Module 5: Classification

Basic Concepts: Risk and Measures of Predictive Accuracy

STAT/BIOSTAT 527, University of Washington
Emily Fox
May 23rd, 2013

The Optimal Prediction

- Assume we *know* the data-generating mechanism
- If our task is prediction, which summary of the distribution $Y \mid x$ should we report?

For x, what fcn $f(x)$ should we choose to predict Y if we can choose any $f(\cdot)$

- Taking a decision-theoretic framework, consider the ***expected loss*** (predictions are penalized by $L(\cdot,\cdot)$)

$$E_{X,Y}[L(Y, f(X))] = E_X\{E_{Y|X}[L(Y, f(x)) \mid X = x]\}$$

- $\hat{f}(\cdot)$ should min ↗
- can min. pointwise

Continuous Responses

- Expected loss $E_X \left\{ E_{Y|X} \left[L(Y, f(x)) \mid X = x \right] \right\}$
- Example: L_2 $L(Y, f(x)) = (Y - f(x))^2$

  Solution: $\hat{f}(x) = E[Y|x]$

- Example: L_1 $L(Y, f(x)) = |Y - f(x)|$

  Solution: $\hat{f}(x) = \text{median}(Y|x)$

- More generally: L_p $L(Y, f(x)) = \left\{ \int |Y - f(x)|^p \right\}^{1/p}$

Proofs: HW

Categorical Responses

- Expected loss $E_X \left\{ E_{Y|X} \left[L(Y, g(x)) \mid X = x \right] \right\}$
- Response:
- Same setup, but need new loss function
- Can always represent loss function with *K* x *K* matrix
- *L* is zeros on the diagonal and non-negative elsewhere
- Typical loss function:

Optimal Prediction

- Expected loss

$$E_X \left\{ E_{Y|X} \left[L(Y, g(x)) \mid X = x \right] \right\} =$$

- Again, can minimize pointwise

$$\hat{g}(x) =$$

  - Example: *K*=2

Optimal Prediction

$$\hat{g}(x) = \arg\min_g \sum_{k=1}^{K} L(\mathcal{G}_k, g) \Pr(\mathcal{G}_k \mid X = x)$$

- With 0-1 loss, we straightforwardly get the ***Bayes classifier***

$$\hat{g}(x) =$$

Optimal Prediction

$$\hat{g}(x) = \mathcal{G}_k \quad \text{if} \quad \Pr(\mathcal{G}_k \mid X = x) = \max_g \Pr(g \mid X = x)$$

- How to approximate the optimal prediction?
  - Don't actually have $p(Y \mid X = x)$
- Nearest neighbor approach
  - Look at *k*-nearest neighbors with majority vote to estimate

Optimal Prediction

$$\hat{g}(x) = \mathcal{G}_k \quad \text{if} \quad \Pr(\mathcal{G}_k \mid X = x) = \max_g \Pr(g \mid X = x)$$

- How to approximate the optimal prediction?
  - Don't actually have $p(Y \mid X = x)$
- Model-based approach
  - Introduce indicators for each class:
  - Consider squared-error loss: $\hat{f}(X) = E[Y \mid X]$
  - Bayes classifier is equivalent to standard regression and L_2 loss, followed by classification to largest fitted value
  - Works in theory, but not in practice…Will look at many other approaches (e.g., logistic regression)

Measuring Accuracy of Classifier

- For a given classifier, how do we assess how well it performs?
- For 0-1 loss, the generalization error is

  with empirical estimate

- Consider binary response and some useful summaries

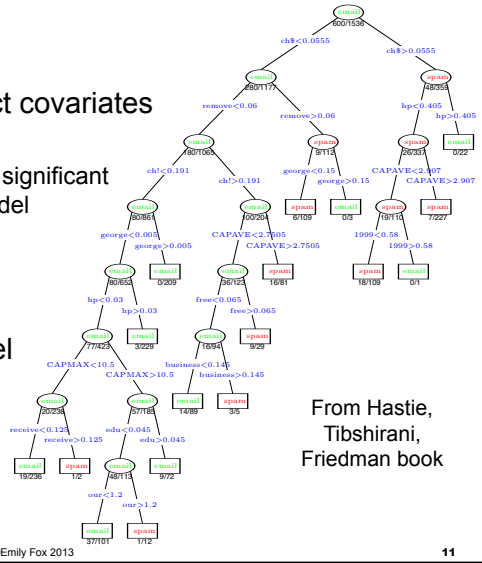
Measuring Accuracy of Classifier

- Sensitivity:
- Specificity:
- False positive rate:
- True positive rate:
- Connections:

