COMP 551 – Applied Machine Learning
Lecture 12: Ensemble learning

Associate Instructor: Herke van Hoof

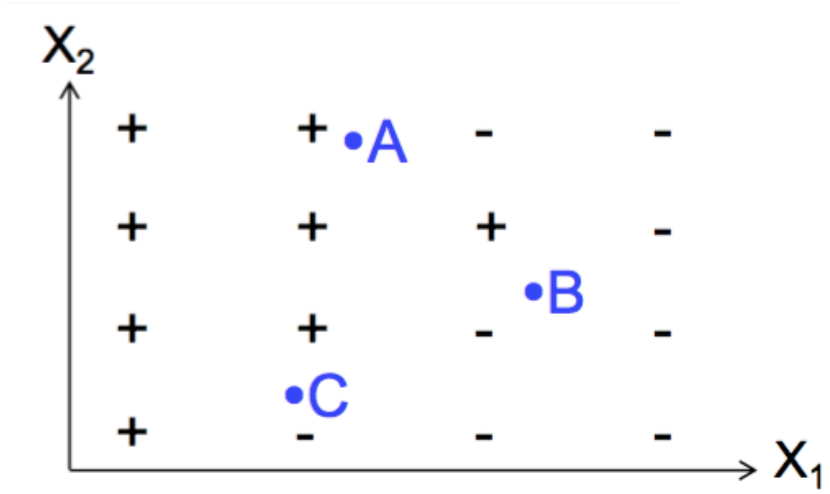
(*herke.vanhoof@mcgill.ca*)

Slides mostly by: Joelle Pineau (*jpineau@cs.mcgill.ca*)

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Today's quiz

1. Output of 1NN for A?
2. Output of 3NN for A?
3. Output of 3NN for B?

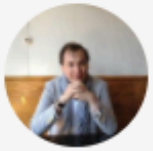


4. Explain in 1-2 sentences the difference between a "lazy" learner (such as nearest neighbour classifier) and an "eager" learner (such as logistic regression classifier).

Project #2

- A note on the contest rules:
  - You are allowed to use the built-in cross-validation methods from libraries like scikit-learn, for all parts.
  - You are allowed to use NLTK or another library for preprocessing your data for all parts
  - You can use an outside corpus to evaluate the features (e.g. TF-IDF).

Project #2



Edouard Grave

Language identification

October 2, 2017

Fast and accurate language identification using fastText

We are excited to announce that we are publishing a fast and accurate tool for text-based language identification. It can recognize more than 170 languages, takes less than 1MB of memory and can classify thousands of documents per second. It is based on fastText library and is released here as open source, free to use by everyone. We are releasing several versions of the model, each optimized for different memory usage, and compared them to the popular tool langid.py.

Project #2

- Some features:
  - Sub-word features (skiing: ski – kii – iin - ing) allows out-of-vocabulary and misspelling
  - Languages in hierarchical tree make use of inbalance in classes
  - K-means and feature selection to reduce model size

Next topic: Ensemble methods

- Recently seen supervised learning methods:
  - Logistic regression, Naïve Bayes, LDA/QDA
  - Decision trees, Instance-based learning

- Core idea of decision trees? Build complex classifiers from simpler ones.

  E.g. Linear separator -> Decision trees

- Ensemble methods use this idea with other 'simple' methods

- Several ways to do this.
  - Bagging
  - Random forests
  - Boosting

Lectures 4,5 – Linear Classification
Lecture 7 – Decision Trees

Ensemble learning in general

- Key idea: Run a base learning algorithm multiple times, then combine the predictions of the different learners to get a final prediction.
  - What's a base learning algorithm?
    - Naïve Bayes, LDA, Decision trees, SVMs, …

Ensemble learning in general

- Key idea: Run a base learning algorithm multiple times, then combine the predictions of the different learners to get a final prediction.
  - What's a base learning algorithm?
    - Naïve Bayes, LDA, Decision trees, SVMs, …

- First attempt: Construct several classifiers **independently**.
  - Bagging.
  - Randomizing the test selection in decision trees (Random forests).
  - Using a different subset of input features to train different trees.

Ensemble learning in general

- Key idea: Run a base learning algorithm multiple times, then combine the predictions of the different learners to get a final prediction.
  - What's a base learning algorithm?
    - Naïve Bayes, LDA, Decision trees, SVMs, …
- First attempt: Construct several classifiers **independently**.
  - Bagging.
  - Randomizing the test selection in decision trees (Random forests).
  - Using a different subset of input features to train different trees.
- More complex approach: **Coordinate** the construction of the hypotheses in the ensemble.

Ensemble methods in general

- Training models independently on same dataset tends to yield same result!
- For an ensemble to be useful, trained models need to be different
  1. Use slightly different (randomized) datasets
  2. Use slightly different (randomized) training procedure

Recall bootstrapping

- Given dataset D, construct a bootstrap replicate of D, called D_k, which has the same number of examples, by drawing samples from D with replacement.
- Use the learning algorithm to construct a hypothesis h_k by training on D_k.
- Compute the prediction of h_k on each of the remaining points, from the set $T_k = D - D_k$.
- Repeat this process K times, where K is typically a few hundred.

