

COMP 551 - Applied Machine Learning

Lecture 12 – Ensemble methods

William L. Hamilton

(with slides and content from Joelle Pineau)

* Unless otherwise noted, all material posted for this course are copyright of the instructor, and cannot be reused or reposted without the instructor's written permission.

MiniProject 2

- Deadline extended by 12 hours due to Kaggle downtime.
- If you still cannot get >50% on Kaggle, come to my office after class.
 - The likely reason is that you are not loading the test data in the right order. You cannot assume that functions like `os.listdir(...)` will iterate over files in a fixed order.
 - Examples of correct ways to load the data:

```
def read_test1(data_path):
    data = []
    folder = "test/"
    for file_name in os.listdir(folder):
        with open(os.path.join(folder, file_name), 'rb') as f:
            review = f.read().decode('utf-8').replace('\n', '').strip().lower()
            data.append([review, int(file_name.split(".")[0])])
    return data

def read_test2(data_path):
    data = []
    folder = "test/"
    for i in range(25000):
        with open(os.path.join(folder, str(i)+".txt"), 'rb') as f:
            review = f.read().decode('utf-8').replace('\n', '').strip().lower()
            data.append([review, i])
    return data
```

MiniProject 2

- There is no minimum accuracy for Naïve Bayes. It just needs to be a correct implementation.
- YOUR LEARNED MODEL MUST NOT USE ANY INFORMATION FROM THE TEST SET.
 - You obviously need to make predictions on the test set but that is the only interaction with the test set that you should have.
 - Cheating by searching for information about the test set or trying to manually label examples in the test set will result in a grade of 0.
 - The teaching staff has a very good sense of how well different models should perform, so cheating to get better Kaggle results and then trying to present a different model in your report will be easy to catch!!!

Next topic: Ensemble methods

- Recently seen supervised learning methods:
 - Logistic regression, Naïve Bayes, LDA/QDA
 - Decision trees, Instance-based learning
 - SVMs
- Now we will discuss the idea of how we can combine the output of different models, an idea called **ensembling**.

Ensemble learning in general

- **Key idea:** Run base learning algorithm(s) 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, ...
- Option 1 (Bagging): Construct K independent models by training the same base learning algorithm on different subsets of the training data.
- Option 2 (Boosting): Incrementally train K models, where each successive model tries to fix the mistakes of the previous models.
- Option 3 (Stacking): Train K different models and combine their output using a “meta-classifier”.

Ensemble learning in general

- **Key idea:** Run base learning algorithm(s) 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, ...
- Option 1 (Bagging): Construct K independent models by training the same base learning algorithm on different subsets of the training data.
- Option 2 (Boosting): Incrementally train K models, where each successive model tries to fix the mistakes of the previous models.

Generally assume that the same base learning algorithm is used to construct all models in the ensemble.
- Option 3 (Stacking): Train K different models and combine their output using a “meta-classifier”.

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 B times, where B is typically a few hundred.
- Similar to cross-validation but using random sampling with replacement.

Estimating bias and variance

- In general, bootstrapping is useful for computing sound statistical estimates of the bias and variance for our model.
- For each point \mathbf{x} , we have a set of estimates $h_1(\mathbf{x}), \dots, h_K(\mathbf{x})$, with $K \leq B$ (since \mathbf{x} might not appear in some replicates).
- The average empirical prediction of \mathbf{x} is: $\hat{h}(\mathbf{x}) = (1/K) \sum_{k=1:K} h_k(\mathbf{x})$.
- We estimate the bias as: $y - \hat{h}(\mathbf{x})$.
- We estimate the variance as: $(1/(K-1)) \sum_{k=1:K} (\hat{h}(\mathbf{x}) - h_k(\mathbf{x}))^2$.

Bagging: Bootstrap aggregation

- If we did all the work to get the hypotheses h_k , 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 \mathbf{x} , we have a set of estimates $h_1(\mathbf{x}), \dots, h_K(\mathbf{x})$, with $K \leq B$ (since \mathbf{x} might not appear in some replicates).
 - The average empirical prediction of \mathbf{x} is: $\hat{h}(\mathbf{x}) = (1/K) \sum_{k=1:K} h_k(\mathbf{x})$.
 - We estimate the bias as: $y - \hat{h}(\mathbf{x})$.
 - We estimate the variance as: $(1/(K-1)) \sum_{k=1:K} (\hat{h}(\mathbf{x}) - h_k(\mathbf{x}))^2$.

