Announcements:

- HW1 due on Friday.
- Today:
  - Review: sub-gradients, lasso
  - Logistic Regression

Variable Selection by Regularization

- Ridge regression: Penalizes large weights
- What if we want to perform “feature selection”?
  - E.g., Which regions of the brain are important for word prediction?
  - Can’t simply choose features with largest coefficients in ridge solution
- Try new (convex) penalty: Penalize non-zero weights
  - Regularization penalty:
  \[ \ell_1 \text{ loss} \quad \| w \|_1 = \sum_i | w_i | \]
  - Leads to sparse solutions
  - Just like ridge regression, solution is indexed by a continuous parameter \( \lambda \)
  - Major impact in: statistics, machine learning & electrical engineering
LASSO Regression

- LASSO: least absolute shrinkage and selection operator
- New objective:
  \[
  \min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} \left( t(x_i) - \mathbf{w}^T \mathbf{h}(x_i) \right)^2 + \frac{1}{2} \sum_{i=1}^{k} |w_i|
  \]

Optimizing the LASSO Objective

- LASSO solution:
  \[
  \hat{\mathbf{w}}_{\text{LASSO}} = \arg \min_{\mathbf{w}} \sum_{i=1}^{N} \left( t(x_i) - (w_0 + \sum_{i=1}^{k} w_i h(x_i)) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|
  \]

(Related) Constrained Optimization

- LASSO solution:
  \[
  \hat{\mathbf{w}}_{\text{LASSO}} = \arg \min_{\mathbf{w}} \sum_{i=1}^{N} \left( t(x_i) - (w_0 + \sum_{i=1}^{k} w_i h(x_i)) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|
  \]

Coordinate Descent

- Given a function \( F(\mathbf{w}) \)
- Want to find minimum
- Often, hard to find minimum for all coordinates, but easy for one coordinate
- Coordinate descent:
  \[
  \text{initialize } \hat{\mathbf{w}} = 0
  \]
  \[
  \text{while not converged} \]
  \[
  1) \text{pick coordinate randomly}
  \]
  \[
  2) \hat{\mathbf{w}} \leftarrow \arg \min_{\mathbf{w}} F(\mathbf{w} | \hat{\mathbf{w}}_{\text{old}}, \cdots, \hat{\mathbf{w}}_{\text{old}})
  \]
- How do we pick next coordinate?
- Super useful approach for “many” problems
- Converges to optimum in some cases, such as LASSO
Optimizing LASSO Objective One Coordinate at a Time

Taking the derivative:
- Residual sum of squares (RSS):
  $\frac{\partial}{\partial w_i} \text{RSS}(w) = -2 \sum_{j=1}^{N} \frac{h_i(x_j)}{t(x_j) - (w_0 + \sum_{k=1}^{K} w_k h_k(x_j))}$
- Penalty term:
  $\frac{\partial}{\partial w_i} \lambda \sum_{k=1}^{K} |w_k|$

Subgradients of Convex Functions
- Gradients lower bound convex functions:
  $\nabla f(w) \preceq 0$
- Gradients are unique at $w$ iff function differentiable at $w$
- Subgradients: Generalize gradients to non-differentiable points:
  - Any plane that lower bounds function:
    $\nabla f(w) \preceq \zeta$

Taking the Subgradient:
- Gradient of RSS term:
  $a_t = \frac{\sum_{j=1}^{N} h_i(x_j)^2}{t(x_j) - (w_0 + \sum_{k=1}^{K} w_k h_k(x_j))}$
- If no penalty:
  $\nabla \text{RSS}(w) = -a_t w_t - c_t$
- Subgradient of full objective:
  $F(w) = a_t w_t - c_t - \lambda |w_t|$
  $\frac{\partial}{\partial w_t} F(w) = a_t - c_t - \lambda \text{sgn}(w_t)$

Setting Subgradient to 0:

$\begin{cases} a_t w_t - c_t - \lambda w_t < 0 & w_t < 0 \\ -c_t - \lambda w_t = 0 & w_t = 0 \\ a_t w_t - c_t + \lambda w_t > 0 & w_t > 0 \end{cases}$
Soft Thresholding

\[ \hat{w}_l = \begin{cases} 
\frac{c_l + \lambda}{a_l} & c_l < -\lambda \\
0 & c_l \in [-\lambda, \lambda] \\
\frac{c_l - \lambda}{a_l} & c_l > \lambda 
\end{cases} \]

Coordinate Descent for LASSO (aka Shooting Algorithm)