Estimating bias and variance

- For each point $\boldsymbol{x}$, we have a set of estimates $h_1(\boldsymbol{x}), \ldots, h_K(\boldsymbol{x})$, with $K \leq B$ (since $\boldsymbol{x}$ might not appear in some replicates).

- The average empirical prediction of $\boldsymbol{x}$ is: $\hat{h}(\boldsymbol{x}) = (1/K) \sum_{k=1:K} h_k(x)$.

- We estimate the bias as: $y - \hat{h}(\boldsymbol{x})$.

- We estimate the variance as: $(1/(K-1)) \sum_{k=1:K} (\hat{h}(\boldsymbol{x}) - h_k(\boldsymbol{x}))^2$.

Bagging: **B**ootstrap **agg**regation

- If we did all the work to get the hypotheses h_b, why not use all of them to make a prediction? (as opposed to just estimating bias/variance/error).

- All hypotheses get to have a vote.
  - For classification: pick the majority class.
  - For regression, average all the predictions.

- Which hypotheses classes would benefit most from this approach?

Bagging

- For each point $\boldsymbol{x}$, we have a set of estimates $h_1(\boldsymbol{x}), \ldots, h_K(\boldsymbol{x})$, with $K \leq B$ (since $\boldsymbol{x}$ might not appear in some replicates).
  - The average empirical prediction of $\boldsymbol{x}$ is: $\hat{h}(\boldsymbol{x}) = (1/K) \sum_{k=1:K} h_k(x)$.
  - We estimate the bias as: $y - \hat{h}(\boldsymbol{x})$.
  - We estimate the variance as: $(1/(K-1)) \sum_{k=1:K} (\hat{h}(\boldsymbol{x}) - h_k(\boldsymbol{x}))^2$.

Bagging

- In theory, bagging eliminates variance altogether.
- In practice, bagging tends to reduce variance and increase bias.
- Use this with “unstable” learners that have high variance, e.g. decision trees, neural networks, nearest-neighbour.

Random forests (Breiman, 2001)

- Basic algorithm:
  - Use K bootstrap replicates to train K different trees.
  - At each node, pick m variables at random (use $m<M$, the total number of features).
  - Determine the best test (using normalized information gain).
  - Recurse until the tree reaches maximum depth (no pruning).

Random forests (Breiman, 2001)

- Basic algorithm:
  - Use *K* bootstrap replicates to train *K* different trees.
  - At each node, pick *m* variables at random (use *m*<*M*, the total number of features).
  - Determine the best test (using normalized information gain).
  - Recurse until the tree reaches maximum depth (no pruning).

- Comments:
  - Each tree has high variance, but the ensemble uses averaging, which reduces variance.
  - Random forests are very competitive in both classification and regression, but still subject to overfitting.

Extremely randomized trees (Geurts et al., 2006)

- Basic algorithm:
  - Construct *K* decision trees.
  - Pick *m* attributes at random (without replacement) and pick a random test involving each attribute.
  - Evaluate all tests (using a normalized information gain metric) and pick the best one for the node.
  - Continue until a desired depth or a desired number of instances (n_{min}) at the leaf is reached.

Extremely randomized trees (Geurts et al., 2005)

- Basic algorithm:
  - Construct *K* decision trees.
  - Pick *m* attributes at random (without replacement) and pick a random test involving each attribute.
  - Evaluate all tests (using a normalized information gain metric) and pick the best one for the node.
  - Continue until a desired depth or a desired number of instances (n_{min}) at the leaf is reached.
- Comments:
  - Very reliable method for both classification and regression.
  - The smaller *m* is, the more randomized the trees are; small *m* is best, especially with large levels of noise. Small n_{min} means less bias and more variance, but variance is controlled by averaging over trees.
  - Compared to single trees, can pick smaller n_{min} (less bias)

Randomization

- For an ensemble to be useful, trained models need to be different
  1. Use slightly different (**randomized**) datasets
     - Bootstrap Aggregation (Bagging)
  2. Use slightly different (**randomized**) training procedure
     - Extremely randomized trees, Random Forests

Randomization in general

- Instead of searching very hard for the best hypothesis, generate lots of random ones, then average their results.
- Examples:
  - Random feature selection Random projections.
- Advantages?

- Disadvantages?

Randomization in general

- Instead of searching very hard for the best hypothesis, generate lots of random ones, then average their results.
- Examples:
  - Random feature selection Random projections.
- Advantages?
  - Very fast, easy, can handle lots of data.
  - Can circumvent difficulties in optimization.
  - Averaging reduces the variance introduced by randomization.
- Disadvantages?

Randomization in general

- Instead of searching very hard for the best hypothesis, generate lots of random ones, then average their results.
- Examples:
  - Random feature selection Random projections.
- Advantages?
  - Very fast, easy, can handle lots of data.
  - Can circumvent difficulties in optimization.
  - Averaging reduces the variance introduced by randomization.
- Disadvantages?
  - New prediction may be more expensive to evaluate (go over all trees).
  - Still typically subject to overfitting.
  - Low interpretability compared to standard decision trees.

Randomization

- For an ensemble to be useful, trained models need to be different
  1. Use slightly different (**randomized**) datasets
     - Bootstrap Aggregation (Bagging)
  2. Use slightly different (**randomized**) training procedure
     - Extremely randomized trees, Random Forests
  3. Alternative method to randomization?

