STA 314: Statistical Methods for Machine Learning I

Lecture 3 - Bias-Variance Decomposition

Chris J. Maddison

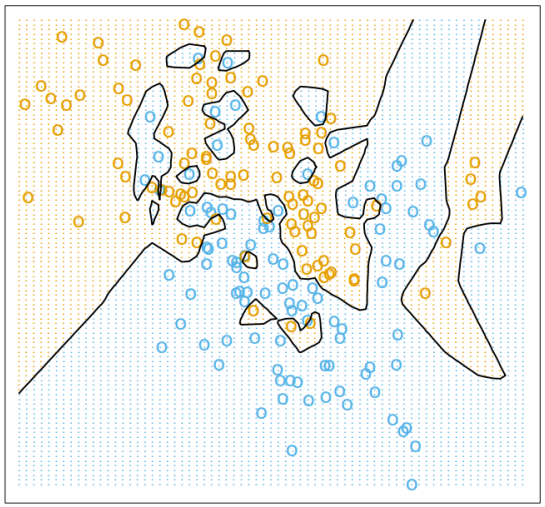
University of Toronto

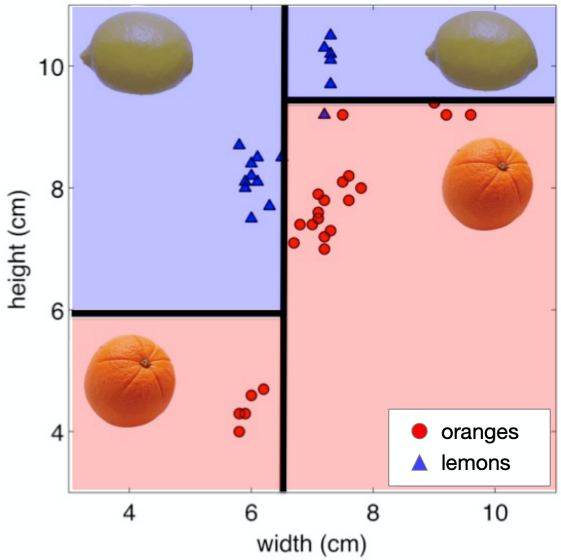
Today

- Today we will talk about the bias-variance decomposition, which is beginning to make more precise our discussion of overfitting and underfitting last class.

Recall: supervised learning

- In supervised learning, our learning algorithms (k-NN, decision trees) produce predictions $\hat{y}^\star(\mathbf{x}) \approx t$ for a query point $\mathbf{x}$.





Recall: supervised learning

- We can think of this as picking a predictor function $\hat{y}^\star \in \mathcal{H}$ from a hypothesis class by minimizing the average loss on the training set

$$\hat{y}^\star = \arg\min_{y \in \mathcal{H}} \hat{\mathcal{R}}[y, \mathcal{D}^{train}]$$

- Then, we measure the average loss on an unseen test set to approximate how well $\hat{y}^\star$ does on the true data generating distribution,

$$\hat{\mathcal{R}}[\hat{y}^\star, \mathcal{D}_{test}] \approx \mathcal{R}[\hat{y}^\star]$$

Recall: supervised learning

- This view of supervise learning is a very idealized view. We sometimes cannot fully optimize the loss.
  - Data is not typically i.i.d. according to a fixed data generating distribution.
  - We often select $\hat{y}^\star$ based on training loss, but sometimes we cannot find the global optimal $\hat{y}^\star$, e.g., decision trees.
- Still, it's a very useful general model for supervised learning.

