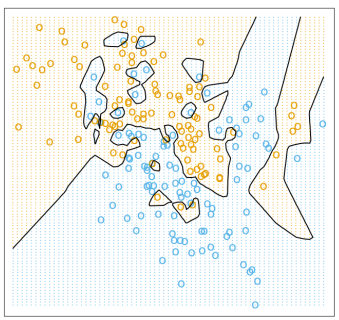
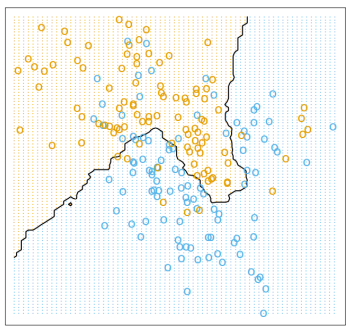
Ensembling Method

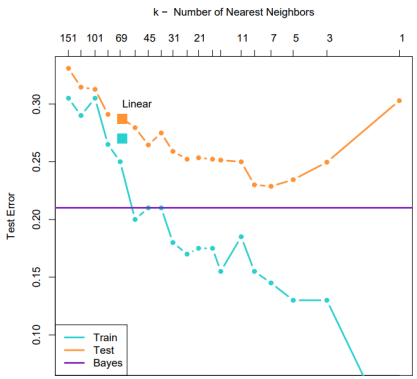
Bias-Variance Decomposition

Quando facciamo delle stime, abbiamo una media E[y]=y.
Ogni singola misura può spostarsi da questa media, è abbastanza normale, ma la media deve essere quella. Se mi discosto da quella media, ho un BIAS. Non è una varianza attenzione!
Varianza: quanto mi sposto dalla media, es: media = 10, varianza = 3, i valori vanno da 7 a 13.
Bias: media = 10, ma io ottengo media = 14, quindi sto distante dalla media.

- Recall that overly simple models underfit the data, and overly complex models overfit.







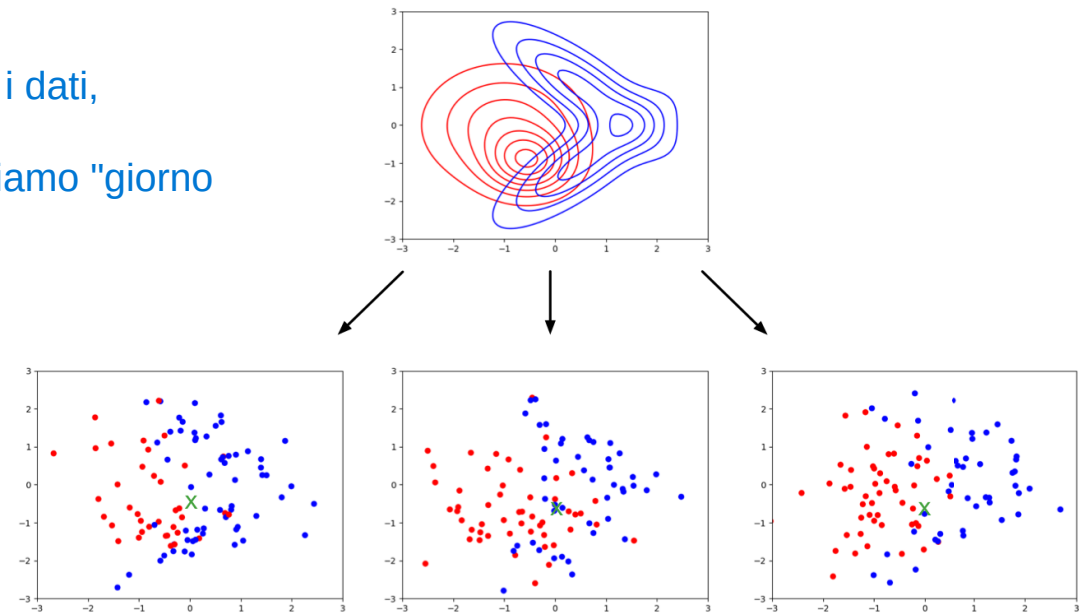
- We can quantify this effect in terms of the bias/variance decomposition.
- Bias and variance of what?

