Last time:

How to get primal variables from dual?

- argmax to conjugate of g

Example:

$$p(x) = f(Ax) + \frac{\lambda}{2}\|x\|^2$$

$$-D(y) = f^*(x) + g^*(A^T y)$$

$$= f^*(x) + \frac{1}{2\lambda} y^T A A^T y \Rightarrow \sup_x \left\{ y^T A - \frac{\lambda}{2}\|x\|^2 \right\}$$

$$\Downarrow$$

$$x = \frac{1}{\lambda} A^T y$$

If have y^*; $x^* = A^T y^*$

assuming strong duality holds

Kernel trick: $\hat{b} = \hat{A} x$

$$= \frac{1}{\lambda} \hat{A} A^T y$$

$$= \frac{1}{\lambda} K(\hat{A}, A) y$$

Admin:

- A2 / A3: pick up ready
- A4 marked version due Monday
- A5 due Wednesday
- Project Proposal due Nov 3rd.

Ensemble Methods

- "models that use other models."
- can give better performance than individual models
- e.g. decision trees, bootstrap, bagging, boosting, random forests

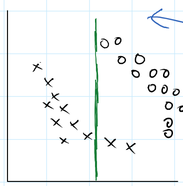
  stacking, jackknifes, etc ...

Decision Stumps

- find variable 'j' and threshold 'T', such that classifier $[(\bar{x}_i)_j \geq T]$ maximizes some score (e.g. Training accuracy) or other scores

$O(np)$

$(\bar{x})_2$



axis aligned

$(\bar{x})_1$

individual variables

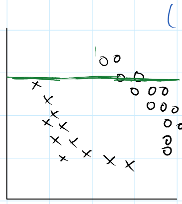
Randomized Decision Stump

- Choose a random subset $\{1, \cdots, d\}$

  only choose "j" from subset

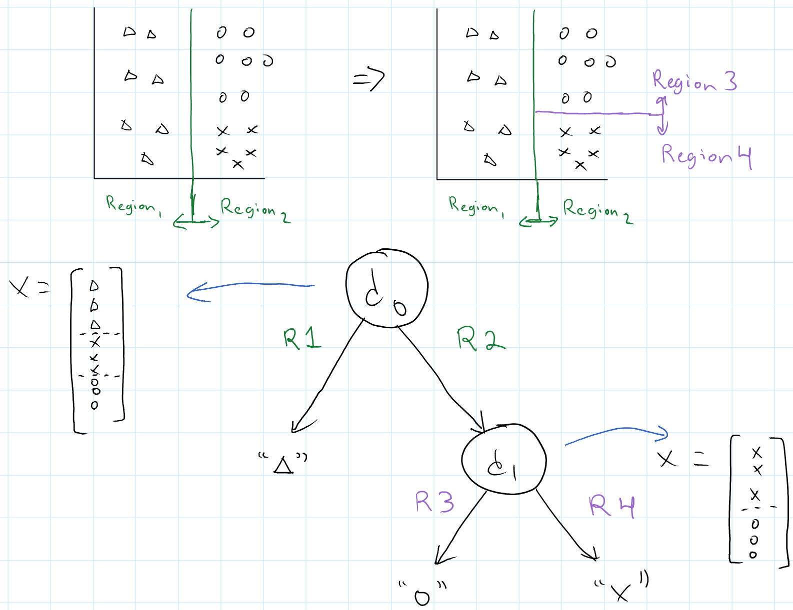
  (not recommended for accuracy, but increases speed)

$(\bar{x})_2$



$(\bar{x})_1$

Decision Trees



Issues:

1. Very interpretable (for shallow depth)

2. Finding optimal tree is NP-Hard (mostly done greedily)
   - start w/ full data set
   - learn decision stump
   - partition into smaller data sets

3. Choosing score
   - last layer: counting
   - intermediate layers "info gain" "mutual information"

4. Choosing classifier
   - [randomized stump], linear classifier, non-linear classifier

   cost increases $\longrightarrow$

5. Pruning: prune nodes that don't decrease validation error

6. CART (Breiman) C 4.5 (Quinlan)

7. tend to be worse than linear models in high dimensions

Infogain

$$I(X,Y) = \underbrace{H(X)}_{\text{entropy before split}} - \underbrace{H(X|Y)}_{\text{entropy after split}}$$

$$H(x) = -\sum_{x \in X} p(x) \log_2(p(x))$$

"entropy" - measure of randomness

$$H(X|Y) = \sum_{y \in Y} p(y) H(X \mid Y=y)$$

zero if split perfectly classifies (otherwise higher)

probability of assigning region 'R'

- Comparison to log sum exp:

$$f(x) = \log\left(\sum_i \exp(x_i)\right)$$

$$f^*(y) = \sum_i y_i \log_e(y_i) \quad \text{s.t.} \quad y_i \geq 0, \quad \sum_i y_i = 1$$

similar to entropy, but with conditions

Stacking (Model Averaging)

- train m different classifiers

naive bayes, Knn, logistic, neural nets ...

stacking: new classifier that combines the outputs

$$\hat{y}_i = f\left(\sum_{j=1}^{m} w_j h_j(\bar{x}_i)\right)$$

Special Cases

1. Linear $h_j(\bar{x}_i) = (\bar{x})_j$
2. Neural Nets $h_j(\bar{x}_i) = \sigma(\bar{w}_{0j}^T x_i)$
3. Generalized Additive $h_j(\bar{x}_i) = g_j((\bar{x}_i)_j)$

* winner of Netflix prize ($1 million)

Bootstrap / Bagging

- Input: a high variance classifier (overfitting)
- Output: a lower variance classifier

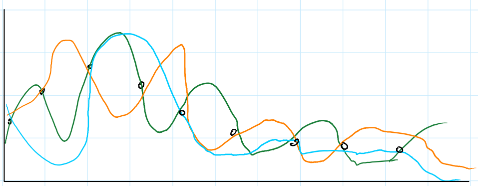
(same dimension)

$$\begin{array}{ccc} D & \Rightarrow & D_j \\ (x_1, y_1) & & (\ \ ) \\ (x_2, y_2) & & (\ \ ) \\ \vdots & & (\ \ ) \\ (x_N, y_N) & & (\ \ ) \end{array}$$

replacement

- basically a "reweighting"
- for large N, selects $(1 - 1/e) \approx 63\%$ of data
- do this 'm' times independently

Bagging: train high variance classifier on each sample, average results



(averaging will provide a good fit)

Random Forests

- Bagging
- Decision Trees
- Info gain
- Random Decision stumps

→ why? - speed
- reduce correlation between classifier

Kinect:

RF to predict body part or background at each pixel.

Boosting

Input: a 'weak' learner,
binary classifier w/ accuracy $> 50\%$ ex: decision stump, decision trees.

Output: a 'strong' learner

AdaBoost (Freund & Schapire)

Set datapoint weights $z_i = \frac{1}{N}$ (weight: examples worth more)

number of weak classifiers to learn

for $j = 1 : m$ when to stop? cross validation

- train 'weak' classifier with weight $\bar{z}$
- choose optimal w_j (exponential loss)
- re-weight $z_i = z_i \exp(w_j I[y_i \neq h_j(\bar{x}_j)])$ increase weight of points you got "wrong"

$\hat{y} = f\left(\sum_j w_j h_j(\bar{x}_i)\right)$

Issues:

- can overfit
- doesn't work if classifiers lack diversity