Classification Tree Spam Example

- Resulting tree of size 17
- Note that there are 13 distinct covariates split on by the tree
  - 11 of these overlap with the 16 significant predictors from the additive model previously explored
- Overall error rate (9.3%) is higher than for additive model

	Predicted	
True	email	spam
email	57.3%	4.0%
spam	5.3%	33.4%



From Hastie, Tibshirani, Friedman book

Classification Tree Spam Example

- Think of ***spam*** and ***email*** as presence and absence of disease
- Using equal losses
  - Sensitivity =
  - Specificity =

	Predicted	
True	email	spam
email	57.3%	4.0%
spam	5.3%	33.4%

From Hastie, Tibshirani, Friedman book

- By varying L_{01} and L_{10}, can increase/decrease sensitivity and decrease/increase specificity of rule
- Which do we want here?
- How?
- Change in rule at leaf:

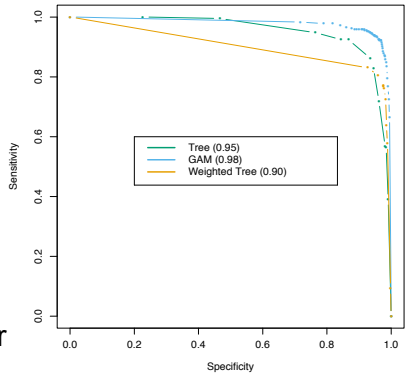
ROC Curves

- ***Receiver operating characteristic*** (ROC) curve summarizes tradeoff between sensitivity and specificity
  - Plot of sensitivity vs. specificity as a function of params of classification rule

- Example: vary L_{01} in [0.1,10]
  - Want specificity near 100%, but in this case sensitivity drops to about 50%

- Summary = area under the curve
  - Tree = 0.95
  - GAM = 0.98

- Instead of Bayes rule at leaf, better to account for unequal losses in constructing tree



From Hastie, Tibshirani, Friedman book

What you need to know

- Again, goal framed as minimizing expected loss

- Loss here is summarized by K x K matrix L
  - Common choice = 0-1 loss

- Bayes classifier chooses most probable class

- Measures of predictive performance:
  - Sensitivity, specificity, true positive rate, false positive rate
  - ROC curve and area under the curve

Readings

- Wakefield – 10.3.2, 10.4.2, 12.8.4
- Hastie, Tibshirani, Friedman – 9.2.3, 9.2.5, 2.4

Module 5: Classification

Linear Methods: Logistic Regression

STAT/BIOSTAT 527, University of Washington
Emily Fox
May 23rd, 2013

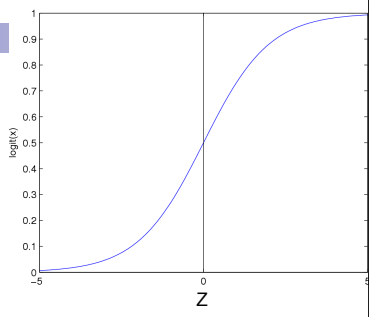
Link Functions

- Estimating p(*Y*|***X***): Why not use standard linear regression?

- Combing regression and probability?
  - Need a mapping from real values to [0,1]
  - A link function!

Logistic Regression

Logistic function (or Sigmoid): $\frac{1}{1+exp(-z)}$



- Learn p(*Y*|***X***) directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:

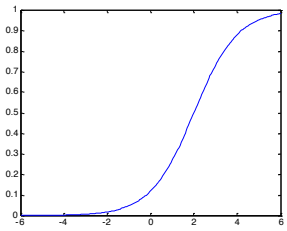
$$p(y=0 \mid x, \beta) = \frac{1}{1+\exp(\beta_0 + \sum_j \beta_j x_j)}$$

Covariates can be discrete or continuous!