Bias-Variance Decomposition: Basic Setup

< valore riga "i" associato alla colonna t(i)">

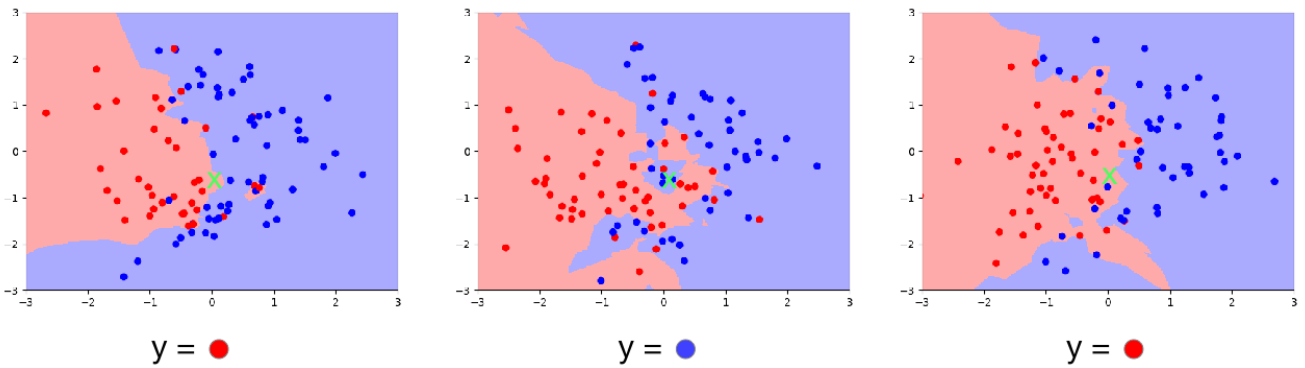
- Suppose the training set consists of pair $(\mathbf{x}^{(i)}, t^{(i)})$ sampled independent and identically distributed (i.i.d.) from a single data generating distribution P_{sample}.
- Pick a fixed query point **x** (the green x in the figure)
- Consider an experiment where we sample lot training sets independently from P_{sample}

qui sto partizionando i dati,
indipendenti,
ad esempio partizioniamo "giorno per giorno".



Bias-Variance Decomposition: Basic Setup

- Let's run our learning algorithm on each training set, and compute its prediction y at the query point $\mathbf{x}$.
- We can view y as a random variable, where the randomness comes from the choice of training set.
- The classiffication accuracy is determined by the distribution of y.

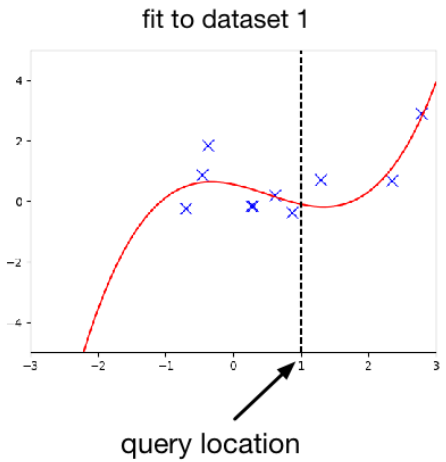


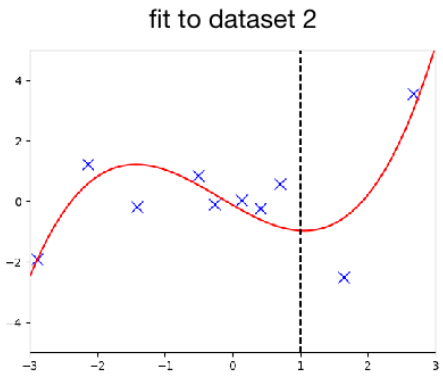
ora, su questi training set, spezzati giorno per giorno, eseguo l'algoritmo di learning.
Quindi sto semplicemente applicando quello che ho fatto fino ad ora, ma separando i dati che sono raggruppati secondo qualche criterio. Per ogni gruppo faccio una predizione, che sarà diversa di gruppo in gruppo probabilmente.