Code Notebook

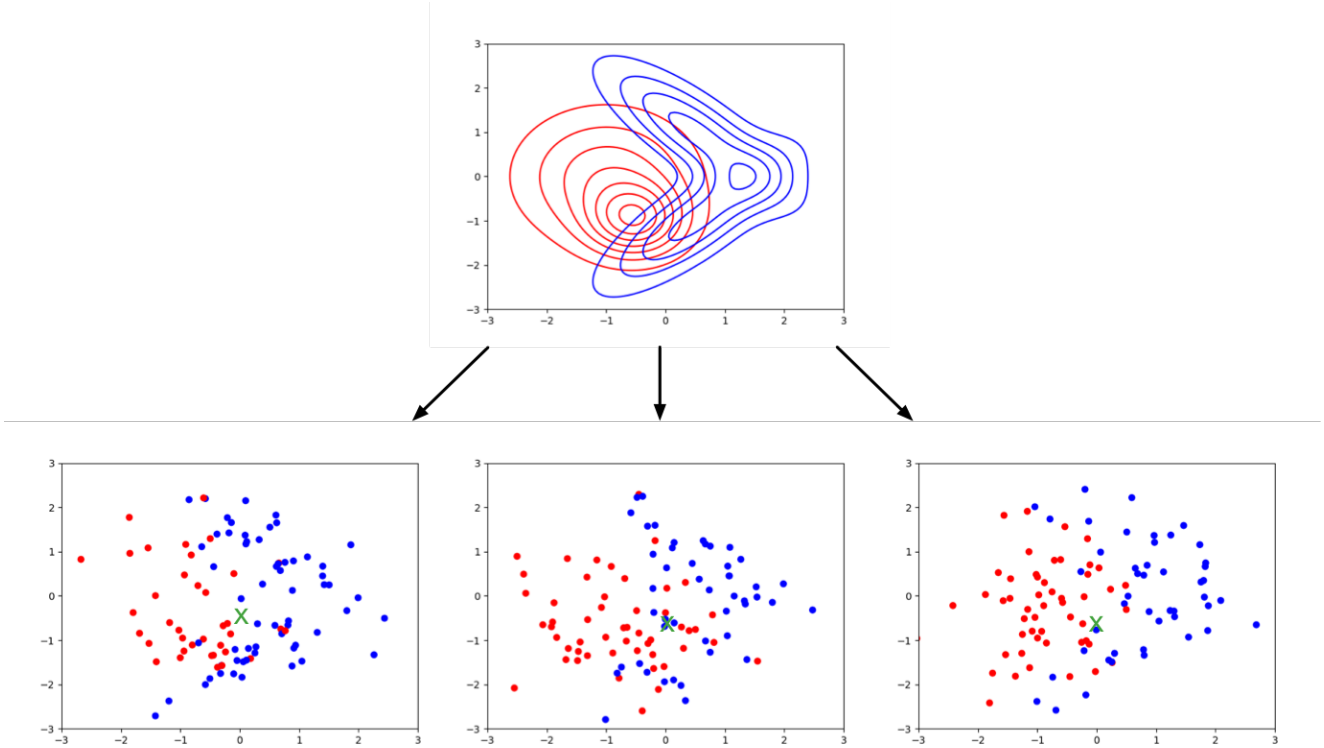
- I've made a code notebook to help these concepts stick.

Bias-Variance Decomposition

- The predictor $\hat{y}^\star$ that we fit on the training set is random and so is its expected loss $\hat{\mathcal{R}}[\hat{y}^\star]$.
- Now we will study the performance of our procedure in terms of the performance we expect, $\mathbb{E}[\hat{\mathcal{R}}[\hat{y}^\star]]$, averaging over the randomness of the training set.
- Specifically, we will decompose $\mathbb{E}[\hat{\mathcal{R}}[\hat{y}^\star]]$ into terms that allow us to understand the effect of a hypothesis class on our performance. This is called the bias-variance decomposition.

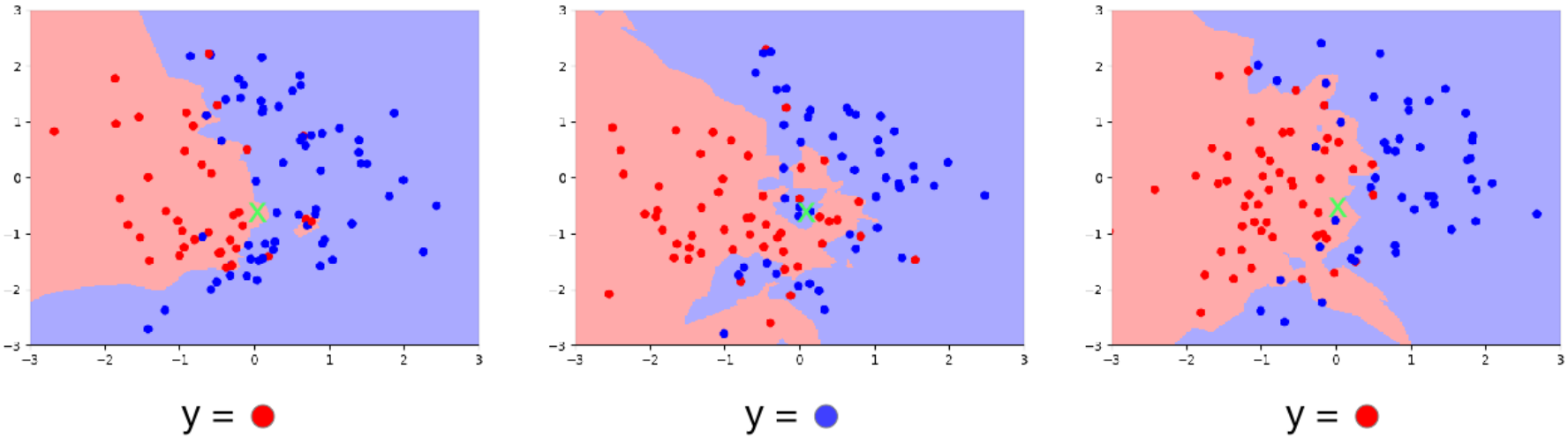
Bias-Variance Decomposition: Basic Setup

