

CSC 411 Lecture 4: Ensembles I

Roger Grosse, Amir-massoud Farahmand, and Juan Carrasquilla

University of Toronto

- We've seen two particular classification algorithms: KNN and decision trees
- Next two lectures: combine multiple classifiers into an ensemble which performs better than the individual members
 - ▶ Generic class of techniques that can be applied to almost any learning algorithm...
 - ▶ ... but are particularly well suited to decision trees
- Today
 - ▶ Understanding generalization using the bias/variance decomposition
 - ▶ Reducing variance using bagging
- Next lecture
 - ▶ Making a weak classifier stronger (i.e. reducing bias) using boosting

Ensemble methods: Overview

- An **ensemble** of predictors is a set of predictors whose individual decisions are combined in some way to classify new examples
 - ▶ E.g., (possibly weighted) majority vote
- For this to be nontrivial, the classifiers must differ somehow, e.g.
 - ▶ Different algorithm
 - ▶ Different choice of hyperparameters
 - ▶ Trained on different data
 - ▶ Trained with different weighting of the training examples
- Ensembles are usually trivial to implement. The hard part is deciding what kind of ensemble you want, based on your goals.

- This lecture: [bagging](#)
 - ▶ Train classifiers independently on random subsets of the training data.
- Next lecture: [boosting](#)
 - ▶ Train classifiers sequentially, each time focusing on training examples that the previous ones got wrong.
- Bagging and boosting serve very different purposes. To understand this, we need to take a detour to understand the bias and variance of a learning algorithm.

Loss Functions

- A **loss function** $L(y, t)$ defines how bad it is if the algorithm predicts y , but the target is actually t .
- Example: **0-1 loss** for classification

$$L_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$

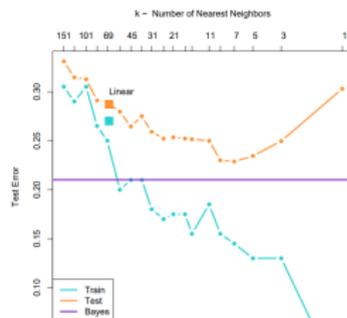
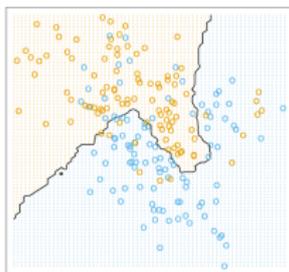
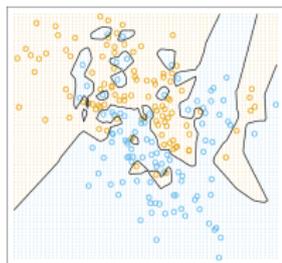
- ▶ Averaging the 0-1 loss over the training set gives the **training error rate**, and averaging over the test set gives the **test error rate**.
- Example: **squared error loss** for regression

$$L_{SE}(y, t) = \frac{1}{2}(y - t)^2$$

- ▶ The average squared error loss is called **mean squared error (MSE)**.

Bias-Variance Decomposition

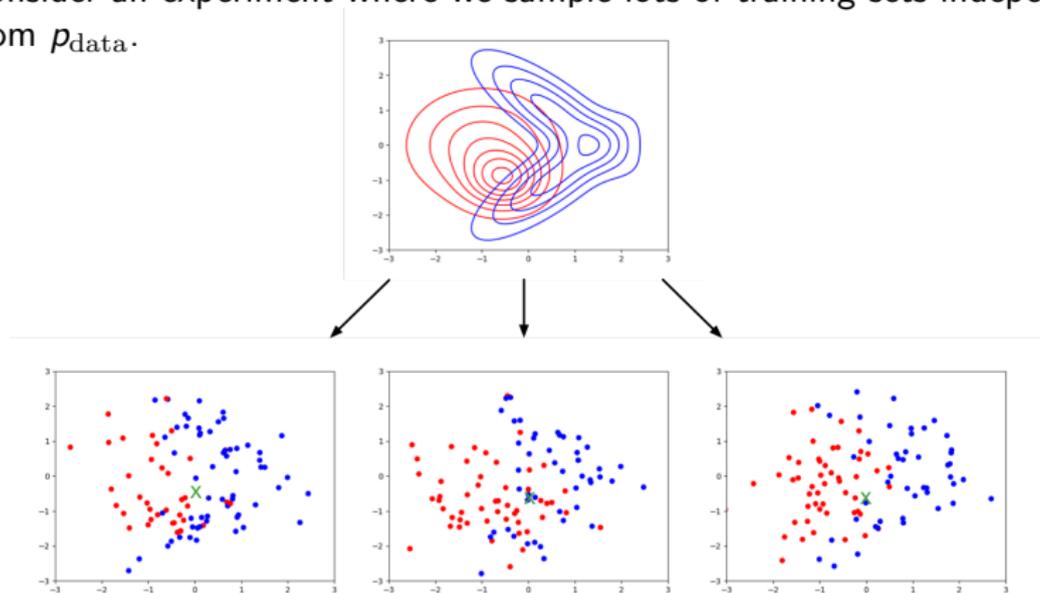
- Recall that overly simple models underfit the data, and overly complex models overfit.