Bagging

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

Random forests (Breiman)

- 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)

- 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., 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.

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.

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 of continuous data, ...
- 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 of continuous data, ...
- **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

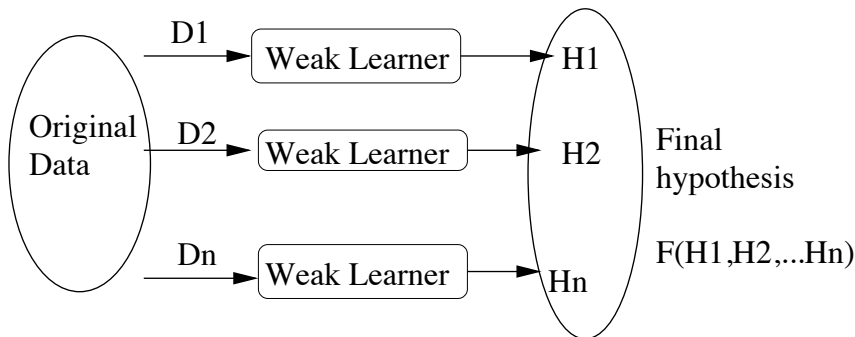
- Instead of searching very hard for the best hypothesis, generate lots of random ones, then average their results.
- Examples: Random feature selection, random projections of continuous data, ...
- **Advantages?**
 - Very fast, easy, can handle lots of data.
 - Can circumvent difficulties in optimization.
 - 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 (e.g., compared to standard decision trees).

Incrementally constructing an ensemble

- In an ensemble, the output on any instance is computed by combining 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

- Basic algorithm:
 - 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 :

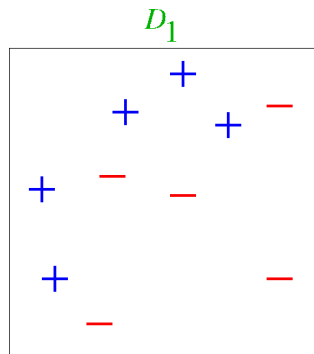
$$J_P(h) = \sum_{\langle \mathbf{x}, y \rangle \in D} J(h(\mathbf{x}), y) P(\langle \mathbf{x}, y \rangle)$$

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

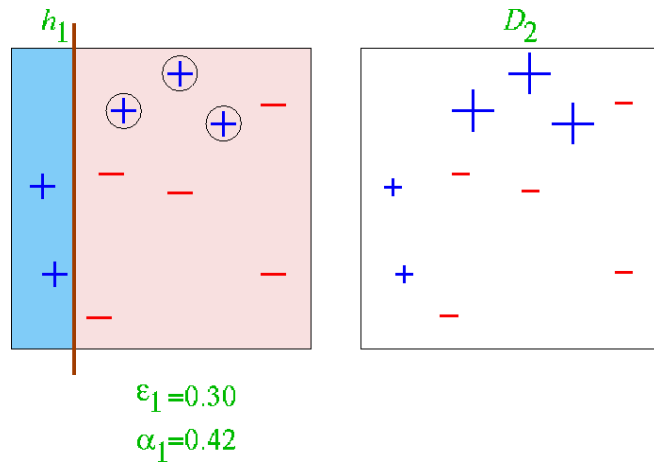
Weak learners

- 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?