- Recall: the training set $\mathcal{D}^{train} = \{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^{N}$ contains N i.i.d. draws from a single data generating distribution p_{data}.
- Consider a fixed query point $\mathbf{x}$ (green $\mathbf{x}$ below).
- Consider sampling many training sets $\mathcal{D}_n^{train}$ independently from p_{data}.



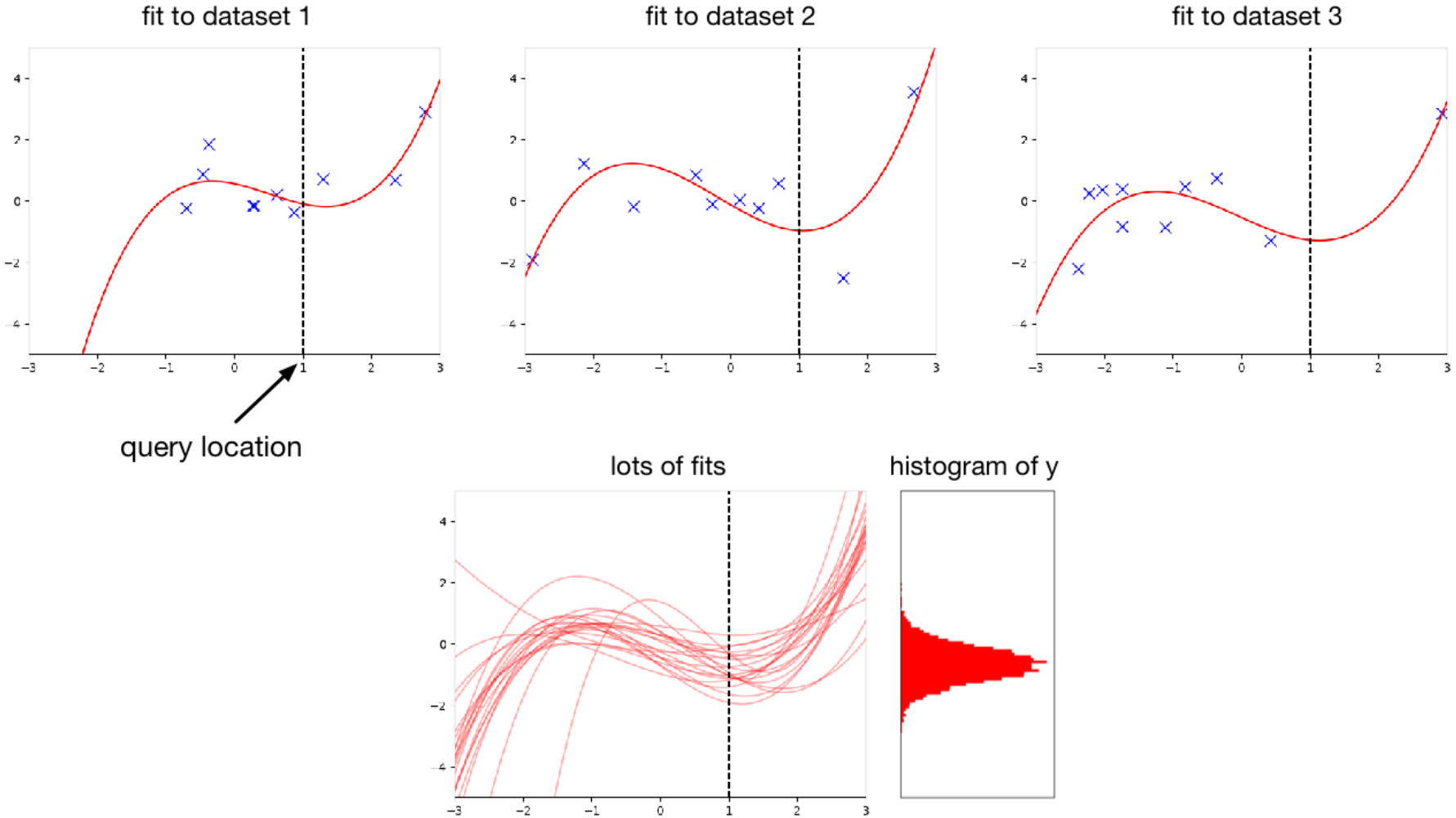
Bias-Variance Decomposition: Basic Setup

