Precept Outline

- Review of Lectures 21 and 22:
  - Randomness
  - Multiplicative Weights
  - Decision Stumps and Boosting

A. Review: Randomness and Multiplicative Weights

Your preceptor will briefly review key points of this week's lectures.

B. Weak Learners and Boosting

In this problem, we will work through a small example of the *weak learner* you will be required to implement in the final programming assignment.

A **decision stump** is a very simple kind of binary classifier for points in k-dimensional space. Its decision depends on three values:

- the **dimension predictor** d_p, an integer between 0 and $k-1$;
- the **value predictor** v_p, an integer; and
- the **sign predictor** $s_p \in \{0, 1\}$.

With these three values, the decision stump outputs a prediction for the **label** (i.e., either 0 or 1) of a sample point $\mathbf{x} = (x_0, x_1, \ldots, x_{k-1})$ as follows:

- if $s_p = 0$, output 0 if $x_{d_p} \leq v_p$ (and output 1 if $x_{d_p} > v_p$);
- if $s_p = 1$, output 1 if $x_{d_p} \leq v_p$ (and output 0 if $x_{d_p} > v_p$).

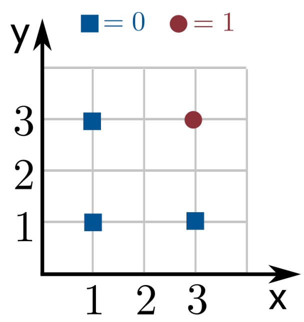
(In the following examples the dimension is $k = 2$, so we can plot the points.)

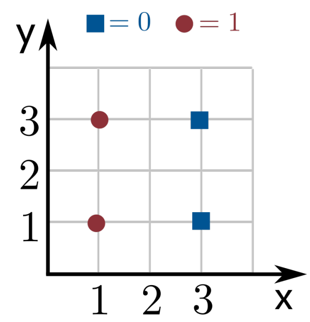
For example, if $d_p = 1$, $v_p = 0$ and $s_p = 1$, the predicted labels of $\mathbf{x} = (0, 0)$, $\mathbf{y} = (100, -2)$ and $\mathbf{z} = (-100, 1)$ are 1, 1 and 0, respectively.

In the dataset examples below, count the number of correctly classified points (i.e., points whose predicted label matches the actual label) for the two decision stumps with the following values:

1. $d_p = 1$, $v_p = 2$ and $s_p = 0$;
2. $d_p = 0$, $v_p = 1$ and $s_p = 1$.

Additionally, determine which one is the best weak learner, i.e. the one that classifies the most points correctly.





(Blue squares denote points labeled 0 and red circles denote points labeled 1. Dimension 0 corresponds to coordinates in the x-axis, while dimension 1 corresponds to the y-axis.)

Unfortunately, no decision stump can classify all points correctly in (the first dataset of) the previous problem. So we will try to get around this by combining multiple decision stumps.

Boosting is a technique that enables us to increase the accuracy of a weak learner (like a decision stump). To apply it, we first assign a weight to each one of the 4 points, which initially is $1/4$ (in general, if we had n points the initial weight would be $1/n$). Now we work in iterations, each of which creates a new decision stump based on the current weights and updates them at the end. After T iterations, we have T decision stumps. To classify a new point, we take the majority decision of each one of the T decision stumps (i.e., if more than half of decision stumps predict 0, then so does the boosted classifier; and likewise for 1).

Each boosting iteration does the following:

- creates a new decision stump for the dataset with the current weights;
- doubles the weights of misclassified points; and
- renormalizes the weights (i.e., divide each by the sum of all so that they sum to 1 again).

Each decision stump we create chooses d_p, v_p and s_p to maximize the *weight* (rather than number) of correctly classified points.

Run the boosting algorithm in the first dataset above for 3 iterations. Verify that the resulting decision stumps now correctly label all points (when taking the majority decision).

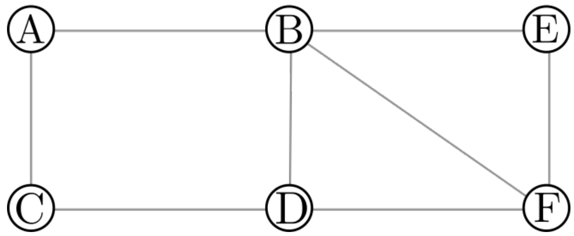
C. Global Mincut

Recall the global mincut problem: you are given a connected, unweighted, undirected graph G. A cut is a set of edges which, if removed, disconnects G. The goal is to find the cut that uses the fewest edges.

In lecture you learned one way of solving this problem: Karger's algorithm. It can be summarized in three steps:

- Assign a random weight (uniform between 0 and 1) to each edge.
- Run Kruskal's MST algorithm until 2 connected components left.
- output the cut defined by the 2 connected components.

Consider the following graph and set of random edge weights. Run Karger's algorithm with these edge weights and find the global cut it produces. Is it a mincut? If not, how many crossing edges does the mincut have?



edge	random weight
A—B	0.2
A—C	0.1
B—D	0.7
B—E	0.6
B—F	0.5
C—D	0.8
D—F	0.4
E—F	0.3