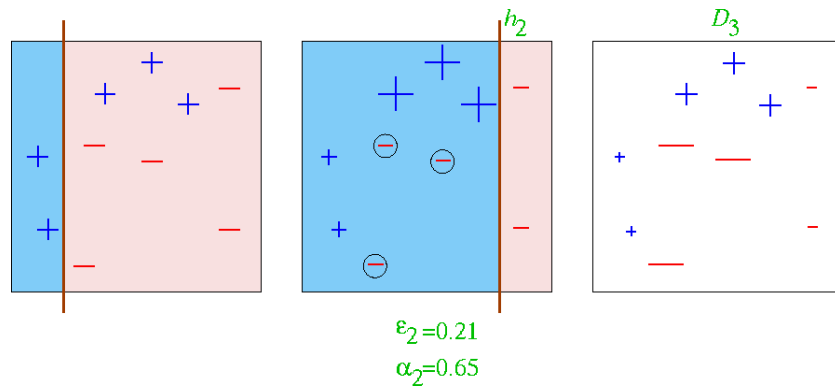
Example



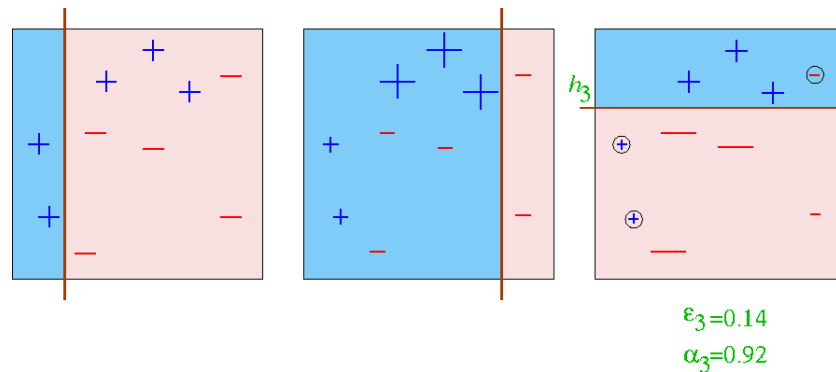
Example: First step



Example: Second step



Example: Third step



Example: Final hypothesis

$$H_{\text{final}} = \text{sign} \left(0.42 \begin{array}{|c|c|} \hline \text{blue} & \text{pink} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{blue} & \text{pink} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{blue} & \text{pink} \\ \hline \end{array} \right)$$

$$= \begin{array}{|c|c|c|} \hline \text{blue} & \text{blue} & \text{pink} \\ \hline \text{blue} & \text{pink} & \text{pink} \\ \hline \end{array}$$

The diagram illustrates the final hypothesis H_{final} as a weighted sum of three weak hypotheses. Each weak hypothesis is represented by a square divided into two regions: a blue region and a pink region, separated by a vertical line. The weights are 0.42, 0.65, and 0.92. The final hypothesis is the sign of the sum of these weighted hypotheses. The resulting hypothesis is shown as a square divided into four regions by two vertical lines, with blue regions containing '+' signs and pink regions containing '-' signs.

AdaBoost (Freund & Schapire, 1995)

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

AdaBoost (Freund & Schapire, 1995)

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

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

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

AdaBoost (Freund & Schapire, 1995)

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

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

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

- 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: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

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

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

- 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: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

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

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

- 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:

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

Won the Gödel Prize in 2003.

Properties of AdaBoost

- Compared to other boosting algorithms, main insight is to **automatically adapt the error rate** 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 = how much better than random is h_t

- AdaBoost gradually reduces the training error exponentially fast.

Why does boosting work?

Why does boosting work?

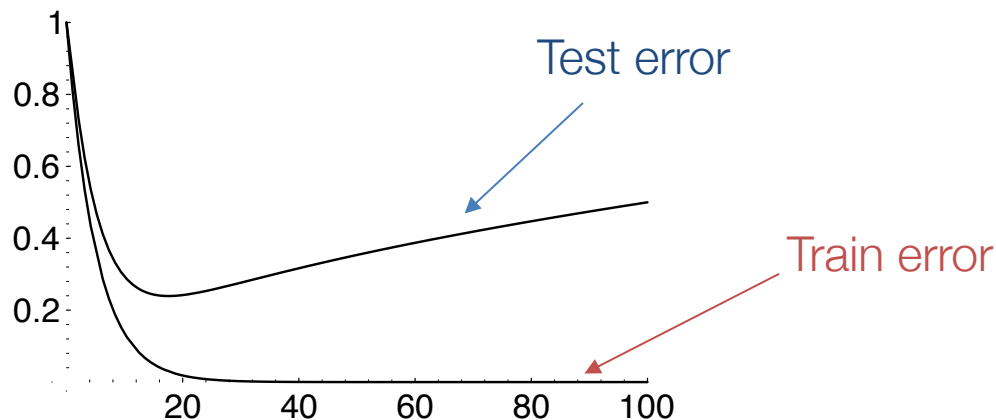
- Weak learners have high bias. By combining them, we get more expressive classifiers. Hence, boosting is a **bias-reduction technique**.