- We can quantify this effect in terms of the [bias/variance decomposition](#).
 - Bias and variance of what?

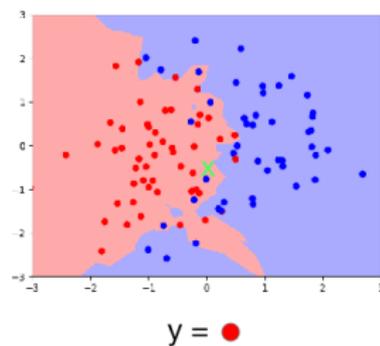
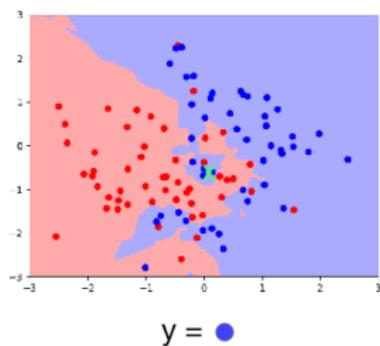
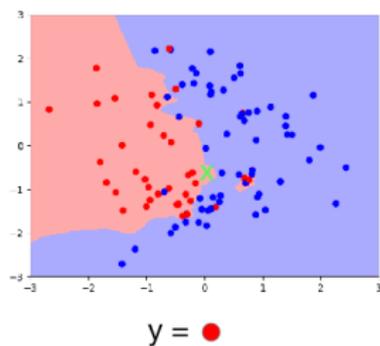
Bias-Variance Decomposition: Basic Setup

- Suppose the training set \mathcal{D} consists of pairs (\mathbf{x}_i, t_i) sampled **independent and identically distributed (i.i.d.)** from a single **data generating distribution** p_{data} .
- Pick a fixed query point \mathbf{x} (denoted with a green \times).
- Consider an experiment where we sample lots of training sets independently from p_{data} .



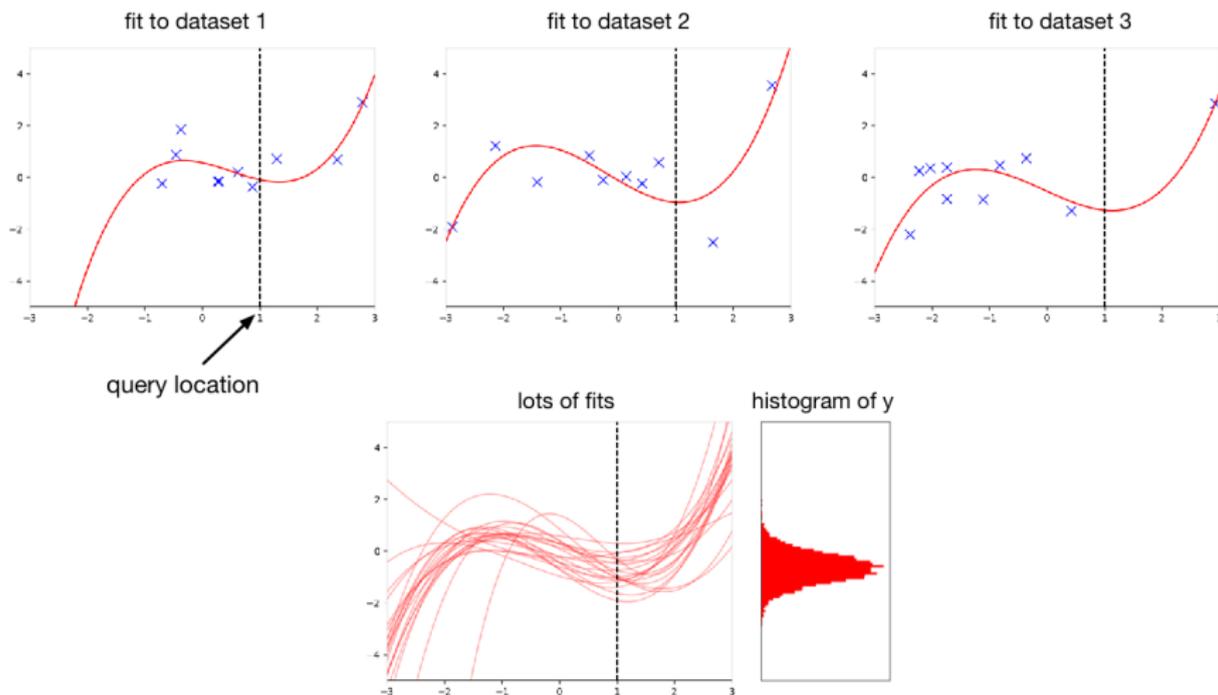
Bias-Variance Decomposition: Basic Setup

