Sparsity

- Vector $w$ is sparse, if many entries are zero:

- Very useful for many tasks, e.g.,
  - **Efficiency**: If $\text{size}(w) = 100B$, each prediction is expensive:
    - If part of an online system, too slow
    - If $w$ is sparse, prediction computation only depends on number of non-zeros
  - **Interpretability**: What are the relevant dimension to make a prediction?
    - E.g., what are the parts of the brain associated with particular words?

- But computationally intractable to perform “all subsets” regression

---

Figure from Tom Mitchell

©2005-2013 Carlos Guestrin
Simple greedy model selection algorithm

- Pick a dictionary of features
  - e.g., polynomials for linear regression
- Greedy heuristic:
  - Start from empty (or simple) set of features $F_0 = \emptyset$
  - Run learning algorithm for current set of features $F_t$
    - Obtain $h_t$
  - Select next best feature $X^*_i$
    - e.g., $X^*_i$ that results in lowest training error learner when learning with $F_t + \{X^*_j\}$
  - $F_{t+1} \Leftarrow F_t + \{X^*_i\}$
  - Recurse

Greedy model selection

- Applicable in many settings:
  - Linear regression: Selecting basis functions
  - Naïve Bayes: Selecting (independent) features $P(X_i|Y)$
  - Logistic regression: Selecting features (basis functions)
  - Decision trees: Selecting leaves to expand
- Only a heuristic!
  - But, sometimes you can prove something cool about it
    - e.g., [Krause & Guestrin ’05]: Near-optimal in some settings that include Naïve Bayes
- There are many more elaborate methods out there
When do we stop???

- **Greedy heuristic:**
  - ... 
  - Select **next best feature** $X^*_i$ 
    - e.g., $X^*_i$ that results in lowest training error learner when learning with $F_t + \{X^*_i\}$ 
  - $F_{t+1} = F_t + \{X^*_i\}$ 
  - **Recurse** 
    - When do you stop???
      - When training error is low enough? 
      - When test set error is low enough? 

---

Regularization in Linear Regression

- Overfitting usually leads to very large parameter choices, e.g.:
  - $-2.2 + 3.1 X - 0.30 X^2$ 
  - $-1.1 + 4,700,910.7 X - 8,585,638.4 X^2 + ...$

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

$L_2$ regularization tends to make smoother functions
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 penalty: Penalize non-zero weights  
  - Regularization penalty:
    - Leads to sparse solutions
    - Just like ridge regression, solution is indexed by a continuous param $\lambda$
    - This simple approach has changed statistics, machine learning & electrical engineering

LASSO Regression

- **LASSO**: least absolute shrinkage and selection operator

- New objective:
Geometric Intuition for Sparsity

Ridge Regression

Lasso

From Rob Tibshirani slides

Optimizing the LASSO Objective

\[ \hat{w}_{\text{LASSO}} = \arg \min_w \sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} |w_i| \]
Coordinate Descent

- Given a function $F$
  - Want to find minimum

- Often, hard to find minimum for all coordinates, but easy for one coordinate

- Coordinate descent:

- 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

$$\sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

- Taking the derivative:
  - Residual sum of squares (RSS):

$$\frac{\partial}{\partial w_\ell} RSS(w) = -2 \sum_{j=1}^{N} h_\ell(x_j) \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)$$

- Penalty term:
Subgradients of Convex Functions

- Gradients lower bound convex functions:

- Gradients are unique at \( \mathbf{w} \) iff function differentiable at \( \mathbf{w} \)

- Subgradients: Generalize gradients to non-differentiable points:
  - Any plane that lower bounds function:

Taking the Subgradient

- Gradient of RSS term:
  
  \[
  \frac{\partial}{\partial \mathbf{w}_\ell} \text{RSS}(\mathbf{w}) = a_\ell \mathbf{w}_\ell - c_\ell
  \]

  - If no penalty:
    - Subgradient of full objective:

\[
\begin{align*}
  a_\ell &= 2 \sum_{j=1}^{N} (h_\ell(x_j))^2 \\
  c_\ell &= 2 \sum_{j=1}^{N} h_\ell(x_j) \left( t(x_j) - (w_0 + \sum_{i \neq \ell} w_i h_i(x_j)) \right)
\end{align*}
\]
Setting Subgradient to 0

\[ \partial_{w_\ell} F(w) = \begin{cases} 
  a_\ell w_\ell - c_\ell - \lambda & w_\ell < 0 \\
  [-c_\ell - \lambda, -c_\ell + \lambda] & w_\ell = 0 \\
  a_\ell w_\ell - c_\ell + \lambda & w_\ell > 0 
\end{cases} \]