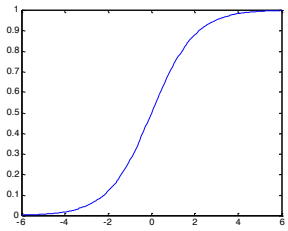
Understanding the Sigmoid

$$g(\beta_0 + \sum_j \beta_j x_j) = \frac{1}{1 + \exp(\beta_0 + \sum_j \beta_j x_j)}$$

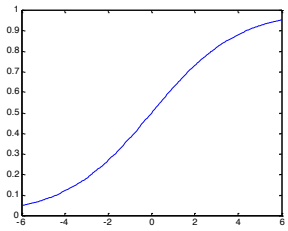
β_0=-2, β_1=-1



β_0=0, β_1=-1

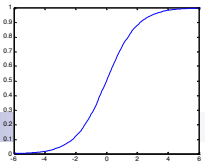


β_0=0, β_1=-0.5



Logistic Regression – a Linear classifier

$$\frac{1}{1 + exp(-z)}$$



$$g(\beta_0 + \sum_j \beta_j x_j) = \frac{1}{1 + \exp(\beta_0 + \sum_j \beta_j x_j)}$$

Very convenient!

$$p(y = 0 \mid x, \beta) = \frac{1}{1 + \exp(\beta_0 + \sum_j \beta_j x_j)}$$

implies

$$p(y = 1 \mid x, \beta) = \frac{\exp(\beta_0 + \sum_j \beta_j x_j)}{1 + \exp(\beta_0 + \sum_j \beta_j x_j)}$$

Examine ratio:

$$\frac{p(y = 1 \mid x, \beta)}{p(y = 0 \mid x, \beta)} = \exp(\beta_0 + \sum_j \beta_j x_j)$$

implies

$$\log \frac{p(y = 1 \mid x, \beta)}{p(y = 0 \mid x, \beta)} = \beta_0 + \sum_j \beta_j x_j$$

linear classification rule!

Loss Function: Conditional Likelihood

- Have a bunch of iid data of the form:

- Discriminative (logistic regression) loss function:
  Conditional Data Likelihood

$$\log p(D_Y \mid D_X, \beta) = \sum_{i=1}^{n} \log p(y_i \mid x_i, \beta)$$

Expressing Conditional Log Likelihood

$$p(y = 0 \mid x, \beta) = \frac{1}{1 + \exp(\beta_0 + \sum_j \beta_j x_j)}$$

$$p(y = 1 \mid x, \beta) = \frac{\exp(\beta_0 + \sum_j \beta_j x_j)}{1 + \exp(\beta_0 + \sum_j \beta_j x_j)}$$

$$l(\beta) = \sum_i \log p(y_i \mid x_i, \beta)$$

$$l(\beta) = \sum_i y_i \log p(y = 1 \mid x_i, \beta) + (1 - y_i) \log p(y = 0 \mid x_i, \beta)$$

Maximizing Conditional Log Likelihood

$$p(y = 0 \mid x, \beta) = \frac{1}{1 + \exp(\beta_0 + \sum_j \beta_j x_j)}$$

$$p(y = 1 \mid x, \beta) = \frac{\exp(\beta_0 + \sum_j \beta_j x_j)}{1 + \exp(\beta_0 + \sum_j \beta_j x_j)}$$

$$l(\beta) = \sum_i \log p(y_i \mid x_i, \beta)$$

$$= \sum_i y_i(\beta_0 + \sum_j \beta_j x_{ij}) - \log(1 + \exp(\beta_0 + \sum_j \beta_j x_{ij})$$

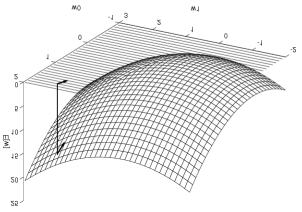
Good news: $l(\boldsymbol{\beta})$ is concave function of $\boldsymbol{\beta}$, no local optima problems