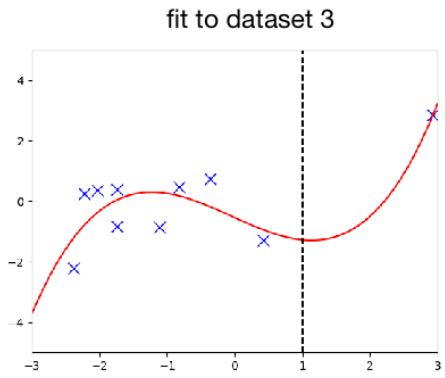
Bias-Variance Decomposition: Basic Setup

Here is the analogous for regression

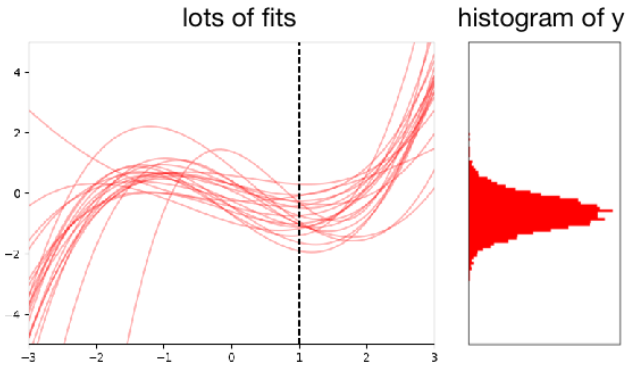
qui abbiamo dei grafici rossi che fittano i vari dataset.







fissiamo una certa "x", e vediamo il valore che assume y(x) in ogni dataset.



come vediamo, mettendo insieme sovraimpressi tutti i dataset, abbiamo delle y diverse per la 'x' cercata, e quindi y è una variabile aleatoria (x è fissa), con tutti gli aspetti di una variabile aleatoria, di conseguenza

Since y is a random variable, we can talk about its expectation, variance, etc.

L'accuratezza dipende dalla distribuzione di "y". Dall'istogramma vediamo che la forma è simil-gaussiana.

Vogliamo: media = y, varianza -> 0, bias ->0 (-> è tendente)

Bias-Variance Decomposition: Basic Setup

- Recap the basic setup
  - Fix a query point $\mathbf{x}$
  - Repeat:
    - Sample a random training dataset $(\mathbf{x}^{(i)}, t^{(i)})$ from the data generating distribution P_{sample}
    - Run the learning algorithm on $(\mathbf{x}^{(i)}, t^{(i)})$ to get a prediction y at $\mathbf{x}$
    - Sample the true target from the conditional distribution $P(y \mid \mathbf{x})$
- This gives a distribution over the loss at $\mathbf{x}$, with expectation $E[\mathcal{L}(y; t) \mid \mathbf{x}]$.
- For each query point $\mathbf{x}$, the expected loss is different. We are interested in minimizing the expectation of this with respect to $\mathbf{x}$ sampled according to P_{sample}.

Bias-Variance Decomposition

- Assume that $t = g(\mathbf{x}) + \epsilon$
  - Noise ϵ is sampled from a normal distribution with mean 0 and variance σ^2 : $\epsilon \sim N(0, \sigma^2)$
  - Noise lower-bounds the performance we can achieve
- Recall we want to minimize the objective function

$$\mathcal{J}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(t^{(i)} - f_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)$$

- We can re-write this as the expected value of the squared error: $E(t - f_{\mathbf{w}}(\mathbf{x}))^2$