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



Bias-Variance Decomposition: Basic Setup

Here is the analogous setup for regression:



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

Bias-Variance Decomposition: Basic Setup

- Recap of basic setup:
 - ▶ Fix a query point \mathbf{x} .
 - ▶ Repeat:
 - ▶ Sample a random training dataset \mathcal{D} i.i.d. from the data generating distribution p_{data} .
 - ▶ Run the learning algorithm on \mathcal{D} to get a prediction y at \mathbf{x} .
 - ▶ Sample the (true) target from the conditional distribution $p(t|\mathbf{x})$.
 - ▶ Compute the loss $L(y, t)$.
- Notice: y is independent of t . (Why?)
- This gives a distribution over the loss at \mathbf{x} , with expectation $\mathbb{E}[L(y, t) | \mathbf{x}]$.
- For each query point \mathbf{x} , the expected loss is different. We are interested in minimizing the expectation of this with respect to $\mathbf{x} \sim p_{\text{data}}$.

Bayes Optimality

- For now, focus on squared error loss, $L(y, t) = \frac{1}{2}(y - t)^2$.
- A first step: suppose we knew the conditional distribution $p(t | \mathbf{x})$. What value y should we predict?
 - ▶ Here, we are treating t as a random variable and choosing y .
- **Claim:** $y_* = \mathbb{E}[t | \mathbf{x}]$ is the best possible prediction.
- **Proof:**

$$\begin{aligned}\mathbb{E}[(y - t)^2 | \mathbf{x}] &= \mathbb{E}[y^2 - 2yt + t^2 | \mathbf{x}] \\ &= y^2 - 2y\mathbb{E}[t | \mathbf{x}] + \mathbb{E}[t^2 | \mathbf{x}] \\ &= y^2 - 2y\mathbb{E}[t | \mathbf{x}] + \mathbb{E}[t | \mathbf{x}]^2 + \text{Var}[t | \mathbf{x}] \\ &= y^2 - 2yy_* + y_*^2 + \text{Var}[t | \mathbf{x}] \\ &= (y - y_*)^2 + \text{Var}[t | \mathbf{x}]\end{aligned}$$

$$\mathbb{E}[(y - t)^2 | \mathbf{x}] = (y - y_*)^2 + \text{Var}[t | \mathbf{x}]$$

- The first term is nonnegative, and can be made 0 by setting $y = y_*$.
- The second term corresponds to the inherent unpredictability, or **noise**, of the targets, and is called the **Bayes error**.
 - ▶ This is the best we can ever hope to do with any learning algorithm. An algorithm that achieves it is **Bayes optimal**.
 - ▶ Notice that this term doesn't depend on y .
- This process of choosing a single value y_* based on $p(t | \mathbf{x})$ is an example of **decision theory**.

- Now return to treating y as a random variable (where the randomness comes from the choice of dataset).
- We can decompose out the expected loss (suppressing the conditioning on \mathbf{x} for clarity):