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 looks for a good **approximation to the log-odds ratio**, 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?**

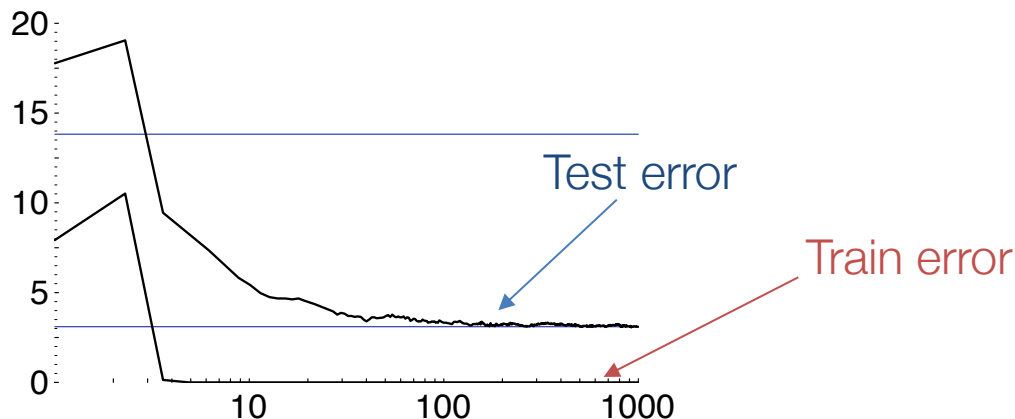
A naïve (but reasonable) analysis of error

- 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!



Bagging vs Boosting

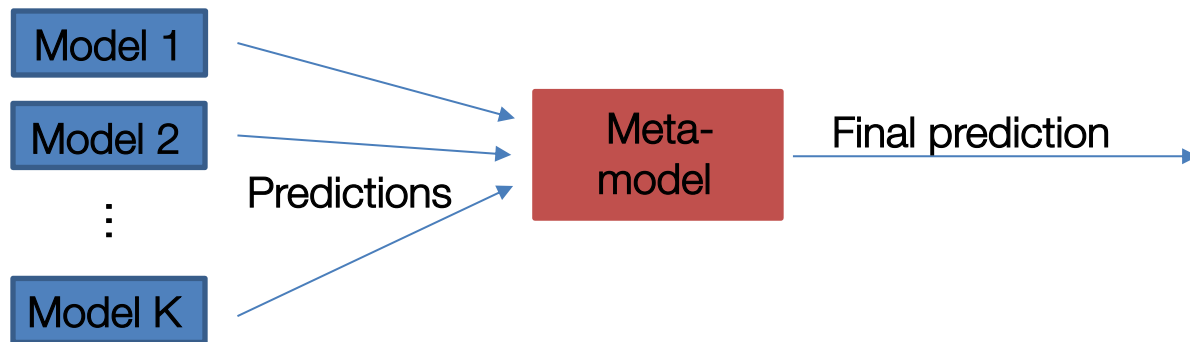
- Bagging is typically faster, but may get a smaller error reduction (not by much).
- Bagging works well with “reasonable” classifiers.
- Boosting works with very simple classifiers.
 - E.g., Boostexter - text classification using decision stumps based on single words.
- Boosting may have a problem if a lot of the data is mislabeled, because it will focus on those examples a lot, leading to overfitting.

Stacking

- Both bagging and boosting assume we have a single “base learning” algorithm.
- But what if we want to ensemble an arbitrary set of classifiers?
 - E.g., combine the output of a SVM, naïve Bayes, and a nearest neighbor model?

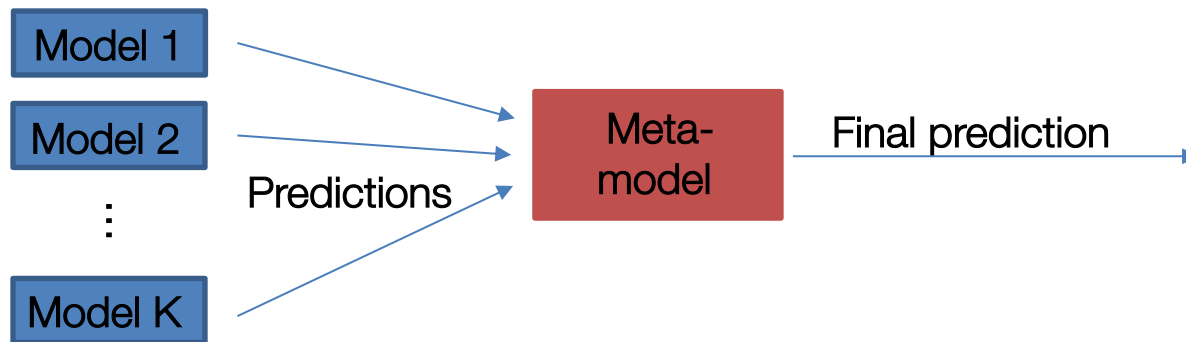
Stacking

- Both bagging and boosting assume we have a single “base learning” algorithm.
- But what if we want to ensemble an arbitrary set of classifiers?
 - E.g., combine the output of a SVM, naïve Bayes, and a nearest neighbor model?



