Announcements:

- HW2 posted today.

Today:
- Review: Logistic Regression
- New: GD, SGD

THUS FAR, REGRESSION: PREDICT A CONTINUOUS VALUE GIVEN SOME INPUTS
Weather prediction revisited

Reading Your Brain, Simple Example

Pairwise classification accuracy: 85%

Person

Animal

Classification

- Learn: \( h: X \rightarrow Y \)
  - \( X \) – features
  - \( Y \) – target classes

- Conditional probability: \( P(Y|X) \)

- Suppose you know \( P(Y|X) \) exactly, how should you classify?
  - Bayes optimal classifier:
    \[
    
    \arg\max_{y} \ P(Y|X) 
    \]

- How do we estimate \( P(Y|X) \)?

Link Functions

- Estimating \( P(Y|X) \): Why not use standard linear regression?
  \[
  y \approx \omega_0 + \sum \omega_i x_i 
  \]

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

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

$$P(Y = 0 | X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)}$$

Features can be discrete or continuous!

Understanding the sigmoid

$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)}$$

Very convenient!

$$P(Y = 0 | X = \langle X_1, ..., X_n \rangle) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

implies

$$P(Y = 1 | X = \langle X_1, ..., X_n \rangle) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

implies

$$P(Y = 1 | X) = \exp(w_0 + \sum_i w_i X_i)$$

implies

$$\ln \frac{P(Y = 1 | X)}{P(Y = 0 | X)} = w_0 + \sum_i w_i X_i$$
Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave. Find optimum with gradient ascent.

Gradient: \( \nabla_w l(w) = \left[ \frac{\partial l(w)}{\partial w_0}, \ldots, \frac{\partial l(w)}{\partial w_i} \right] \)

Update rule: \( \Delta w = \eta \nabla_w l(w) \)

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

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: \( x_1, \ldots, x_n \)

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

\[
\ell(w) = \sum_{j} y_j \ln \left( \frac{e^{x_j^T w}}{1 + e^{x_j^T w}} \right) + (1 - y_j) \ln \left( 1 + e^{x_j^T w} \right)
\]

Expressing Conditional Log Likelihood

\[
l(w) = \ln \prod_{j} P(Y = y_j | x_j, w)
\]

Maximizing Conditional Log Likelihood

\[
l(w) = \ln \prod_{j} P(Y = y_j | x_j, w) = \sum_{j} y_j \left( w_0 + \sum_{i} w_i x_{i,j} \right) - \ln(1 + \exp(w_0 + \sum_{i} w_i x_{i,j}))
\]

\[
l(w) \text{ is concave function of } w, \text{ no local optima problems}
\]

\[
\text{Bad news: no closed-form solution to maximize } l(w)
\]

\[
\text{Good news: concave functions easy to optimize}
\]
Maximize Conditional Log Likelihood:
Gradient ascent

\[
\ell(w) = \sum_j y_j (w_0 + \sum_i w_i x_{i,k}) - \ln(1 + \exp(w_0 + \sum_i w_i x_{i,k}))
\]

\[
\frac{\partial \ell}{\partial w_i} = \sum_j \frac{x_{i,k} \exp(-w_0 - \sum_i w_i x_{i,k})}{1 + \exp(-w_0 - \sum_i w_i x_{i,k})}
\]

\[
= \sum_j x_{i,k} (y_j - \hat{P}(y_j = 1 | x_i, w))
\]

Gradient Ascent for LR

Gradient ascent algorithm: iterate until change < \(\varepsilon\)

\[
w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_y [y - \hat{P}(y = 1 | x, \hat{w})]
\]

For \(i = 1, \ldots, k,\)

\[
w_i^{(t+1)} = w_i^{(t)} + \eta \sum_y x_i [y - \hat{P}(y = 1 | x, \hat{w})]
\]

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 + \ldots\]

- Regularized least-squares (a.k.a. ridge regression), for \(\lambda > 0:\)

\[
w^* = \arg \min_w \sum_j \left[ \ell(x_j) - \sum_i w_i h_i(x_j) \right]^2 + \lambda \sum_i w_i^2
\]

Linear Separability

\[
\sum_i w_i x_i > 0
\]

\[
\sum_i w_i x_i < 0
\]
Large parameters → Overfitting

- 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., $L_2$: 
  $$\ell(w) = \ln \prod_{j=1}^{N} P(y_j|x_j, w) - \frac{\lambda}{2} ||w||_2^2$$
- Practical note about $w_0$:
  $$\frac{\partial \ell}{\partial w_i} \bigg|_{w=0}$$
- Gradient of regularized likelihood:

Standard v. Regularized Updates

- Maximum conditional likelihood estimate
  $$w^* = \arg \max_w \ln \prod_{j=1}^{N} P(y_j|x_j, w)$$
  $$w_i^{(t+1)} = w_i^{(t)} + \eta \sum_{j=1}^{N} x_{ij} [y_j - P(Y_j = 1 | x_j, w)]$$