- For each training set $\mathcal{D}_n^{train}$, run learning alg. to get a predictor $\hat{y}_n^\star \in \mathcal{H}$.
- Compute the prediction $\hat{y}_n^\star(\mathbf{x})$ and compare it to a label t drawn from $p_{\text{data}}(t|\mathbf{x})$.
- We can view $\hat{y}_n^\star$ as a random variable, where the randomness comes from the choice of training set.



Bias-Variance Decomposition: Basic Setup

Here is the analogous setup for regression:



Bias-Variance Decomposition: Basic Setup

- Imagine now this process:
  - Fix a query point $\mathbf{x}$.
  - Sample the (true) target t from the conditional distribution $p_{\text{data}}(t|\mathbf{x})$.
  - Repeat:
    - Sample a random training dataset $\mathcal{D}_n^{train}$ i.i.d. from the data generating distribution p_{data}.
    - Run the learning algorithm on $\mathcal{D}_n^{train}$ to get a prediction $\hat{y}_n^\star(\mathbf{x})$ from $\mathcal{H}$ at $\mathbf{x}$.
    - Compute the loss $L(\hat{y}_n^\star(\mathbf{x}), t)$.
  - Average the losses.
- This gives a distribution over the loss at $\mathbf{x}$, with expectation $\mathbb{E}[L(\hat{y}^\star(\mathbf{x}), t) \,|\, \mathbf{x}]$ taken over t and the *random* training set $\mathcal{D}^{train}$ where $\hat{y}^\star = \arg\min_{y \in \mathcal{H}} \hat{\mathcal{R}}[y, \mathcal{D}^{train}]$.
- If we take an expectation over $\mathbf{x}$, then we get $\mathbb{E}[\mathbb{E}[L(\hat{y}^\star(\mathbf{x}), t) \,|\, \mathbf{x}]] = \mathbb{E}[\mathcal{R}[\hat{y}^\star]]$.

Bayes Optimality

- I hope that built up an intuition. Now we will work towards the decomposition we promised.
- For now, focus on squared error loss, $L(y, t) = \frac{1}{2}(y - t)^2$ with $y, t \in \mathbb{R}$.
- A first step: suppose we knew the conditional distribution $p_{\text{data}}(t \mid \mathbf{x})$. What is the best deterministic value $y(\mathbf{x}) \in \mathbb{R}$ should we predict?
  - Here, we are treating t as a random variable and choosing $y(\mathbf{x})$.
- **Claim:** $y^\star(\mathbf{x}) = \mathbb{E}[t \mid \mathbf{x}]$ is the best possible prediction.

Bayes Optimality

Proof: Consider a fixed $y \in \mathbb{R}$. First, expand the square

$$\mathbb{E}[(y - t)^2 \mid \mathbf{x}] = \mathbb{E}[y^2 - 2yt + t^2 \mid \mathbf{x}]$$

then distribute expectation

$$= y^2 - 2y\mathbb{E}[t \mid \mathbf{x}] + \mathbb{E}[t^2 \mid \mathbf{x}]$$

then apply a variance identity

$$= y^2 - 2y\mathbb{E}[t \mid \mathbf{x}] + \mathbb{E}[t \mid \mathbf{x}]^2 + \mathrm{Var}[t \mid \mathbf{x}]$$

then apply the definition of $y^\star(\mathbf{x})$

$$= y^2 - 2yy^\star(\mathbf{x}) + y^\star(\mathbf{x})^2 + \mathrm{Var}[t \mid \mathbf{x}]$$

and collect terms

$$= (y - y^\star(\mathbf{x}))^2 + \mathrm{Var}[t \mid \mathbf{x}]$$

Bayes Optimality

Proof (continued): We've shown

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

The second term doesn't depend on y and the first term is smallest when $y = y^\star(\mathbf{x})$. This concludes our proof.