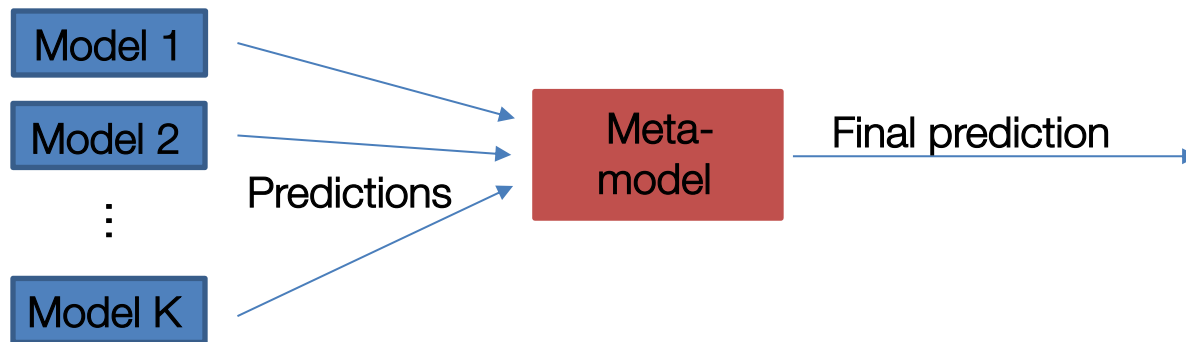
Stacking

- Train K distinct base models, $h_k, k=1 \dots K$, on the training set $\langle \mathbf{x}_i, y_i \rangle, i=1 \dots n$
- Make a new training set where the new features are the predictions of the base models: $\langle h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \dots, h_K(\mathbf{x}_i), y_i \rangle, i=1 \dots n$
 - (Can also add the original feature information, \mathbf{x}_i , to the new training set)
- Train a meta-model m on this new training set.
 - (Possibly train multiple meta-models and repeat the process.)



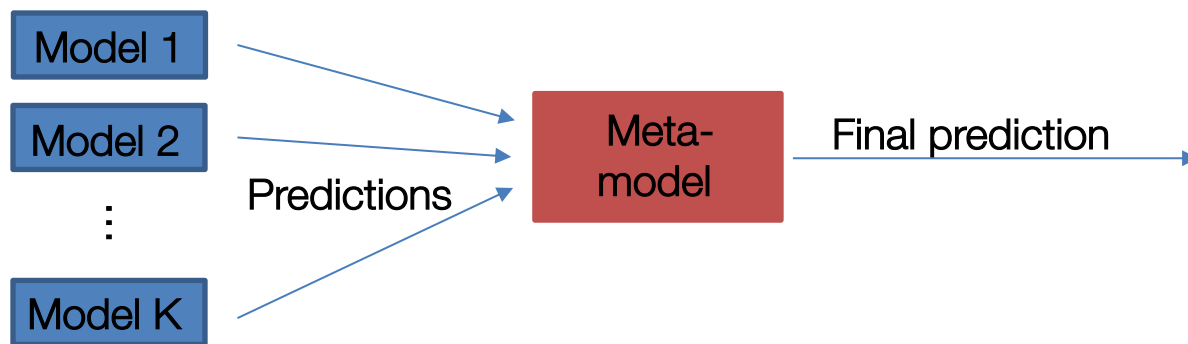
Meta-model

- What is the meta-model? Any supervised model will work!
- Common choices:
 - Averaging (for regression tasks)
 - Majority vote (for classification tasks)
 - Linear regression (for regression tasks)
 - Logistic regression (for classification tasks)



When does stacking work?

- Stacking works best when the base models have **complimentary strengths and weaknesses** (i.e., different inductive biases).
- For example: combining k-nearest neighbor models with different k-values, Naïve Bayes, and logistic regression. Each of these models has different underlying assumptions so (hopefully) they will be complimentary.



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).
- Bagging is mainly a variance-reduction technique, useful for complex hypotheses.
- Boosting focuses on harder examples, and gives a weighted vote to the hypotheses.
- Boosting works by reducing bias and increasing classification margin.
- Stacking is a generic approach to ensemble various models and performs very well in practice.