Bad news: no closed-form solution to maximize $l(\boldsymbol{\beta})$

Good news: concave functions easy to optimize

Optimizing Concave Function – Gradient Ascent

- Conditional likelihood for logistic regression is concave
- Find optimum with gradient ascent



Gradient: $\nabla_\beta l(\beta) = \left[\frac{\partial l(\beta)}{\partial \beta_0}, \ldots, \frac{\partial l(\beta)}{\partial \beta_d}\right]'$

Step size, $\eta > 0$

Update rule: $\Delta\beta = \eta \nabla_\beta l(\beta)$

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \frac{\partial l(\beta)}{\partial \beta_j}$$

- Gradient ascent is simplest of optimization approaches
  - e.g., Conjugate gradient ascent can be much better

Maximize Conditional Log Likelihood: Gradient ascent

$$l(\beta) = \sum_i y_i(\beta_0 + \sum_j \beta_j x_{ij}) - \log(1 + \exp(\beta_0 + \sum_j \beta_j x_{ij}))$$

Gradient Ascent for LR

Gradient ascent algorithm: iterate until change < ε

$$\beta_0^{(t+1)} \leftarrow \beta_0^{(t)} + \eta \sum_i \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right)$$

For *j*=1,…,*d*,

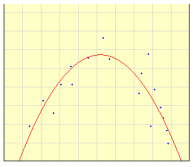
$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \sum_i x_{ij} \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right)$$

repeat

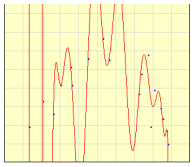
Regularization in Linear Regression

- Overfitting usually leads to very large parameter choices, e.g.:

-2.2 + 3.1 X – 0.30 X^2



-1.1 + 4,700,910.7 X – 8,585,638.4 X^2 + …

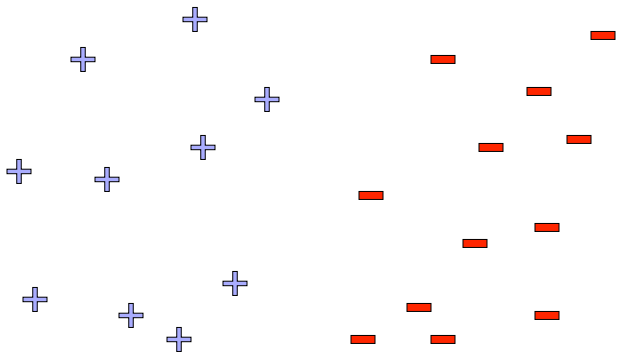


even for n ≈ p, p large

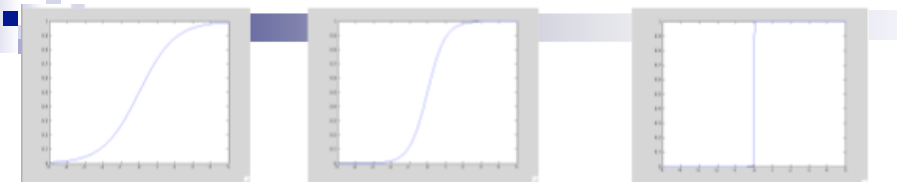
- ***Regularized*** or ***penalized*** regression aims to impose a "complexity" penalty by penalizing large weights
  - "Shrinkage" method

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta^T x_i))^2 + \lambda ||\beta||$$

Linear Separability



Large Parameters → Overfitting



$$\frac{1}{1+e^{-x}} \qquad \frac{1}{1+e^{-2x}} \qquad \frac{1}{1+e^{-100x}}$$

- If data is linearly separable, weights go to infinity
  - In general, leads to overfitting:
- Penalizing high weights can prevent overfitting…

Regularized Conditional Log Likelihood

- Add regularization penalty, e.g., L_2:

$$l(\beta) = \log \prod_{i=1}^{n} p(y_i \mid x_i, \beta) - \frac{\lambda}{2}||\beta||_2^2$$

