Module 5: Classification

Basic Concepts: Risk and Measures of Predictive Accuracy

STAT/BIOSTAT 527, University of Washington
Emily Fox
May 20th, 2014

The Optimal Prediction

- Assume we *know* the data-generating mechanism $Y|X$
- 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 $L(\cdot,\cdot)$

$$E_{X,Y}[L(\underbrace{Y}_{\text{truth}}, \underbrace{f(X)}_{\text{pred.}})] = E_X\{E_{Y|X}[L(Y,f(x)) \mid X=x]\}$$

- $\hat{f}(\cdot)$ 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 $L(Y, f(x)) = (Y - f(x))^2$ focus so far

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

  Proofs: HW

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

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

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

Categorical Responses $L(k, j)$

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

- Response: $Y \in \{G_1, \ldots, G_K\}$ ← # of classes

- Same setup, but need new loss function
- Can always represent loss function with *K* x *K* matrix

  $L_{jk} = L(j, k)$ truth, pred. class $= \begin{cases} 0, & j = k \\ \geq 0, & j \neq k \end{cases}$

- *L* is zeros on the diagonal and non-negative elsewhere
- Typical loss function:

  $L_{jk} = L(j, k) = \begin{cases} 0, & j = k \\ 1, & j \neq k \end{cases}$ unit cost for all poss. mistakes

Optimal Prediction

$E(X) = \sum_x x\, P(X=x)$

- Expected loss

$$E_X\left\{E_{Y|X}\left[L(Y,g(x)) \mid X=x\right]\right\} = E_X\left(\sum_{k=1}^{K} L(G_k, g(x)) \cdot P(G_k \mid X=x)\right)$$

- Again, can minimize pointwise

$$\hat{g}(x) = \arg\min_g \sum_{k=1}^{K} L(g_k, g)\, P(G_k \mid X=x)$$

  - Example: K=2

$(g=0) \qquad\qquad (g=1)$

$$L(1,0)P(G_1|x) + \cancel{L(0,0)P(G_0|x)} > \cancel{L(1,1)P(G_1|x)} + L(0,1)P(G_0|x)$$

$$L(1,0)P(G_1|x) > L(0,1)[1 - P(G_1|x)]$$

$$P(G_1|x) > \frac{L(0,1)}{L(0,1) + L(1,0)}$$

$\hat{g}(x) = 1$

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***

$$\hat{g}(x) = \arg\min_g\, [1 - P(g \mid X=x)] \quad \text{(general)}$$

OR

$$\hat{g}(x) = G_k \text{ if } P(G_k \mid X=x) = \max_g P(g \mid X=x)$$

(classify to most probable class)

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

before (L_2)

$E[Y|X] = \text{avg}(y_i)\ x_i \in \text{nbhd}(x)$

$$P(G_m | X=x) \approx \frac{1}{K} \sum_{x_i \in \text{nbhd}(x)} \mathbb{I}(y_i = m)$$

classify w/ largest $P(G_m|X=x)$ label (most common of k-NN)

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: $Y = [0\,0\,1\,0\,0 \ldots 0]$ $Y_k = 1$ if $Y = G_k$
  - Consider squared-error loss: $\hat{f}(X) = E[Y \mid X]$

$$E[Y_k | X] = P(Y = G_k | X)$$

  - Bayes classifier is equivalent to standard regression and L_2 loss, followed by classification to largest fitted value

$$\hat{f}(x) = \begin{bmatrix} 0.1 \\ 0.02 \\ \vdots \end{bmatrix} \text{largest} \rightarrow \text{class}$$

  - 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

$$E_{X,Y}[g(x) \neq Y] = P_{XY}(g(x) \neq Y)$$

with empirical estimate

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(g(x_i) \neq y_i)$$ ← chosen classifier

- Consider binary response and some useful summaries

$$y = \begin{cases} 0 & \text{no disease} \\ 1 & \text{disease} \end{cases}$$

decision rule $g(x) = \begin{cases} 0 & \text{predict no disease} \\ 1 & \text{predict disease} \end{cases}$

Measuring Accuracy of Classifier

- Sensitivity: prob. of pred. disease for a diseased individual

$$P(G(x)=1 \mid Y=1)$$

- Specificity: " " no disease given individual's not diseased

$$P(G(x)=0 \mid Y=0)$$

- False positive rate:

$$P(g(x)=1 \mid Y=0)$$

- True positive rate:

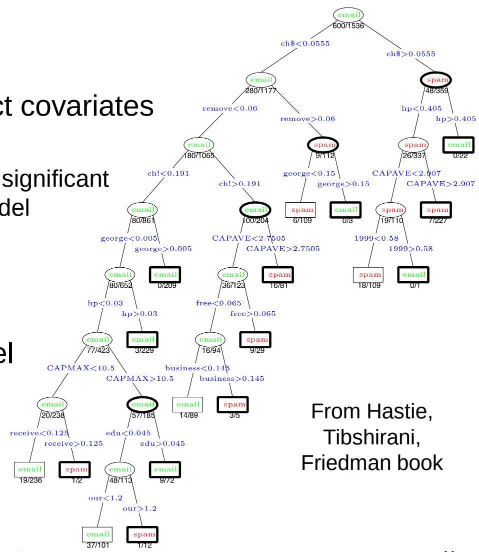
$$P(g(x)=1 \mid Y=1)$$

- Connections:

sensitivity = TPR , specificity = 1 - FPR

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



True	Predicted email	Predicted 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 = $100 \times \frac{33.4}{33.4+5.3} = 86.3\%$
  - Specificity = $100 \times \frac{57.3}{57.3+4.0} = 93.4\%$

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

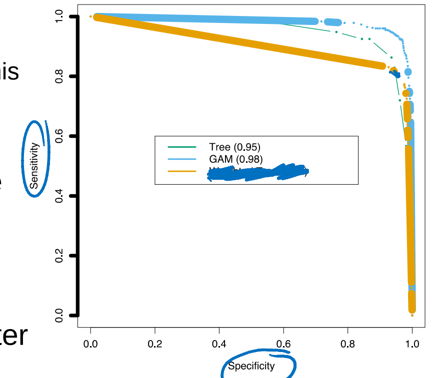
From Hastie, Tibshirani, Friedman book

- By varying L_{01} and L_{10}, can increase/decrease sensitivity and decrease/increase specificity of rule
- Which do we want here? avoid marking 'email' as 'spam' ↑0 ↑!
- How? $L_{01} > L_{10} = 1$ .... high specificity
- Change in rule at leaf:

predict 'spam' if proport. of 'spam' at leaf $\geq \frac{L_{01}}{L_{10}+L_{01}}$

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 ... see book



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 (K classes)
  - Common choice = 0-1 loss
- Bayes classifier chooses most probable class (intuitive)
- 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 20th, 2014

Link Functions

Focus on $Y \in \{0,1\}$

or generally $Y \in \{1,\dots,k\}$

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

$$\underbrace{P(Y|X)}_{\in[0,1]} = \underbrace{\beta_0 + \sum_j \beta_j h_j(x)}_{\text{range: } (-\infty,\infty)}$$

BAD

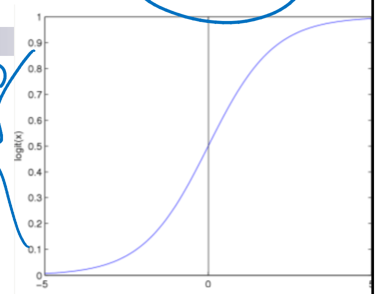
- Combing regression and probability?
  - Need a mapping from real values to [0,1]
  - A link function!

$g: \mathbb{R} \to [0,1]$

many options, but here's a useful one-

Logistic Regression

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



$(0,1)$

$\mathbb{R}$

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

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

choice/label

z: linear just like in regression

$(-\infty,\infty)$

$P(Y=1|X) = \dfrac{e^{(\dots)}}{1+e^{(\dots)}}$

$\beta_0 + \sum_j \beta_j x_j$

not bounded, could be neg.

after applying the sigmoid, output is in $[0,1]$

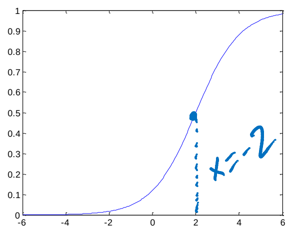
Covariates can be discrete or continuous!

Understanding the Sigmoid

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

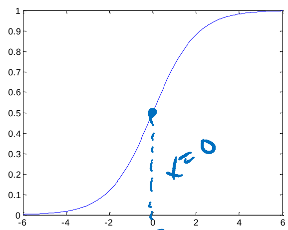
$\swarrow$ $d=1$

β_0=-2, β_1=-1



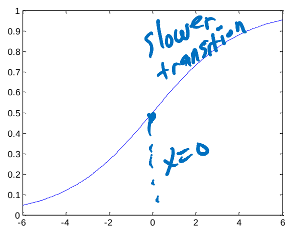
z

β_0=0, β_1=-1



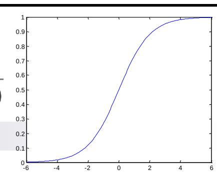
z

β_0=0, β_1=-0.5



Logistic Regression – a Linear classifier

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



$P(Y=0|X,\beta)$

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

$\beta_0 + \sum_j \beta_j x_j > 0$

$\rightarrow g(\text{" "}) < 0.5$

$\rightarrow P(Y=0) < 0.5$

$\rightarrow$ predict class 1

$\beta_0 + \sum_j \beta_j x_j = 0$ hyperplane

$(\text{" "} < 0)$

$\rightarrow g(\cdot) > 0.5$

$\rightarrow$ predict class 0

Very convenient!