Bias-Variance Decomposition

$$
\begin{aligned}
E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] &= E[(t - g(\mathbf{x}) + g(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}))^2] \\
&= E[(t - g(\mathbf{x}))^2] + E[(g(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}))^2] \\
&\quad + 2E[(g(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}))(t - g(\mathbf{x}))] \\
&= E[(t - g(\mathbf{x}))^2] + E[(g(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}))^2] \\
&\quad + 2(E[g(\mathbf{x})f_{\mathbf{w}}(\mathbf{x})] + E[tg(\mathbf{x})] - E[tf_{\mathbf{w}}(\mathbf{x})] - E[g(\mathbf{x})^2])
\end{aligned}
$$

the last four terms cancel out...Therefore

$$
\begin{aligned}
E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] &= E[(t - g(\mathbf{x}))^2] + E[(g(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}))^2] \\
&= Var[\epsilon] + E[(g(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}))^2]
\end{aligned}
$$

Bias-Variance Decomposition

$$
\begin{aligned}
E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] &= Var[\epsilon] + E[(g(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}))^2] \\
&= Var[\epsilon] + E[(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})] + E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))^2] \\
&= Var[\epsilon] + E[(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})])^2] + E[(E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))^2] \\
&\quad + 2E[(E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})])] \\
&= Var[\epsilon] + E[(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})])^2] + E[(E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))^2] \\
&\quad + 2(E[g(\mathbf{x})E[f_{\mathbf{w}}(\mathbf{x})]] - E[E[f_{\mathbf{w}}(\mathbf{x})]^2] - E[g(\mathbf{x})f_{\mathbf{w}}(\mathbf{x})] + E[f_{\mathbf{w}}(\mathbf{x})E[f_{\mathbf{w}}(\mathbf{x})]])
\end{aligned}
$$

...by simalar arguments as before we get

$$E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] = Var[\epsilon] + E[(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})])^2] + E[(E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))^2]$$

$$E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] = \text{bias}(f_{\mathbf{w}}(\mathbf{x}))^2 + \text{var}(f_{\mathbf{w}}(\mathbf{x})) + \sigma^2$$

Bias-Variance Decomposition

$$E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] = \text{bias}(f_{\mathbf{w}}(\mathbf{x}))^2 + \text{var}(f_{\mathbf{w}}(\mathbf{x})) + \sigma^2$$

- We just split the expected loss into three terms:
  - bias: how wrong the expected prediction is (corresponds to underfitting)
  - variance: the amount of variability in the predictions (corresponds to overfitting)
  - Bayes error: the inherent unpredictability of the targets
- Even though this analysis only applies to squared error, we often loosely use ”bias” and ”variance" as synonyms for ”underfitting" and ”overfitting".

Bagging: Motivation

- Suppose we could somehow sample m independent training sets from P_{sample}.
- We could then compute the prediction y_i based on each one, and take the average $y = \frac{1}{m}\sum_{i=1}^{m} y_i$.
- How does this affect the three terms of the expected loss?
  - **Bayes error: unchanged**, since we have no control over it
  - **Bias: unchanged,** since the averaged prediction has the same expectation

