Last time:

How to get primal variables from dual?
- \( \text{argmax} \) to conjugate of \( g \)

Example:
\[
\begin{align*}
\ell(x) &= 5(Ax) + \frac{1}{2}||x||^2 \\
-\ell(y) &= \frac{1}{2}x^T(A^TA)y + \frac{1}{2}y^TAy \\
&= \frac{1}{2}x^T(A^TA)y + \frac{1}{2}y^TAy \\
\implies & \sup_x \left\{ y^TAx - \frac{1}{2}||x||^2 \right\} \\
&= y^TAy \\
&= \frac{1}{\lambda} y \\
\text{if have } y^*, x^* = A^Ty^* \\
\text{assuming strong duality holds}
\end{align*}
\]

Kernel trick:
\[
\begin{align*}
\ell(x) &= \frac{1}{2}x^T(A^TA)x \\
&= \frac{1}{2}x^T(A^TA)y \\
&= \frac{1}{2}x^T(A^TA)y \\
&= x_k(A^TA)y
\end{align*}
\]

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

Decision Stumps
- find variable \( i \) and threshold \( t \)
such that classifier \( [ (x_i) : i > t ] \)
maximizes some score
(e.g. Training accuracy) or other scores

\[
\begin{array}{c}
\text{(x)} \\
\text{axis aligned} \\
\text{(x)}
\end{array}
\]

Randomized Decision Stump
- choose a random subset \( \{ 1, \ldots, d \} \)
only choose "i" from subset
(not recommended for accuracy, but increases speed)

\[
\begin{array}{c}
\text{(x)} \\
\text{individual variables} \\
\text{(x)}
\end{array}
\]
Decision Trees

Issues:
1. Very interpretable (for shallow depth)
2. Finding optimal tree is NP-Hard
   - start w/ full data set
   - learn decision stump
   - partition into smaller data sets
   (mostly done greedily)
3. Choosing Score
   - last layer: counting
   - intermediate layers "info gain"
   "mutual information"
4. Choosing classifier
   - randomized stump
   linear classifier non-linear classifier
   cost increases
5. Pruning: prune nodes that don't decrease validation error
6. CART C4.5
   (Breiman) (Quinlan)
7. Tend to be worse than linear models in high dimensions
Information gain:

\[ I(X, Y) = H(X) - H(X | Y) \]

Entropy before split:

\[ H(X) = -\sum_{x \in X} p(x) \log_a(p(x)) \]

"Entropy" - measure of randomness

Entropy after split:

\[ H(X | Y) = \sum_{y \in Y} p(y) H(X | Y=y) \]

\[ \text{zero if split perfectly; higher otherwise} \]

Comparison to log sum exp:

\[ f(x) = \log \left( \frac{1}{c} \exp(x_i) \right) \]

\[ f^*(y) = \sum_{i} y_i \log_a(y_i) \quad \text{s.t.} \quad y_i > 0 \]

\[ \sum_{i} y_i = 1 \]

Similar to entropy, but with conditions

**Stacking** (Model Averaging)

- Train \( n \) different classifiers

- Naive Bayes, kNN, logistic, neural nets, ...

- Stacking: new classifier that combines the outputs

\[ \hat{y}_i = \hat{f} \left( \sum_{j=1}^{n} \omega_j \cdot h_j(\overline{x}_i) \right) \]

**Special Cases**

1. Linear: \( h_j(\overline{x}_i) = \overline{x}_i \)

2. Neural Nets: \( h_j(\overline{x}_i) = g_k \left( \overline{w}_j^T \cdot \overline{x}_i \right) \)

3. Generalized Additive: \( h_j(\overline{x}_i) = g_j \left( \overline{r}_j \cdot \overline{x}_i \right) \)

*Winner of Netflix prize (41 million)
Bootstrap / Bagging

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

\[
D_1 \Rightarrow D_3 \Rightarrow \ldots \Rightarrow D_N
\]

- basically a "reweightng"
- for large \( N \), selects \( (1 - \frac{1}{e}) \approx 63\% \) of data
- do this \( m \) times independently

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

![Graph showing data smoothing]

(Averaging will provide a good fit)

**Random Forests**

- Bagging
- Decision Trees
- Info gain
- Random Decision Stumps

\[ \text{why? - speed} \]
\[ \text{- reduce correlation between classifier} \]

**Kinect**:

RF to predict body part or background at each pixel.
Boosting

Input: a ‘weak’ learner, binary classifier w/ accuracy > 50% (e.g. decision stump, decision tree)

Output: a ‘strong’ learner

AdaBoost (Freund & Schapire)

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

for \( j = 1 \rightarrow m \)

- train ‘weak’ classifier with weight \( \frac{Z_i}{Z} \)
- choose optimal \( w_j \) (exponential loss)
- re-weight \( Z_i = Z_i \exp(w_j [y_i \neq h_j(x_i)]) \) (increase weight of points you got “wrong”)

\( \hat{y} = \text{sign}(\sum \frac{w_j h_j(x_i))}{N}) \)

Issues:
- can overfit
- doesn’t work if classifiers lack diversity