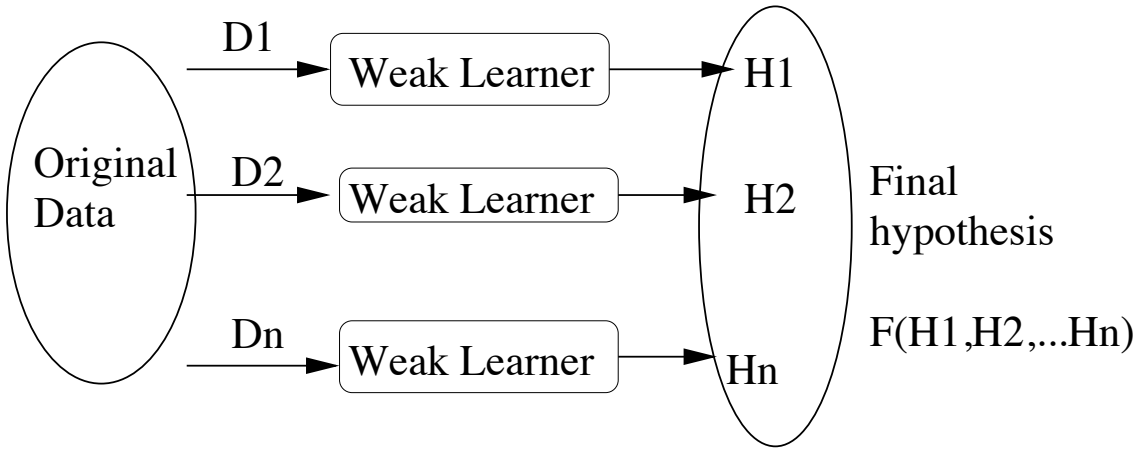
Additive models

- In an ensemble, the output on any instance is computed by averaging the outputs of several hypotheses.

- **Idea**: Don't construct the hypotheses independently. Instead, new hypotheses should focus on instances that are problematic for existing hypotheses.
  - If an example is difficult, more components should focus on it.

Boosting

- Boosting:
  - Use the training set to train a simple predictor.
  - Re-weight the training examples, putting more weight on examples that were not properly classified in the previous predictor.
  - Repeat *n* times.
  - Combine the simple hypotheses into a single, accurate predictor.



Notation

- Assume that examples are drawn independently from some probability distribution P on the set of possible data D.

- Let $J_P(h)$ be the expected error of hypothesis h when data is drawn from P:

$$J_P(h) = \sum_{<\mathbf{x},y>} J(h(\mathbf{x}),y) P(<\mathbf{x},y>)$$

where $J(h(\mathbf{x}),y)$ could be the squared error, or 0/1 loss.

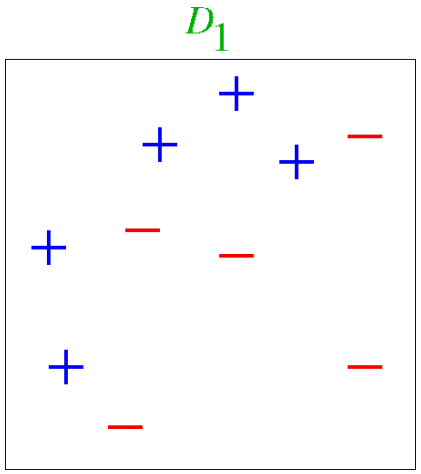
Weak learners

- Assume we have some "weak" binary classifiers:
  - A decision stump is a single node decision tree: $x_i > t$
  - A single feature Naïve Bayes classifier.
  - A 1-nearest neighbour classifier.

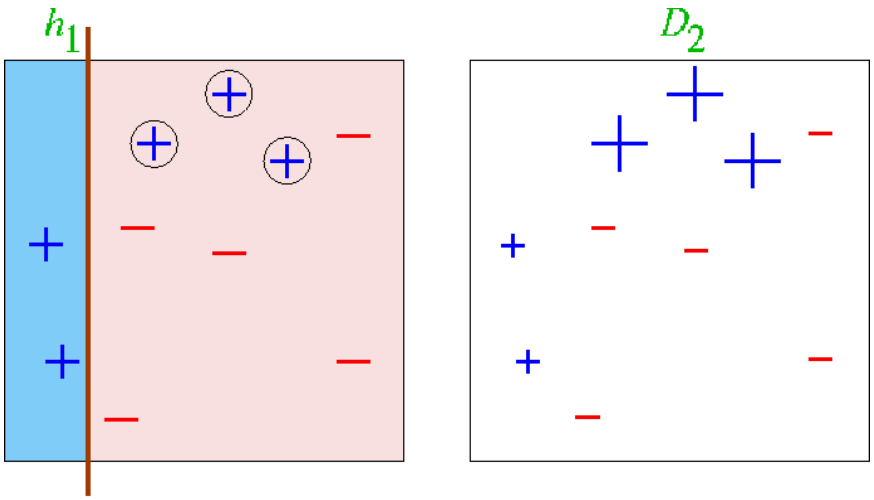
- "Weak" means $J_P(h) < 1/2 - \gamma$ (assuming 2 classes), where $\gamma > 0$
  - So true error of the classifier is only slightly better than random.