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

implies

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

$1 - p(y=0)$

Examine ratio:

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

class 1 wins
else class 0
(under 0-1 loss)

linear classification rule!

implies

$$\underbrace{\log \frac{p(y = 1 \mid x, \beta)}{p(y = 0 \mid x, \beta)}}_{\text{log odds}} = \underbrace{\beta_0 + \sum_j \beta_j x_j}_{\text{linear}} > 0 \rightarrow \text{class 1 wins!}$$

Loss Function: Conditional Likelihood

- Have a bunch of iid data of the form:

x	y
	0
	1
	0

(x_1, y_1)
$\vdots$

(x_i, y_i) iid $i = 1, \dots, n$
$= (D_x, D_y)$

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

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

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

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

Expressing Conditional Log Likelihood

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

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

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

$$= \sum_i \begin{cases} \log p(y=1 \mid x_i,\beta) & \text{if } y_i=1 \\ \log p(y=0 \mid x_i,\beta) & \text{if } y_i=0 \end{cases}$$

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

$$= \sum_i y_i \log\left(\frac{\exp(\beta_0+\sum_j \beta_j x_j)}{1+\exp(\beta_0+\sum_j \beta_j x_j)}\right) + (1-y_i)\log\left(\frac{1}{1+\exp(\cdots)}\right)$$

$$= \sum_i y_i\left(\beta_0+\sum \beta_j x_j\right) - \log\left(1+\exp(\cdots)\right)$$

Maximizing Conditional Log Likelihood

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

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

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

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

↑fixed

parameters to opt.

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

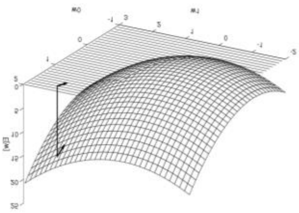
$x \in \mathbb{R}$

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

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, η>0

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

$$\underset{\text{newest.}}{\beta_j^{(t+1)}} \leftarrow \underset{\text{oldest.}}{\beta_j^{(t)}} + \eta \frac{\partial l(\beta)}{\partial \beta_j}$$

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

often, η gets smaller w/ iteration

$\eta \propto \frac{k}{t}$ (const "t" ...)

Maximize Conditional Log Likelihood: Gradient ascent

$\frac{\partial}{\partial s} \ln f(s) = \frac{f'(s)}{f(s)}$ $\quad$ $\frac{\partial}{\partial s} e^{f(s)} = f'(s)\, e^{f(s)}$

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

$\nabla l(\beta)$: $\frac{\partial l(\beta)}{\partial \beta_j} = \sum_{i=1} y_i x_{ij} - \frac{x_{ij} \exp(\ldots)}{1+\exp(\ldots)}$ $\quad (j>0)$

$= \sum_i x_{ij} \left(y_i - \hat{p}(y=1|x_i,\beta)\right)$ $\quad p(y=1|x,\beta)$

weighted by contribution of jth covariate to pred.

how far is my pred. from truth

Gradient Ascent for LR

start w/ $\beta^{(0)} = 0$

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*,

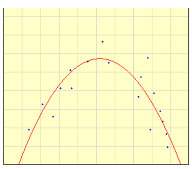
$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \sum_i x_{ij} \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right)$$

repeat

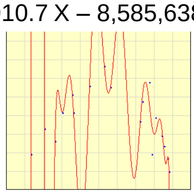
Regularization in Linear Regression

- 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 + …

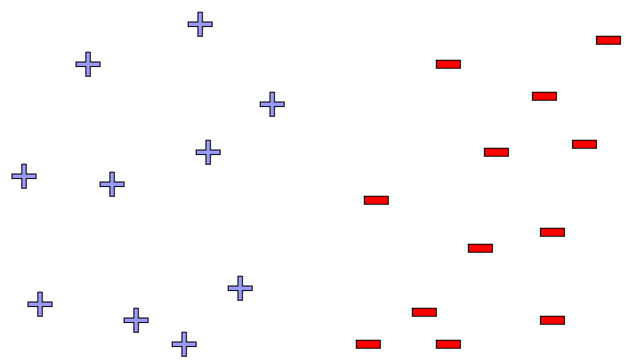


even for $n > p$, p large

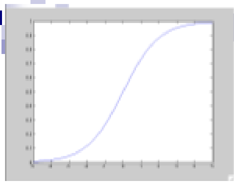
- ***Regularized*** or ***penalized*** regression aims to impose a "complexity" penalty by penalizing large weights
  - "Shrinkage" method

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

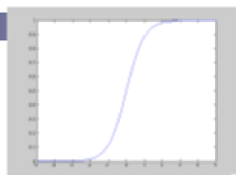
Linear Separability



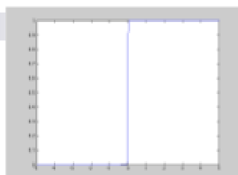
Large Parameters → Overfitting



$$\frac{1}{1+e^{-x}}$$



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



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

- 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

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

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

- Regularized maximum conditional likelihood estimate

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

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

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???

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

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:

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

- Stochastic gradient ascent updates:
  - Online setting:

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

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