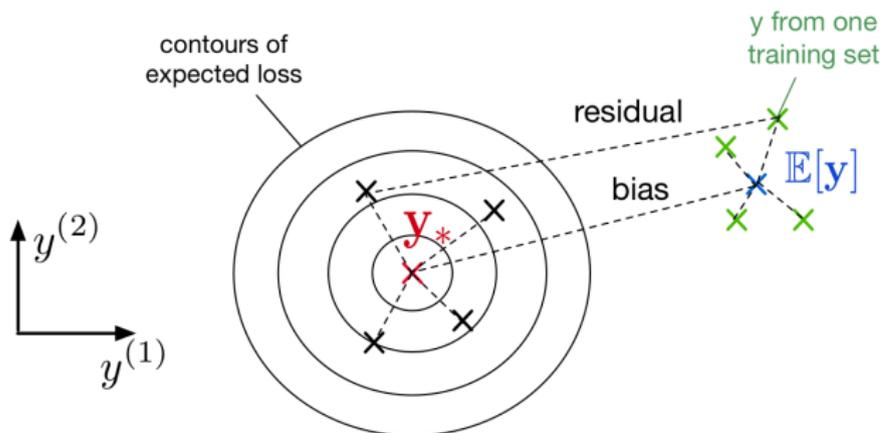
$$\begin{aligned}\mathbb{E}[(y - t)^2] &= \mathbb{E}[(y - y_*)^2] + \text{Var}(t) \\ &= \mathbb{E}[y_*^2 - 2y_*y + y^2] + \text{Var}(t) \\ &= y_*^2 - 2y_*\mathbb{E}[y] + \mathbb{E}[y^2] + \text{Var}(t) \\ &= y_*^2 - 2y_*\mathbb{E}[y] + \mathbb{E}[y^2] + \text{Var}(y) + \text{Var}(t) \\ &= \underbrace{(y_* - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\text{Var}(y)}_{\text{variance}} + \underbrace{\text{Var}(t)}_{\text{Bayes error}}\end{aligned}$$

$$\mathbb{E}[(y - t)^2] = \underbrace{(y_* - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\text{Var}(y)}_{\text{variance}} + \underbrace{\text{Var}(t)}_{\text{Bayes error}}$$

- 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”.

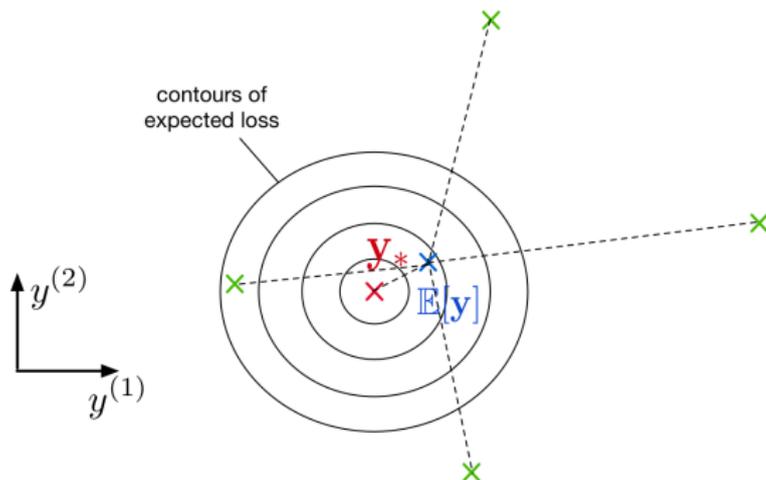
Bias/Variance Decomposition: Another Visualization

- We can visualize this decomposition in **output space**, where the axes correspond to predictions on the test examples.
- If we have an overly simple model (e.g. KNN with large k), it might have
 - ▶ high bias (because it's too simplistic to capture the structure in the data)
 - ▶ low variance (because there's enough data to get a stable estimate of the decision boundary)



Bias/Variance Decomposition: Another Visualization

- If you have an overly complex model (e.g. KNN with $k = 1$), it might have
 - ▶ low bias (since it learns all the relevant structure)
 - ▶ high variance (it fits the quirks of the data you happened to sample)



Now, back to bagging!

Bagging: Motivation

- Suppose we could somehow sample m independent training sets from $\mathcal{P}_{\text{data}}$.
- 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

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

- ▶ **Variance: reduced**, since we're averaging over independent samples

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

Bagging: The Idea

- In practice, running an algorithm separately on independently sampled datasets is very wasteful!
- Solution: **bootstrap aggregation**, or **bagging**.
 - ▶ Take a single dataset \mathcal{D} with n examples.
 - ▶ Generate m new datasets, 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!

Bagging: The Idea

- Problem: the datasets are not independent, so we don't get the $1/m$ variance reduction.
 - ▶ Possible to show that if the sampled predictions have variance σ^2 and correlation ρ , then

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

- Ironically, it can be advantageous to introduce *additional* variability into your algorithm, as long as it reduces the correlation between samples.
 - ▶ Intuition: you want to invest in a diversified portfolio, not just one stock.
 - ▶ Can help to use average over multiple algorithms, or multiple configurations of the same algorithm.

- **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

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.
 - ▶ There is still correlation between classifiers.
- Random forest solution: Add more randomness.