- Questions:
  - How do we re-weight the examples?
  - How do we combine many simple predictors into a single classifier?

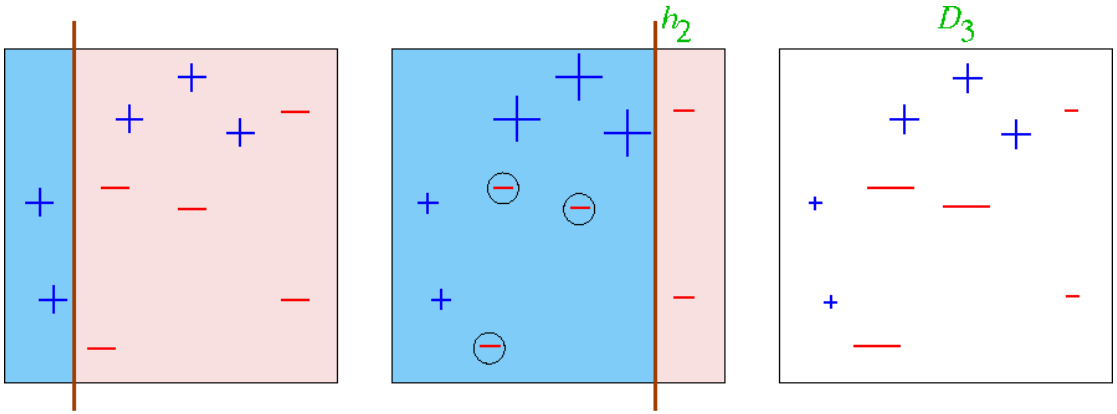
Example



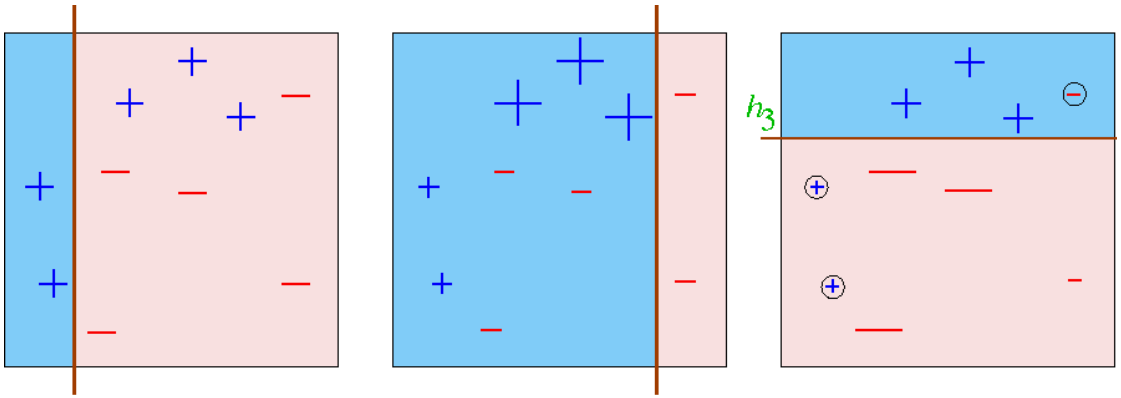
Example: First step



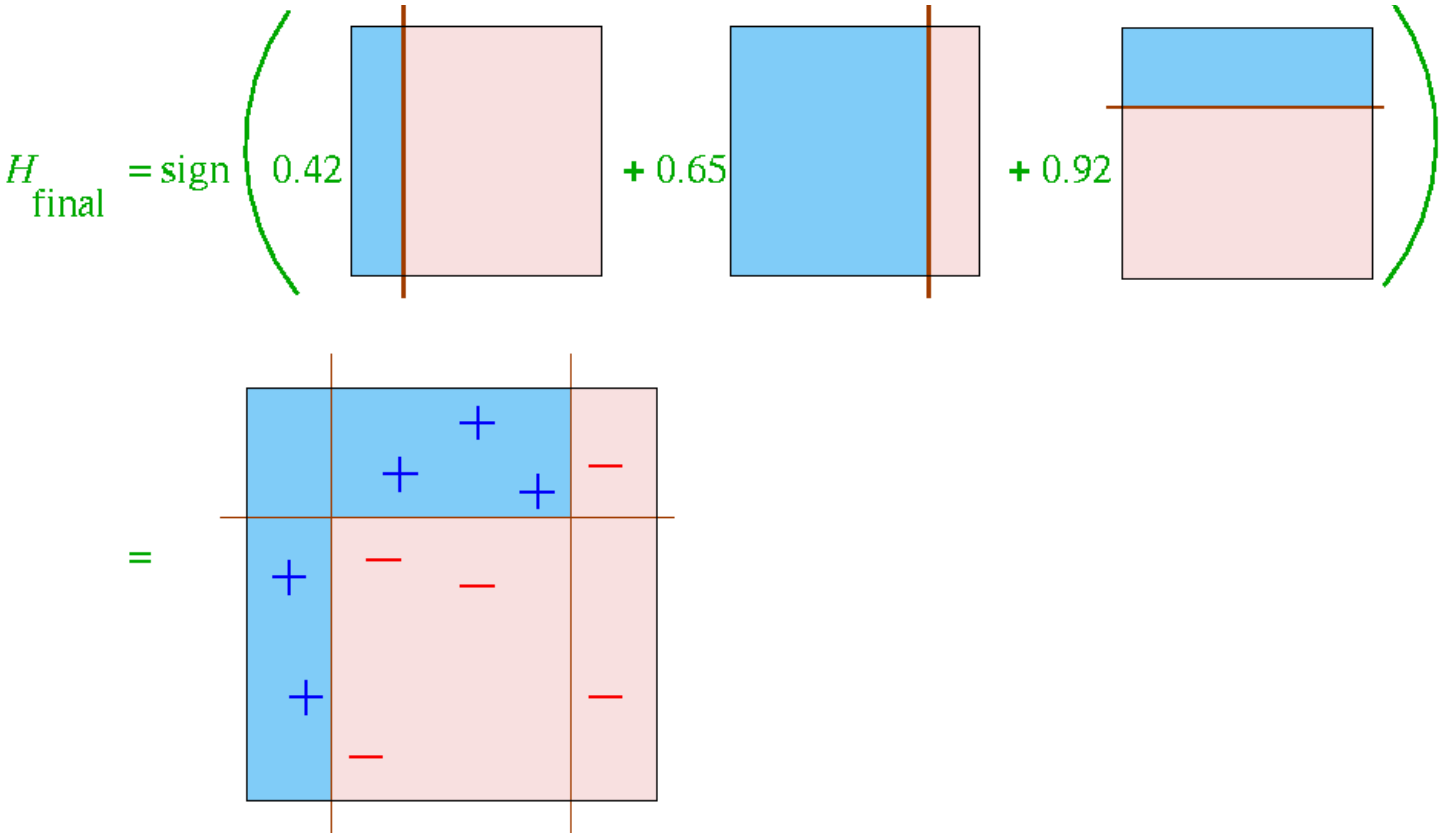
Example: Second step



Example: Third step



Example: Final hypothesis



AdaBoost (Freund & Schapire, 1995)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

AdaBoost (Freund & Schapire, 1995)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t} \left[h_t(x_i) \neq y_i \right].$$

AdaBoost (Freund & Schapire, 1995)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t} \left[h_t(x_i) \neq y_i \right].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$. weight of weak learner *t*