- Repeat until convergence
  - Pick a coordinate \( l \) at (random or sequentially)
  - Set:
    \[ \hat{w}_l = \begin{cases} 
\frac{(c_l + \lambda)}{a_l} & c_l < -\lambda \\
0 & c_l \in [-\lambda, \lambda] \\
\frac{(c_l - \lambda)}{a_l} & c_l > \lambda 
\end{cases} \]
  - Where:
    \[ a_l = -\sum_{ji \neq l} (h_i(x_j))^2 \]
    \[ c_l = -\sum_{ji} (a_i) \left( \frac{1}{a_l} - \frac{1}{\sum_j a_j (x_j)} \right) \]

  - For convergence rates, see Shalev-Shwartz and Tewari 2009
  - Other common technique = LARS
  - Least angle regression and shrinkage, Efron et al. 2004

Recall: Ridge Coefficient Path

Now: LASSO Coefficient Path

Typical approach: select \( \lambda \) using cross validation
What you need to know

- Variable Selection: find a sparse solution to learning problem
- L₁ regularization is one way to do variable selection
  - Applies beyond regression
  - Hundreds of other approaches out there
- LASSO objective non-differentiable, but convex ➔ Use subgradient
- No closed-form solution for minimization ➔ Use coordinate descent
- Shooting algorithm is simple approach for solving LASSO

Sample size issues?

\[ L.S. \quad \text{How many samples do I need to get a "good" solution?} \]

\[ E_T [\mathbb{L}(\mathbf{w})] - \mathbb{L}(\mathbf{w}) \leq \frac{1}{b} \sigma^2 \]

Feature selection (use \( \mathbf{w} \) test. out of \( \mathbf{w} \))

\[ \mathbf{w} = \frac{\mathbf{X}^T \mathbf{y}}{\sigma^2} \]
Weather prediction revisited

Classification

- Learn: $h: X \mapsto Y$
  - $X$ – features
  - $Y$ – target classes

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

- Suppose you know $P(Y|X)$ exactly, how should you classify?
  - Bayes optimal classifier:
    \[ f(x) = \arg\max_y P(Y=y | X) \]
  - How do we estimate $P(Y|X)$?

Reading Your Brain, Simple Example

Pairwise classification accuracy: 85%

Person

Animal

Link Functions

- Estimating $P(Y|X)$: Why not use standard linear regression?
  \[ y \approx w_0 + \sum_i w_i x_i \]

- 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 link function
- Sigmoid applied to a linear function of the input features:

\[
P(Y = 0|X, W) = \frac{1}{1 + \exp(-z)}
\]

\[
f(Y = 1|X, w) = \frac{1}{1 - \frac{1}{1 + \exp(-z)}}
\]

\[
\frac{e^{z}}{1 + e^{z}} = w_{0} + \sum_{i} w_{i} x_{i}
\]

Features can be discrete or continuous!

Very convenient!

\[
P(Y = 0 | X = < X_{1}, ... X_{n} >) = \frac{1}{1 + \exp(w_{0} + \sum_{i} w_{i} X_{i})}
\]

implies

\[
P(Y = 1 | X = < X_{1}, ... X_{n} >) = \frac{\exp(w_{0} + \sum_{i} w_{i} X_{i})}{1 + \exp(w_{0} + \sum_{i} w_{i} X_{i})}
\]

implies

\[
P(Y = 1 | X) = \frac{\exp(w_{0} + \sum_{i} w_{i} X_{i})}{1 + \exp(w_{0} + \sum_{i} w_{i} X_{i})}
\]

linear classification rule!

Understanding the sigmoid

\[
g(w_{0} + \sum_{i} w_{i} x_{i}) = \frac{1}{1 + \exp(w_{0} + \sum_{i} w_{i} x_{i})}
\]

\[
f(Y = 0 | X, w)
\]

\[
f(Y = 1 | X, w)
\]

\[
\frac{e^{z}}{1 + e^{z}} = w_{0} + \sum_{i} w_{i} x_{i}
\]

\[
\frac{1}{1 + e^{z}} = w_{0} + \sum_{i} w_{i} x_{i}
\]
Optimizing concave function – Gradient ascent

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

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

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

\[ w_{t+1} = w_t + \eta \frac{\partial l(w)}{\partial w} \]

- 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: \( (x_i, y_i) \) \( i = 1 \) to \( n \)
  - Discriminative (logistic regression) loss function:
    - Conditional Data Likelihood
    \[ \prod_{i=1}^{N} P(y_i | x_i, w) = \prod_{i=1}^{N} \left( \frac{e^{w_0 + \sum_{j} w_j x_{i,j}}}{1 + e^{w_0 + \sum_{j} w_j x_{i,j}}} \right) \]