$$E[y] = E\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = E[y_i]$$

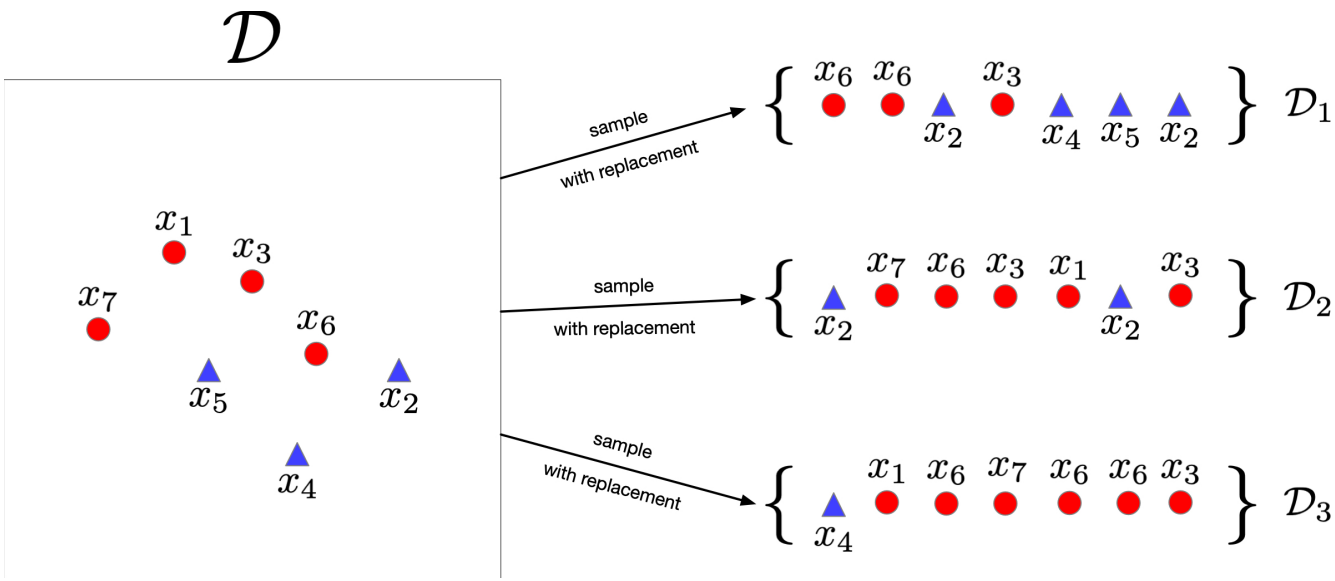
- **Variance: reduced**, since we are avaraging over independent samples

$$Var[y] = Var\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m^2}\sum_{i=1}^{m} Var[y_i] = \frac{1}{m} Var[y_i]$$

Bagging: The Idea

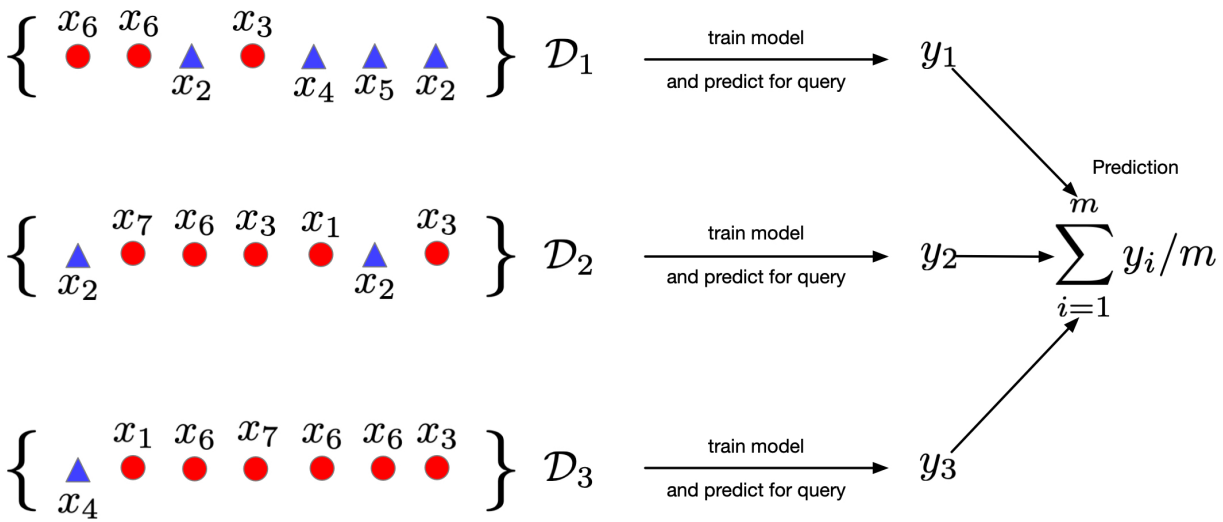
- In practice, the sampling distribution P_{sample} is often finite or expensive to sample from.
- So training separate models on independently sampled datasets is very wasteful of data!
- **Solution:** given training set $\mathcal{D}$, use the empirical distribution $P_{\mathcal{D}}$ as a proxy for P_{sample}. This is called bootstrap aggregation, or bagging.
  - Take a single dataset $\mathcal{D}$ with n examples.
  - Generate m new datasets ('resamples" or "bootstrap samples"), each by sampling n training examples from $\mathcal{D}$, with replacement.
  - Average the predictions of models trained on each of these datasets.
- The bootstrap is one of the most important ideas in all of statistics!
  - Intuition: As $|\mathcal{D}| \to \infty$, we have $P_{\mathcal{D}} \to P_{sample}$.