AdaBoost (Freund & Schapire, 1995)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t} \left[h_t(x_i) \neq y_i \right].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$. weight of weak learner *t*
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

AdaBoost (Freund & Schapire, 1995)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t} \left[h_t(x_i) \neq y_i \right].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$. weight of weak learner *t*
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

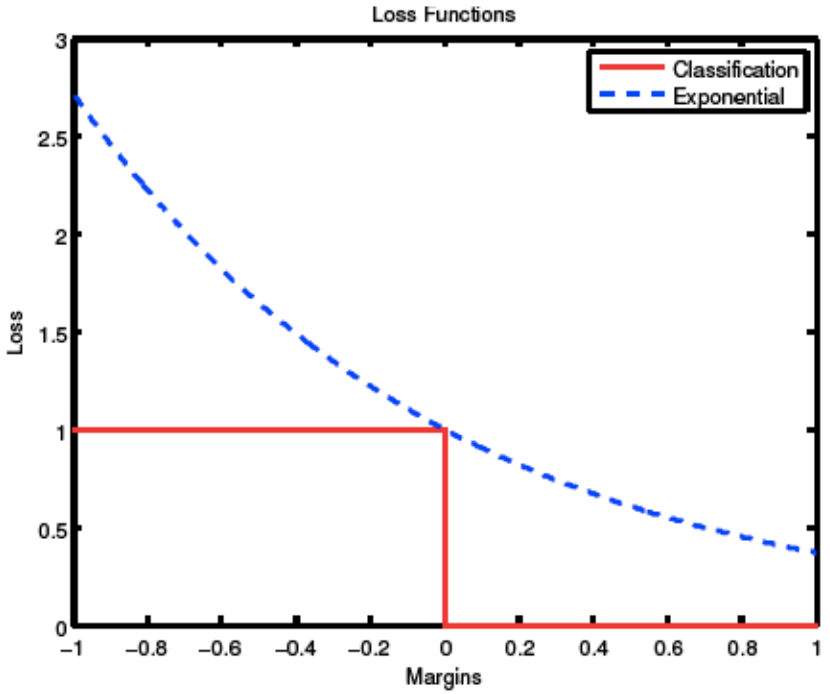
$$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Why these equations?