Soft Thresholding

\[ \hat{w}_\ell = \begin{cases} 
  (c_\ell + \lambda)/a_\ell & c_\ell < -\lambda \\
  0 & c_\ell \in [-\lambda, \lambda] \\
  (c_\ell - \lambda)/a_\ell & c_\ell > \lambda 
\end{cases} \]

From Kevin Murphy textbook
Coordinate Descent for LASSO (aka Shooting Algorithm)

- Repeat until convergence
  - Pick a coordinate $l$ at (random or sequentially)
    - Set: $\hat{w}_l = \begin{cases} 
      (c_l + \lambda)/a_l & c_l < -\lambda \\
      0 & c_l \in [-\lambda, \lambda] \\
      (c_l - \lambda)/a_l & c_l > \lambda 
    \end{cases}$
    - Where:
      \[
      a_l = 2 \sum_{j=1}^{N} (h_l(x_j))^2 \\
      c_l = 2 \sum_{j=1}^{N} h_l(x_j) \left( t(x_j) - (w_0 + \sum_{i \neq l} w_i h_i(x_j)) \right)
      \]
  - 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

- Typical approach: select $\lambda$ using cross validation
Now: LASSO Coefficient Path

From Kevin Murphy textbook

LASSO Example

<table>
<thead>
<tr>
<th>Term</th>
<th>Least Squares</th>
<th>Ridge</th>
<th>Lasso</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>2.465</td>
<td>2.452</td>
<td>2.468</td>
</tr>
<tr>
<td>lcavol</td>
<td>0.680</td>
<td>0.420</td>
<td>0.533</td>
</tr>
<tr>
<td>lweight</td>
<td>0.263</td>
<td>0.238</td>
<td>0.169</td>
</tr>
<tr>
<td>age</td>
<td>−0.141</td>
<td>−0.046</td>
<td></td>
</tr>
<tr>
<td>lbph</td>
<td>0.210</td>
<td>0.162</td>
<td>0.002</td>
</tr>
<tr>
<td>svi</td>
<td>0.305</td>
<td>0.227</td>
<td>0.094</td>
</tr>
<tr>
<td>lcp</td>
<td>−0.288</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>gleason</td>
<td>−0.021</td>
<td>0.040</td>
<td></td>
</tr>
<tr>
<td>pgg45</td>
<td>0.267</td>
<td>0.133</td>
<td></td>
</tr>
</tbody>
</table>

From Rob Tibshirani slides
Debiasing

What you need to know

- **Variable Selection**: find a sparse solution to learning problem
- $L_1$ regularization is one way to do variable selection
  - Applies beyond regressions
  - Hundreds of other approaches out there
- LASSO objective non-differentiable, but convex $\Rightarrow$ Use subgradient
- No closed-form solution for minimization $\Rightarrow$ Use coordinate descent
- Shooting algorithm is very simple approach for solving LASSO
THUS FAR, REGRESSION: PREDICT A CONTINUOUS VALUE GIVEN SOME INPUTS
Weather prediction revisited

Temperature

Pairwise classification accuracy: 85%

Person

Animal

[Mitchell et al.]
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:

How do we estimate $P(Y|X)$?

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

- Learn $P(Y|X)$ directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:
    
    $$P(Y = 0|X, W) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

Features can be discrete or continuous!

Understanding the sigmoid

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

$w_0=-2, w_1=-1$  $w_0=0, w_1=-1$  $w_0=0, w_1=-0.5$
Logistic Regression – a Linear classifier

\[
g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}
\]

Very convenient!

\[
P(Y = 0 \mid X =< X_1, \ldots X_n>) = \frac{1}{1 + e^{w_0 + \sum_i w_i X_i}}
\]

implies

\[
P(Y = 1 \mid X =< X_1, \ldots X_n>) = \frac{e^{w_0 + \sum_i w_i X_i}}{1 + e^{w_0 + \sum_i w_i X_i}}
\]

implies

\[
\frac{P(Y = 1 \mid X)}{P(Y = 0 \mid X)} = e^{w_0 + \sum_i w_i X_i}
\]

implies

\[
\ln \frac{P(Y = 1 \mid X)}{P(Y = 0 \mid X)} = w_0 + \sum_i w_i X_i
\]