- Practical note about β_0:

- Gradient of regularized likelihood:

Standard v. Regularized Updates

- Maximum conditional likelihood estimate

$$\hat{\beta} = \arg\max_{\beta} \log \prod_{i=1}^{n} p(y_i \mid x_i, \beta)$$

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \sum_i x_{ij} \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right)$$

- Regularized maximum conditional likelihood estimate

$$\hat{\beta} = \arg\max_{\beta} \log \prod_{i=1}^{n} p(y_i \mid x_i, \beta) - \frac{\lambda}{2} \sum_{j=1}^{d} \beta_j^2$$

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \left\{ -\lambda \beta_j^{(t)} + \sum_i x_{ij} \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right) \right\}$$

Stopping Criterion

$$l(\beta) = \log \prod_{i=1}^{n} p(y_i \mid x_i, \beta) - \frac{\lambda}{2}||\beta||_2^2$$

- When do we stop doing gradient ascent?
- Because *l*(**w**) is strongly concave:
  - i.e., because of some technical condition

$$l(\beta^*) - l(\beta) \leq \frac{1}{2\lambda}||\nabla l(\beta)||_2^2$$

- Thus, stop when:

Digression: Logistic Regression for *K* > 2

- Logistic regression in more general case (*K* classes), where *Y* in {1,…,*K*}

Digression: Logistic Regression for *K* > 2

- Logistic regression in more general case, where *Y in* {1,…,*K*}

for *k*<*K*

$$p(y = k|\mathbf{x}, \beta) = \frac{\exp(\beta_{k0} + \sum_{j=1}^{d} \beta_{kj} x_j)}{1 + \sum_{k'=1}^{K-1} \exp(\beta_{k'0} + \sum_{j=1}^{d} \beta_{k'j} x_j)}$$

for *k*=*K* (normalization, so no weights for this class)

$$p(y = K|\mathbf{x}, \beta) = \frac{1}{1 + \sum_{k'=1}^{K-1} \exp(\beta_{k'0} + \sum_{j=1}^{d} \beta_{k'j} x_j)}$$

Estimation procedure is basically the same as what we derived!

The Cost, The Cost!!! Think about the cost…

- What's the cost of a gradient update step for LR???

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \left\{ -\lambda \beta_j^{(t)} + \sum_i x_{ij} \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right) \right\}$$

Gradient ascent in Terms of Expectations

- "True" objective function:

$$l(\beta) = E_x[l(\beta, x)] = \int p(x) l(\beta, x) dx$$

- Taking the gradient:

- "True" gradient ascent rule:

- How do we estimate expected gradient?

SGD: Stochastic Gradient Ascent (or Descent)

- "True" gradient: $\nabla l(\beta) = E_x[\nabla l(\beta, x)]$

- Sample based approximation:

- What if we estimate gradient with just one sample???
  - Unbiased estimate of gradient
  - Very noisy!
  - Called stochastic gradient ascent (or descent)
    - Among many other names
  - VERY useful in practice!!!

Stochastic Gradient Ascent for Logistic Regression

- Logistic loss as a stochastic function:

$$E_x[l(\beta, x)] = E_x\left[\log p(y \mid x, \beta) - \frac{\lambda}{2}||\beta||_2^2\right]$$

- Batch gradient ascent updates:

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta\left\{-\lambda\beta_j^{(t)} + \frac{1}{n}\sum_{i=1}^{n} x_{ij}\left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)})\right)\right\}$$

- Stochastic gradient ascent updates:
  - Online setting:

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta\left\{-\lambda\beta_j^{(t)} + x_{i(t),j}\left(y_{i(t)} - \hat{p}(y = 1 \mid x_{i(t)}, \beta^{(t)})\right)\right\}$$

What you should know…

- Classification: predict discrete classes rather than real values
- Logistic regression model: Linear model
  - Logistic function maps real values to [0,1]
- Optimize conditional likelihood
- Gradient computation
- Overfitting
- Regularization
- Regularized optimization
- Cost of gradient step is high, use stochastic gradient descent