- Loss function:

$$L = \sum_{i=1}^{N} e^{-m_i}$$

$$m_i = y_i \sum_{k=1}^{K} \alpha_k h_k(x_i)$$



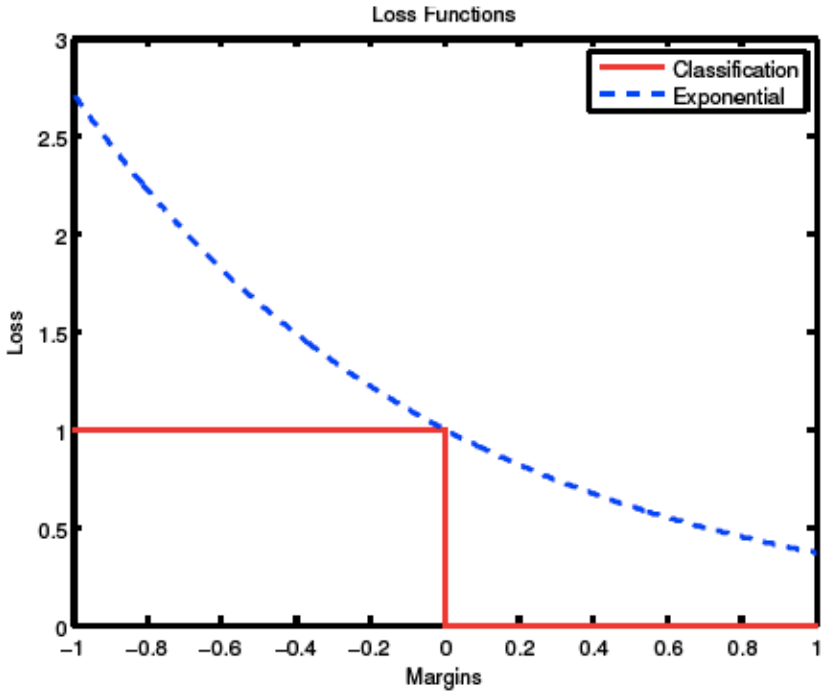
- Has a gradient
- Upper bound on classification loss
- Stronger signal for wrong classifications
- Stronger signal if wrong and far from boundary

Why these equations?

- Loss function:

$$L = \sum_{i=1}^{N} e^{-m_i}$$

$$m_i = y_i \sum_{k=1}^{K} \alpha_k h_k(x_i)$$



- Update equations are derived from this loss function

Properties of AdaBoost

- Compared to other boosting algorithms, main insight is to automatically adapt the error rate at each iteration.

Properties of AdaBoost

- Compared to other boosting algorithms, main insight is to automatically adapt the weights at each iteration.

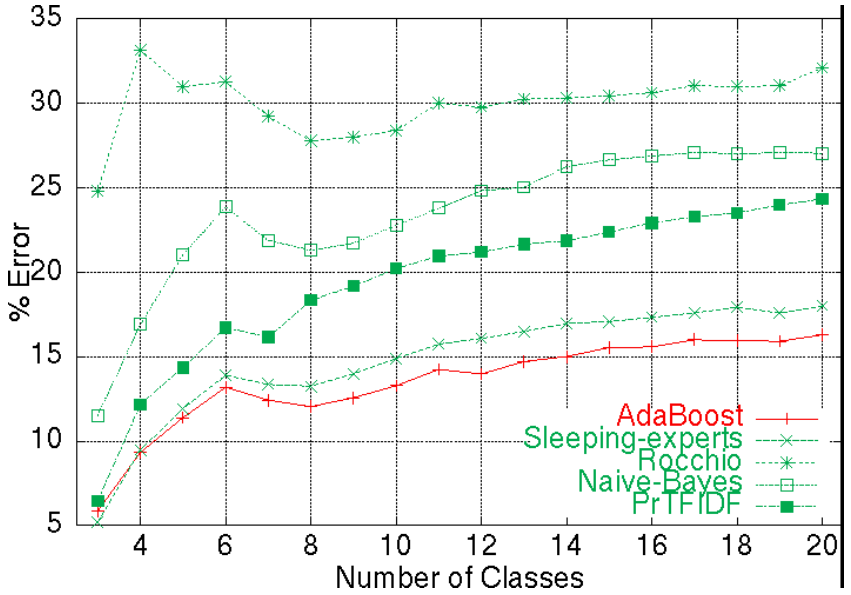
- Training error on the final hypothesis is at most:

$$\prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_t \gamma_t^2\right)$$

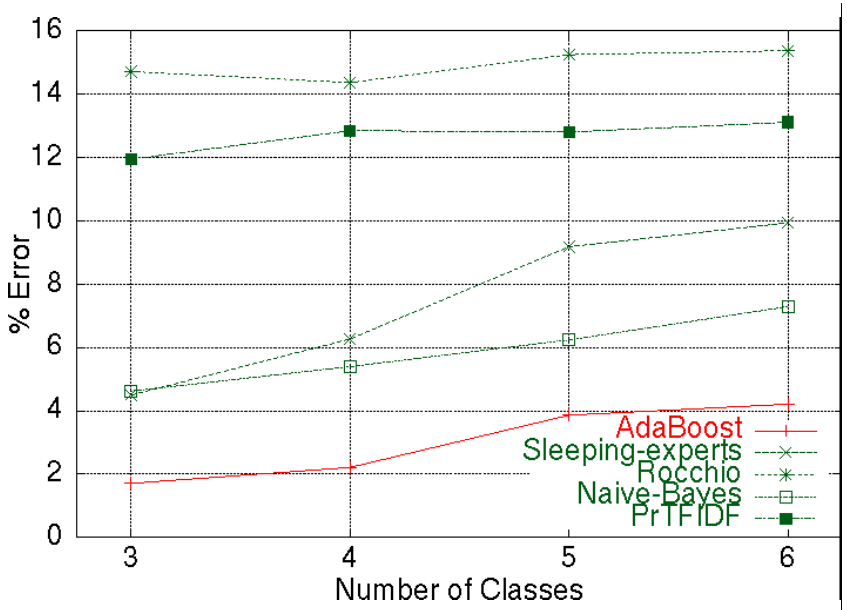
recall: γ_t is how much better than random is h_t

- AdaBoost gradually reduces the training error exponentially fast.