Expressing Conditional Log Likelihood

\[ l(w) = \sum_{j} y_j \ln P(Y = 1 | x_j, w) + (1 - y_j) \ln P(Y = 0 | x_j, w) \]

Maximizing Conditional Log Likelihood

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

\[ = \sum_{j} y_j (w_0 + \sum_{i} w_i x_{i,j}) - \ln(1 + e^{w_0 + \sum_{i} w_i x_{i,j}}) \]

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

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

Good news: concave functions easy to optimize
Maximize Conditional Log Likelihood:

Gradient ascent

\[ l(w) = \sum_j y_j^j (w_0 + \sum_i^n w_i x_i^j) - \ln(1 + \exp(w_0 + \sum_i^n w_i x_i^j)) \]

Gradient Ascent for LR

Gradient ascent algorithm: iterate until change < \varepsilon

\[
w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_j [y_j^j - P(Y^j = 1 | x^j, \tilde{\theta})]
\]

For i=1,...,k,

\[
w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y_j^j - P(Y^j = 1 | x^j, \tilde{\theta})]
\]

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:
  
  \[
w^* = \arg\min_w \sum_j \left( y_j - \sum_i w_i x_i^j \right)^2 + \lambda \sum_i w_i^2 \]

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

Standard v. Regularized Updates

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

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

Please Stop!! Stopping criterion

$$\ell(w) = \ln \prod_{j=1}^N P(y_j | x_j, w) - \lambda |w|^2$$
- When do we stop doing gradient descent?
- Because $\ell(w)$ is strongly concave:
  - i.e., because of some technical condition
    $$\ell(w^*) - \ell(w) \leq \frac{1}{2\lambda} \|\nabla \ell(w)\|^2$$
- Thus, stop when:
Digression: Logistic regression for more than 2 classes

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

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

for $c > 0$

$$P(Y = c | x, w) = \frac{\exp(w_c x_0 + \sum_{i=1}^{C-1} w_i x_i)}{1 + \sum_{c' = 0}^{C-1} \exp(w_{c'} x_0 + \sum_{i=1}^{C-1} w_{c'} x_i)}$$

for $c = 0$ (normalization, so no weights for this class)

$$P(Y = 0 | x, w) = \frac{1}{1 + \sum_{c' = 0}^{C-1} \exp(w_{c'} x_0 + \sum_{i=1}^{C-1} w_{c'} x_i)}$$

Learning 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???

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left( -\lambda w_i^{(t)} + \sum_{j} x_i^j [y_j - P(Y = 1 | x_j^i, w)] \right)$$
Learning Problems as Expectations

Minimizing loss in training data:
- Given dataset: Sampled iid from some distribution $p(x)$ on features:
- Loss function, e.g., hinge loss, logistic loss, ...
- We often minimize loss in training data:
  $$\ell_D(w) = \frac{1}{N} \sum_{j=1}^N \ell(w, x_j)$$
- However, we should really minimize expected loss on all data:
  $$\ell(w) = E_x [\ell(w, x)] = \int p(x) \ell(w, x) dx$$
- So, we are approximating the integral by the average on the training data

Gradient ascent in Terms of Expectations

“True” objective function:
  $$\ell(w) = E_x [\ell(w, x)] = \int p(x) \ell(w, x) dx$$
- Taking the gradient:
- “True” gradient ascent rule:
- How do we estimate expected gradient?

SGD: Stochastic Gradient Ascent (or Descent)

“True” gradient:
  $$\nabla \ell(w) = E_x [\nabla \ell(w, x)]$$
- 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:
  $$w^{(t+1)} = w^{(t)} + \eta \left\{ -\lambda w^{(t)} + \frac{1}{N} \sum_{j=1}^N x_j(y_j - P(Y=1|x_j, w^{(t)})) \right\}$$
- Stochastic gradient ascent updates:
  - Online setting:
    $$w_i^{(t+1)} = w_i^{(t)} + \eta_i \left\{ -\lambda w_i^{(t)} + x_i(y_i - P(Y=1|x_i, w_i^{(t)})) \right\}$$
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 \( x \)
  - 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

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

Stopping criterion

\[
\ell(w) = \ln \prod_j P(y_j | x_j, w) - \lambda ||w||^2
\]

- Regularized logistic regression is strongly concave
  - Negative second derivative bounded away from zero:

Strong concavity (convexity) is super helpful!
- For example, for strongly concave \( \ell(w) \):
  \[
  \ell(w^*) - \ell(w) \leq \frac{1}{2\lambda} ||\nabla \ell(w)||^2
  \]

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: \( O(\ln(1/\epsilon)) \)