- 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 $y^\star(\mathbf{x})$ based on $p_{\text{data}}(t \mid \mathbf{x})$ is an example of decision theory.

Bayes Optimality

- But, in practice, our prediction $\hat{y}^\star(\mathbf{x})$ is not $y^\star(\mathbf{x})$. Instead, it is a random variable (where the randomness comes from randomness of the training set).
- Key fact: $\hat{y}^\star$ is independent of t given $\mathbf{x}$.
- We are going to show that the expected loss (over $\mathbf{x}$, t and $\hat{y}^\star$) of our trained predictor decomposes into three terms.

$$\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2]$$

$$= \mathrm{E}\left[\underbrace{(y^\star(\mathbf{x}) - \mathbb{E}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}])^2}_{\text{bias}} + \underbrace{\mathrm{Var}(\hat{y}^\star(\mathbf{x}) \mid \mathbf{x})}_{\text{variance}} + \underbrace{\mathrm{Var}(t \mid \mathbf{x})}_{\text{Bayes error}}\right]$$

- Intuition:
  - bias: how wrong the expected prediction is
  - variance: the amount of variability in the predictions
  - Bayes error: the inherent unpredictability of the targets

Bayes Optimality

- Let's prove the decomposition.
- To do this, we'll use the tower property of expectation, twice.

$$\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2] = \mathbb{E}[\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2 \,|\, \mathbf{x}]]$$

$$\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2 \,|\, \mathbf{x}] = \mathbb{E}[\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2 \,|\, \mathbf{x}, \hat{y}^\star(\mathbf{x})] \,|\, \mathbf{x}]$$

Bayes Optimality

Let's start with the inner term. We can use our previous result (because we are conditioning on $\hat{y}^\star(\mathbf{x})$, so we can treat it like a constant).

$$\begin{aligned}&\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2 \mid \mathbf{x}, \hat{y}^\star(\mathbf{x})] \\ &= (\hat{y}^\star(\mathbf{x}) - y^\star(\mathbf{x}))^2 + \mathrm{Var}[t \mid \mathbf{x}, \hat{y}^\star(\mathbf{x})]\end{aligned}$$

then expand the square

$$= \hat{y}^\star(\mathbf{x})^2 - 2\hat{y}^\star(\mathbf{x})y^\star(\mathbf{x}) + y^\star(\mathbf{x})^2 + \mathrm{Var}[t \mid \mathbf{x}, \hat{y}^\star(\mathbf{x})]$$

and since $\hat{y}^\star(\mathbf{x})$ is independent of t given $\mathbf{x}$, we drop the conditioning in the variance term

$$= \hat{y}^\star(\mathbf{x})^2 - 2\hat{y}^\star(\mathbf{x})y^\star(\mathbf{x}) + y^\star(\mathbf{x})^2 + \mathrm{Var}[t \mid \mathbf{x}].$$

Bayes Optimality

Now we will "integrate out" $\hat{y}^\star(\mathbf{x})$. Using our result from the previous slide, first we expand the square,

$$\mathbb{E}[\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2 \mid \mathbf{x}, \hat{y}^\star(\mathbf{x})] \mid \mathbf{x}]$$
$$= \mathbb{E}[\hat{y}^\star(\mathbf{x})^2 - 2\hat{y}^\star(\mathbf{x})y^\star(\mathbf{x}) + y^\star(\mathbf{x})^2 + \mathrm{Var}[t \mid \mathbf{x}] \mid \mathbf{x}]$$

then distribute expectation

$$= \mathbb{E}[\hat{y}^\star(\mathbf{x})^2 \mid \mathbf{x}] - 2\,\mathbb{E}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}]y^\star(\mathbf{x}) + y^\star(\mathbf{x})^2 + \mathrm{Var}[t \mid \mathbf{x}]$$

then apply a variance identity

$$= \mathbb{E}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}]^2 + \mathrm{Var}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}] - 2\,\mathbb{E}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}]y^\star(\mathbf{x}) + y^\star(\mathbf{x})^2 + \mathrm{Var}[t \mid \mathbf{x}]$$

and collect terms

$$= (\mathbb{E}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}] - y^\star(\mathbf{x}))^2 + \mathrm{Var}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}] + \mathrm{Var}[t \mid \mathbf{x}]$$

