CSC321 Tutorial 6:
Part 1: recurrent neural network
Part 2: combining models (Bagging & AdaBoost)

Yue Li
Email: yueli@cs.toronto.edu

Wed 11-12 Feb 26
Fri 10-11 Feb 28

Part 1 Recurrent neural network; see handwritten notes: `http://www.cs.utoronto.ca/~yueli/CSC321_UTM_2014_files/tut6_rnn.pdf`

Materials are based on course readings: Learning internal representations by error propagation, pp 354-362: `http://www.cs.toronto.edu/~hinton/absps/pdp8.pdf`

Part 2 combining models: Bagging (Breiman, 1996)

General idea:

1. Sample *with replacement* (aka bootstrap) $N'_1, \ldots, N'_m$ data points from the original N data points $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$; (input) $\mathbf{Y} = \{y_1, \ldots, y_N\}$ (response)
2. Train m models f_j ($j \in \{1, \ldots, m\}$) on the $N'_1, \ldots, N'_m$ data
3. Perform prediction on new test data $\mathbf{x}_i$ to predict y_i:

   For continuous y_j:

$$\hat{y}_j = \frac{1}{m} \sum_{j=1}^{m} f_j(\mathbf{x}_i, \boldsymbol{\theta}_j)$$

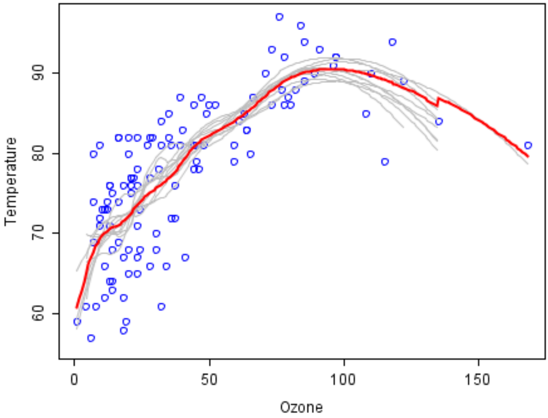
   For discrete y_j:

$$\hat{y}_j = \arg\max_k \sum_j I(f_j(\mathbf{x}_i, \boldsymbol{\theta}_j), k)$$

   where $I(f_m(\mathbf{x}_j, \boldsymbol{\theta}_j), k)$ returns 1 if $f_m(\mathbf{x}_j, \boldsymbol{\theta}_j) = k$; 0 otherwise

Why does Bagging work?

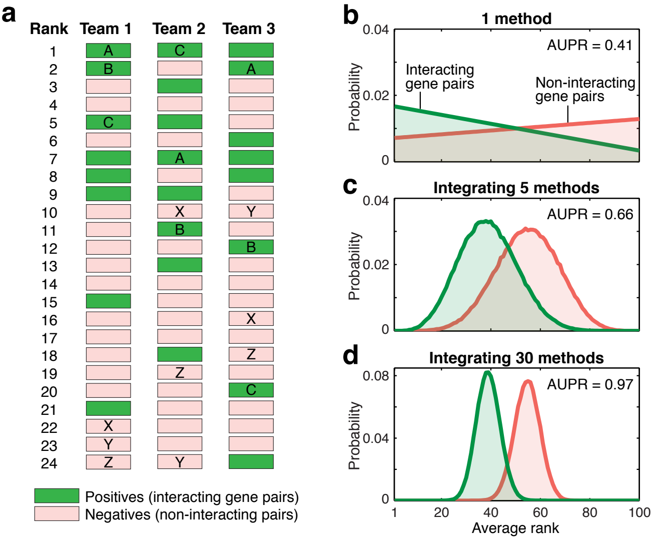
- Effective on "unstable" learning algorithms where small changes in the training set result in large changes in predictions (Breiman, 1996)



http://en.wikipedia.org/wiki/Bootstrap_aggregating

Wisdom of crowds for robust gene network inference (DREAM5)

(Marbach *et al.* (2012). Nature Methods, **9**(8), 796-804)