- Regularized maximum conditional likelihood estimate
  $$w^* = \arg \max_w \ln \prod_{j=1}^{N} P(y_j|x_j, w) - \frac{\lambda}{2} \sum_{i=1}^{k} w_i^2$$
  $$w_i^{(t+1)} = w_i^{(t)} + \eta \left[ -\lambda w_i^{(t)} + \sum_{j} x_{ij} [y_j - P(Y_j = 1 | x_j, w)] \right]$$

Please Stop!! Stopping criterion

- When do we stop doing gradient descent?
- Because $\ell(w)$ is strongly concave:
  i.e., because of some technical condition
  $$\ell(w^*) - \ell(w) \leq \frac{1}{2\lambda} ||\nabla \ell(w)||_2^2$$
- Thus, stop when: $\nabla \ell(w)$ is small
Convergence rates for gradient descent/ascent

- Number of Iterations to get to accuracy \( \ell(w^*) - \ell(w) \leq \epsilon \)
  - If func Lipschitz: \( O(1/\epsilon^2) \)
  - If gradient of func Lipschitz: \( O(1/\epsilon) \)
  - If func is strongly convex: \( O(\ln(1/\epsilon)) \)

Digression: Logistic regression for more than 2 classes

- Logistic regression in more general case (C classes), where \( Y \in \{0, \ldots, C-1\} \)

Digression: Logistic regression more generally

- Logistic regression in more general case, where \( Y \in \{0, \ldots, C-1\} \)
  
  \[
  P(Y = c | x, w) = \frac{\exp(w_{0c} + \sum_{i=1}^{k} w_{ci} x_i)}{1 + \sum_{c'=0}^{C-1} \exp(w_{0c'} + \sum_{i=1}^{k} w_{c'i} x_i)}
  \]
  
  for \( c > 0 \) (normalization, so no weights for this class)
  
  \[
  P(Y = 0 | x, w) = \frac{1}{1 + \sum_{c'=0}^{C-1} \exp(w_{0c'} + \sum_{i=1}^{k} w_{c'i} x_i)}
  \]

  Learning procedure is basically the same as what we derived!

Stochastic Gradient Descent

Machine Learning – CSE546
Sham Kakade
University of Washington
October 18, 2016
The Cost, The Cost!!! Think about the cost...

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

\[
\begin{align*}
\mathbf{w}_t^{(l+1)} & = \mathbf{w}_t^{(l)} + \eta \left( -\lambda \mathbf{w}_t^{(l)} + \sum_{j=1}^{N} x_j^l [y_j^l - P(y_j^l = 1 | \mathbf{x}_j^l, \mathbf{w}_t)] \right) \\
\end{align*}
\]

The cost of updating \( \mathbf{w} \) is \( O(N^2 \lambda) \) naively, \( O(NA^2) \) for all coordinates. Rese \( \hat{\Gamma} \) completion \( O(NA) \).

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset:
    - Sampled iid from some distribution \( p(\mathbf{x}) \) on features:
  - Loss function, e.g., hinge loss, logistic loss, etc.
  - We often minimize loss in training data:
    \[
    \ell_D(\mathbf{w}) = \frac{1}{N} \sum_{j=1}^{N} \ell(w, \mathbf{x}_j)
    \]

- However, we should really minimize expected loss on all data:
  \[
  \ell(\mathbf{w}) = E_\mathbf{x} [\ell(\mathbf{w}, \mathbf{x})] = \int p(\mathbf{x}) \ell(\mathbf{w}, \mathbf{x}) d\mathbf{x}
  \]

- So, we are approximating the integral by the average on the training data.

Gradient ascent in Terms of Expectations

- “True” objective function:
  \[
  \ell(\mathbf{w}) = E_\mathbf{x} [\ell(\mathbf{w}, \mathbf{x})] = \int p(\mathbf{x}) \ell(\mathbf{w}, \mathbf{x}) d\mathbf{x}
  \]

- Taking the gradient:

- “True” gradient ascent rule:

- How do we estimate expected gradient?

SGD: Stochastic Gradient Ascent (or Descent)

- “True” gradient:
  \[
  \nabla \ell(\mathbf{w}) = E_\mathbf{x} [\nabla \ell(\mathbf{w}, \mathbf{x})]
  \]

- 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:
  \[ E_x [\ell(w, x)] = E_x [\ln P(y|x, w) - \lambda ||w||_2^2] \]
- Batch gradient ascent updates:
  \[ w_{t+1} = w_t + \eta \left\{ -\lambda w_t + \frac{1}{N} \sum_{i=1}^{N} x_i y_i - P(Y=1|x^{(i)}, w^{(i)}) \right\} \]
- Stochastic gradient ascent updates:
  - Online setting:
    \[ w^{(i+1)}_i = w^{(i)}_i + \eta_i \left\{ -\lambda w^{(i)}_i + x^{(i)}_i y^{(i)} - P(Y=1|x^{(i)}, w^{(i)}) \right\} \]

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

Stochastic Gradient Ascent: general case

- Given a stochastic function of parameters:
  - Want to find maximum
- Start from \(w^{(0)}\)
- Repeat until convergence:
  - Get a sample data point \(x^{(i)}\)
  - Update parameters:
- Works on the online learning setting!
- Complexity of each gradient step is constant in number of examples!
- In general, step size changes with iterations