Bagging



In this example $n = 7$, $m = 3$

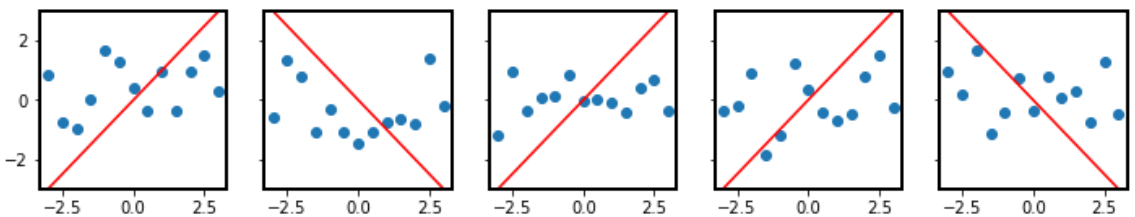
Bagging



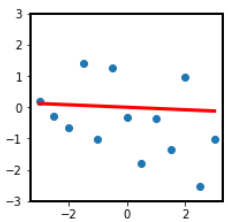
predicting on a query point x

Bagging: Effect on Hypothesis Space

- We saw that in case of squared error, bagging does not affect bias.
- But it can change the hypothesis space
- Illustrative example:
  - $x \sim \mathcal{U}(-3, 3), t \sim \mathcal{N}(0, 1)$
  - $\mathcal{H} = \{wx | w \in \{-1, 1\}\}$
  - Sampled datasets & fitted hypotheses:



  - Ensembled hypotheses (mean over 1000 samples):



  - The ensembled hypothesis is not in the original hypothesis space!

Bagging for Binary Classification

- If our classifiers output real-valued probabilities, $z_i \in [0, 1]$, then we can average the predictions before thresholding:

$$y_{bagged} = \mathbb{I}(z_{bagged} > 0,5) = \mathbb{I}\left(\sum_{i=1}^{m} \frac{z_i}{m} > 0,5\right)$$

- If our classiffers output binary decisions, $y_i \in \{0; 1\}$, then we can average the predictions before thresholding:

$$y_{bagged} = \mathbb{I}\left(\sum_{i=1}^{m} \frac{y_i}{m} > 0,5\right)$$

  This is the same as taking a majority vote.
- A bagged classifier can be stronger than the average underyling model.

Bagging: Effect of Correlation

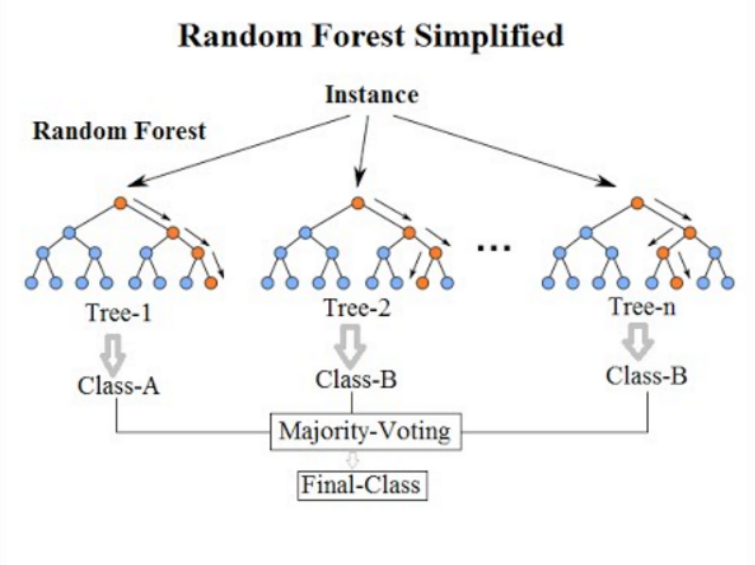
- Problem: the datasets are not independent, so we don't get the $1/m$ variance reduction.
  - it is possible to show that if the and correlation ρ, then