Bayes Optimality

Recap: we've just shown:

$$\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2 \mid \mathbf{x}] =$$
$$= \underbrace{(y^\star(\mathbf{x}) - \mathbb{E}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}])^2}_{\text{bias}} + \underbrace{\text{Var}(\hat{y}^\star(\mathbf{x}) \mid \mathbf{x})}_{\text{variance}} + \underbrace{\text{Var}(t \mid \mathbf{x})}_{\text{Bayes error}}$$

Applying the tower property of expectation again, we get

$$\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2] = \mathbb{E}[\mathbb{E}[(\hat{y}^\star(\mathbf{x}) - t)^2 \mid \mathbf{x}]]$$
$$= \text{E}\left[\underbrace{(y^\star(\mathbf{x}) - \mathbb{E}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}])^2}_{\text{bias}} + \underbrace{\text{Var}(\hat{y}^\star(\mathbf{x}) \mid \mathbf{x})}_{\text{variance}} + \underbrace{\text{Var}(t \mid \mathbf{x})}_{\text{Bayes error}}\right]$$

Bayes Optimality

- Let's step back and consider what we just did. First, recall:
  - Picking a predictor by minimizing the average loss on the training set

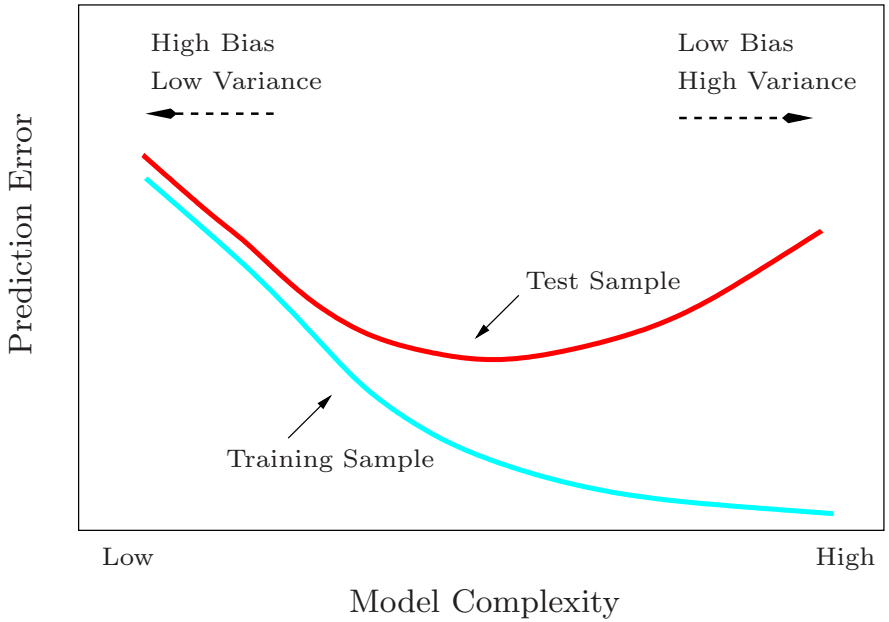
$$\hat{y}^\star = \arg\min_{y \in \mathcal{H}} \hat{\mathcal{R}}[y, \mathcal{D}^{train}]$$

    returns a random predictor $\hat{y}^\star$.
  - We're interested in our performance in terms of expected loss $\mathcal{R}[\hat{y}^\star]$, which is random (due to randomness of the $\hat{y}^\star$).
- So, to summarize our performance on average, we want to study $\mathbb{E}[\mathcal{R}[\hat{y}^\star]]$. We've shown:

$$\mathrm{E}[\mathcal{R}[\hat{y}^\star]] = \mathrm{E}\left[\underbrace{(y^\star(\mathbf{x}) - \mathbb{E}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}])^2}_{\text{bias}} + \underbrace{\mathrm{Var}[\hat{y}^\star(\mathbf{x}) \mid \mathbf{x}]}_{\text{variance}} + \underbrace{\mathrm{Var}[t \mid \mathbf{x}]}_{\text{Bayes error}}\right]$$

- How does our choice of $\mathcal{H}$ interact with this analysis?