Average ranking: $r_{\mathrm{Borda}}(I) = \frac{1}{K}\sum_{j=1}^{K} r_j(I)$

AdaBoost general idea

- Given training data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with labels $\mathbf{Y} = \{y_1, \ldots, y_N\}$, where $y_i \in \{-1, +1\}$

  e.g., eye detection in an image of D pixels $\mathbf{x}_i$, where $y_i = +1$ is eye; $y_i = -1$ for non-eye;
- Task: Seek a strong classifier by combining K weak classifiers to predict y_i from the training data as accurate as possible
- Intuition: Mistakes made by the k^{th} weak classifier should be taken more seriously by the $(k+1)^{th}$ classifier
- NB: The weak classifiers must be reasonably better than random guess (i.e., more accurate than 50% chance of making a right/wrong decision by tossing a coin)

Algorithm 1 AdaBoost

for k = 1 to K classifiers **do**

Fit weak classifier k to minimize the objective function:

$$\epsilon_k = \frac{\sum_i w_i^{(k)} I[f_k(\mathbf{x}_i, \boldsymbol{\theta}_k) \neq y_i]}{\sum_i w_i^{(k)}} \tag{1}$$

$$\alpha_k = \ln(\frac{1-\epsilon_k}{\epsilon_k}) > 0 \tag{2}$$

for i = 1 to N training cases **do**

$$w_i^{(k+1)} = w_i^{(k)} e^{\alpha_k I[f_k(\mathbf{x}_i, \boldsymbol{\theta}_k) \neq y_i]} \tag{3}$$

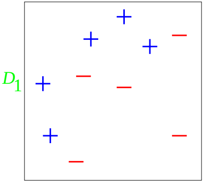
end for

end for

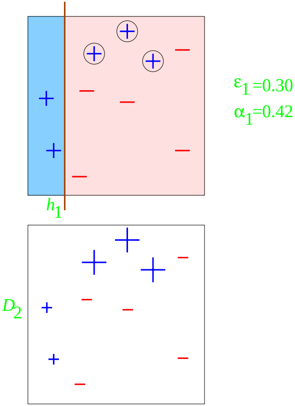
Final prediction:

$$\hat{y}_i = sign\left(\sum_k \alpha_k f_k(\mathbf{x}_i, \boldsymbol{\theta}_k)\right) \tag{4}$$

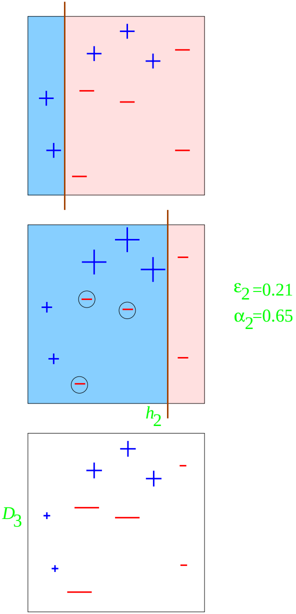
Toy Example



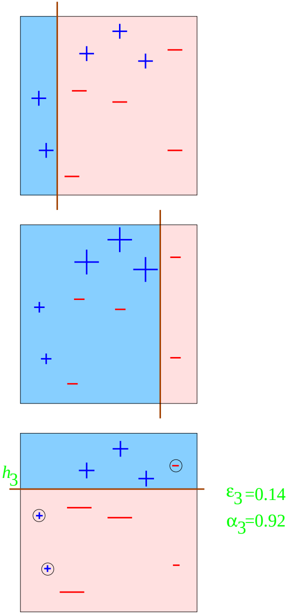
Round 1



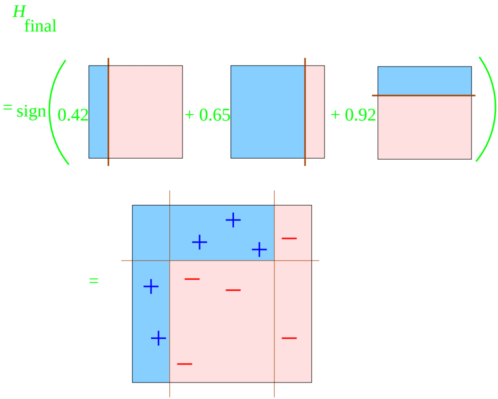
Round 2



Round 3



Final Hypothesis



* See demo at
`www.research.att.com/~yoav/adaboost`

TA office hours before midterm:
12-1:30 March 5 (Wednesday next week) at DV1160