$$Var[y] = Var\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2$$

- It can be advantageous to introduce additional variability into your algorithm, as long as it reduces the correlation between samples.
  - Can help to use average over multiple algorithms, or multiple configurations of the same algorithm.

Random Forests

- Random forests = bagged decision trees, with one extra trick to decorrelate the predictions
  - When choosing each node of the decision tree, choose a random set of *d* input features, and only consider splits on those features



- Random forests are probably the best black-box machine learning algorithm - they often work well with no tuning whatsoever.
  - one of the most widely used algorithms in Kaggle competitions

Bagging Summary

- Bagging reduces overfitting by averaging predictions.
- Used in most competition winners
  - Even if a single model is great, a small ensemble usually helps.
- Limitations:
  - Does not reduce bias in case of squared error.
  - There is still correlation between classifiers.
  - Random forest solution: Add more randomness.
  - Naive mixture (all members weighted equally).
  - If members are very different (e.g., different algorithms, different data sources, etc.), we can often obtain better results by using a principled approach to weighted ensembling.
- Boosting, up next, can be viewed as an approach to weighted ensembling that strongly decorrelates ensemble members.

Boosting

- Boosting
  - Train classifiers sequentially, each time focusing on training examples that the previous ones got wrong.
  - The shifting focus strongly decorrelates their predictions.
- To focus on specific examples, boosting uses a weighted training set

Weighted Training Set

- The misclassification rate $\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$ weights each training set equally
- **Key Idea**: we can learn a classifier using different costs (aka weights) for examples.
  - classifier “tries harder" on examples with higher cost
- Change cost function:

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}[h(x^{(n)}) \neq t^{(n)}] \quad \text{becomes} \quad \frac{1}{N}\sum_{n=1}^{N} w^{(n)}\mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$$

- Usually require $w^{(n)} > 0$ and $\sum_{n=1}^{N} w^{(n)} = 1$

AdaBoost [Freund & Shapire, 1997]

- A meta-learning algorithm with great theoretical and empirical performance
- Turns a base learner, i.e., a weak learner/classifier into a high performance classifier
- Creates an ensemble of weak learner by repeatedly emphasizing misspredicted instances

AdaBoost (Adaptive Boosting)

- We can now describe the AdaBoost algorithm.
- Given a base classifier, the key steps of AdaBoost are:
  1. At each iteration, re-weight the training samples by assigning larger weights to samples (i.e., data points) that were classiffied incorrectly.
  2. Train a new base classifier based on the re-weighted samples.
  3. Add it to the ensemble of classifiers with an appropriate weight.
  4. Repeat the process many times.
- Requirements for base classifier:
  - Needs to minimize weighted error.
  - Ensemble may get very large, so base classifier should be fast. It turns out that any so-called weak learner/classifier suffices.

  Individually, weak learners may have high bias (underfit). By making each classifier focus on previous mistakes, AdaBoost reduces bias.

Weak Learner/classifier

- (Informal) Weak learner is a learning algorithm that outputs a hypothesis (e.g., a classifier) that performs slightly better than chance, e.g., it predicts the correct label with probability 0.51 in binary label case.
- We are interested in weak learners that are computationally efficient.
  - Decision trees
  - Even simpler: Decision Stump: A decision tree with a single split