Bayes Optimality

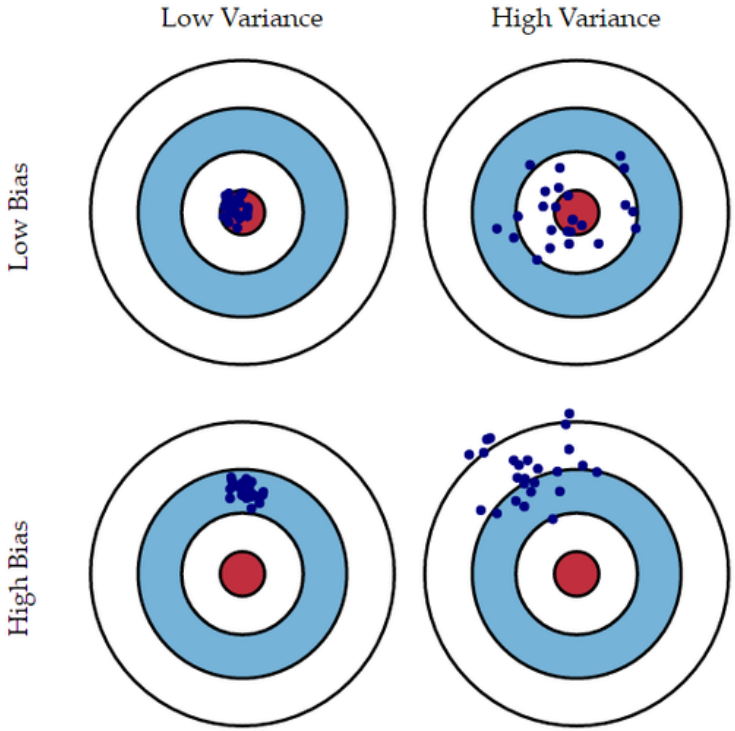


Source: ESL

- If $\mathcal{H}$ is large, then $\hat{y}^\star$ can get close $y^\star$, therefore reducing bias. It's also sensitive to the finite training set, therefore increasing variance.
- If $\mathcal{H}$ is small, then $\hat{y}^\star$ is typically from $y^\star$, therefore increasing bias. It's less sensitive to the finite training set, therefore reducing variance.
- Even though this analysis only applies to squared error, we often loosely use "bias" and "variance" as synonyms for "underfitting" and "overfitting".

