optimize

Concluding Remarks: Big Data

- Linear methods are the most widely used in statistical modeling – from regression to classification
- Computationally, matrix algebra and gradient descent are the foundation
- Parallel algorithms have received a lot of attention in recent years
- So widely applicable to large datasets