Real data set: Text categorization

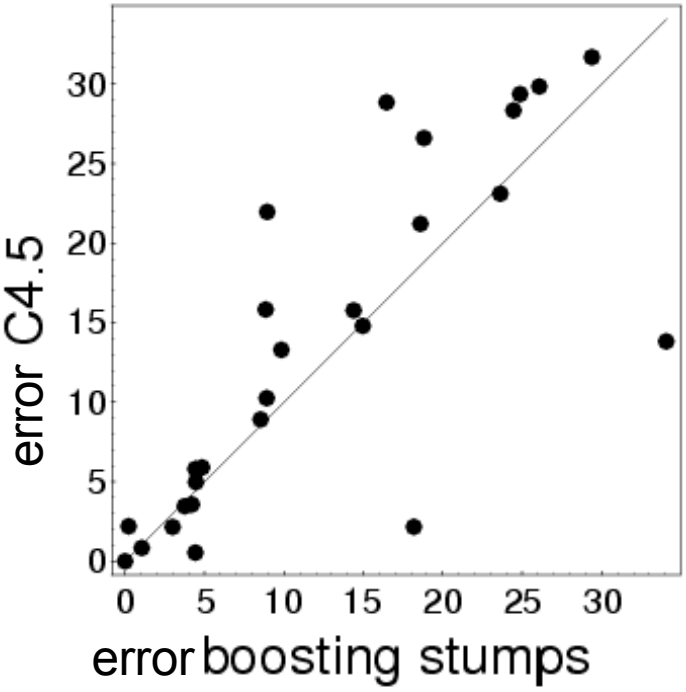


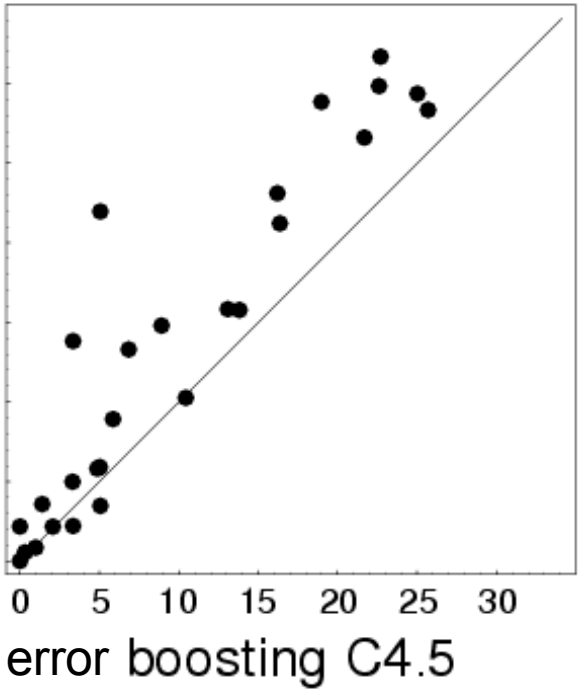
database: AP



database: Reuters

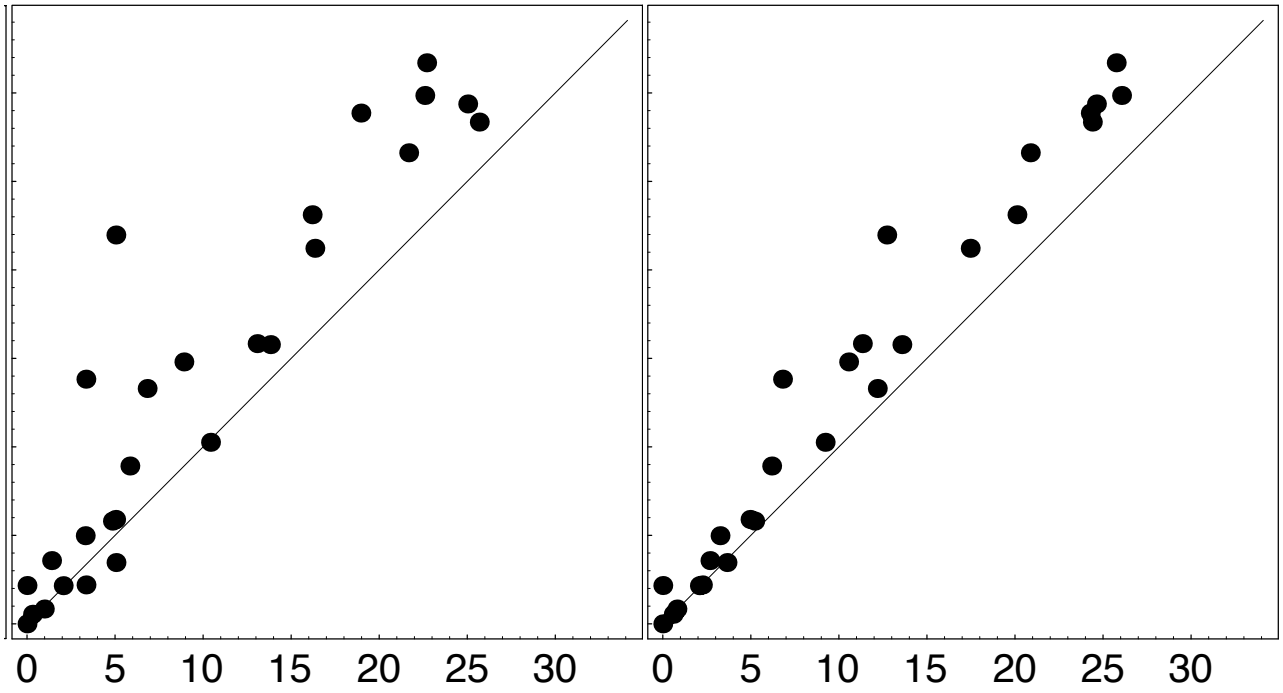
Boosting empirical evaluation





C4.5: Lecture 7 – Decision Trees

Bagging vs Boosting



boosting C4.5 bagging C4.5

Bagging vs Boosting

- Bagging is typically faster, but may get a smaller error reduction (not by much).
- Bagging works well with “reasonable” classifiers.
- Boosting works with very simple classifiers.

  E.g., Boostexter - text classification using decision stumps based on single words.
- Boosting may have a problem if a lot of the data is mislabeled, because it will focus on those examples a lot, leading to overfitting.

Why does boosting work?

Why does boosting work?

- Weak learners have high bias. By combining them, we get more expressive classifiers. Hence, boosting is a bias-reduction technique.

Why does boosting work?

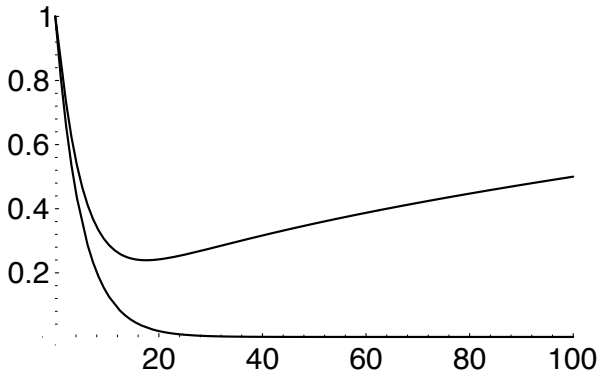
- Weak learners have high bias. By combining them, we get more expressive classifiers. Hence, boosting is a bias-reduction technique.

- Adaboost minimizes an upper bound on the misclassifcation error, within the space of functions that can be captured by a linear combination of the base classifiers.

- What happens as we run boosting longer? Intuitively, we get more and more complex hypotheses. How would you expect bias and variance to evolve over time?

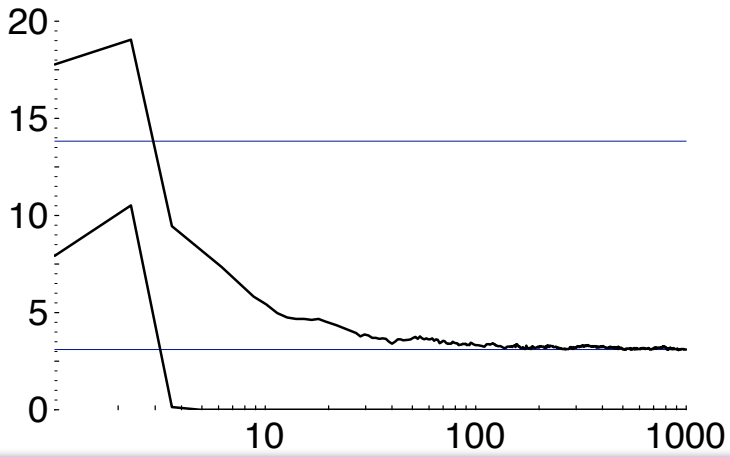
A naïve (but reasonable) analysis of error

- Expect the training error to continue to drop (until it reaches 0).
- Expect the test error to increase as we get more voters, and h_f becomes too complex.



Actual typical run of AdaBoost

- Test error does not increase even after 1000 runs! (more than 2 million decision nodes!)
- Test error continues to drop even after training error reaches 0!
- These are consistent results through many sets of experiments!
- Conjecture: Boosting does not overfit!



What you should know

- Ensemble methods combine several hypotheses into one prediction.
- They work better than the best individual hypothesis from the same class because they reduce bias or variance (or both).
- Extremely randomized trees are a bias-reduction technique.
- Bagging is mainly a variance-reduction technique, useful for complex hypotheses.
- Main idea is to sample the data repeatedly, train several classifiers and average their predictions.
- Boosting focuses on harder examples, and gives a weighted vote to the hypotheses.
- Boosting works by reducing bias and increasing classification margin.