Bias and Variance

- Throwing darts = predictions for each draw of a dataset

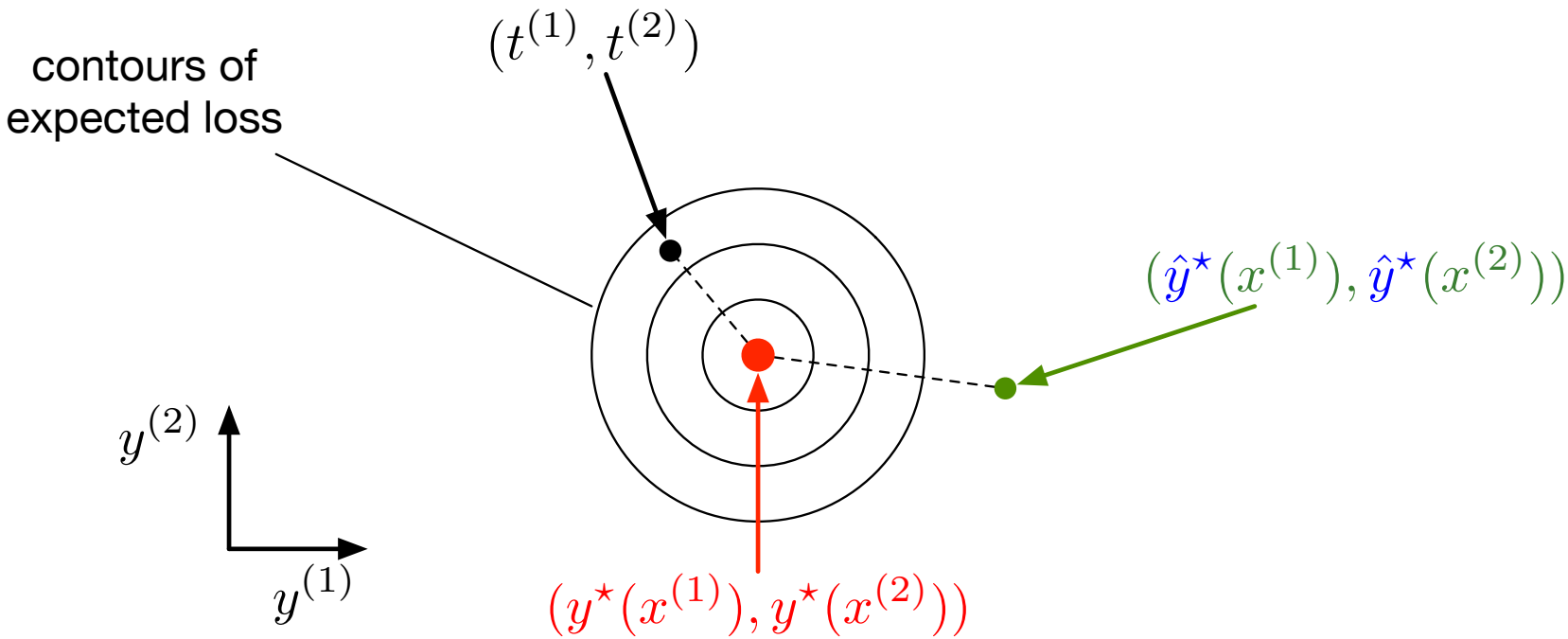


Source: ESL.

- Be careful, the expected loss averages over points $\mathbf{x}$ from the data distribution, so this produces its own type of variance.

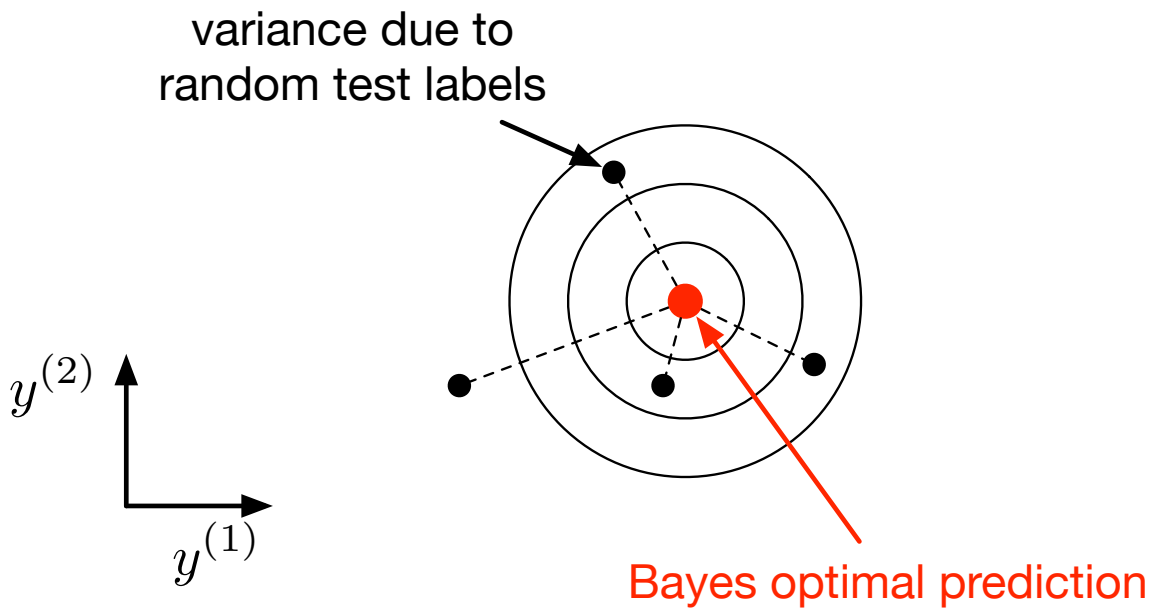
Bias/Variance Decomposition: Another Visualization

- In practice, measure the average loss $\hat{\mathcal{R}}[\hat{y}^\star, \mathcal{D}_{test}]$ on the test set instead of $\mathcal{R}[\hat{y}^\star]$.
- Let's visualize the bias-variance decomposition by plotting the space of predictions of the model, where each axis correspond to predictions on a two test examples $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$.



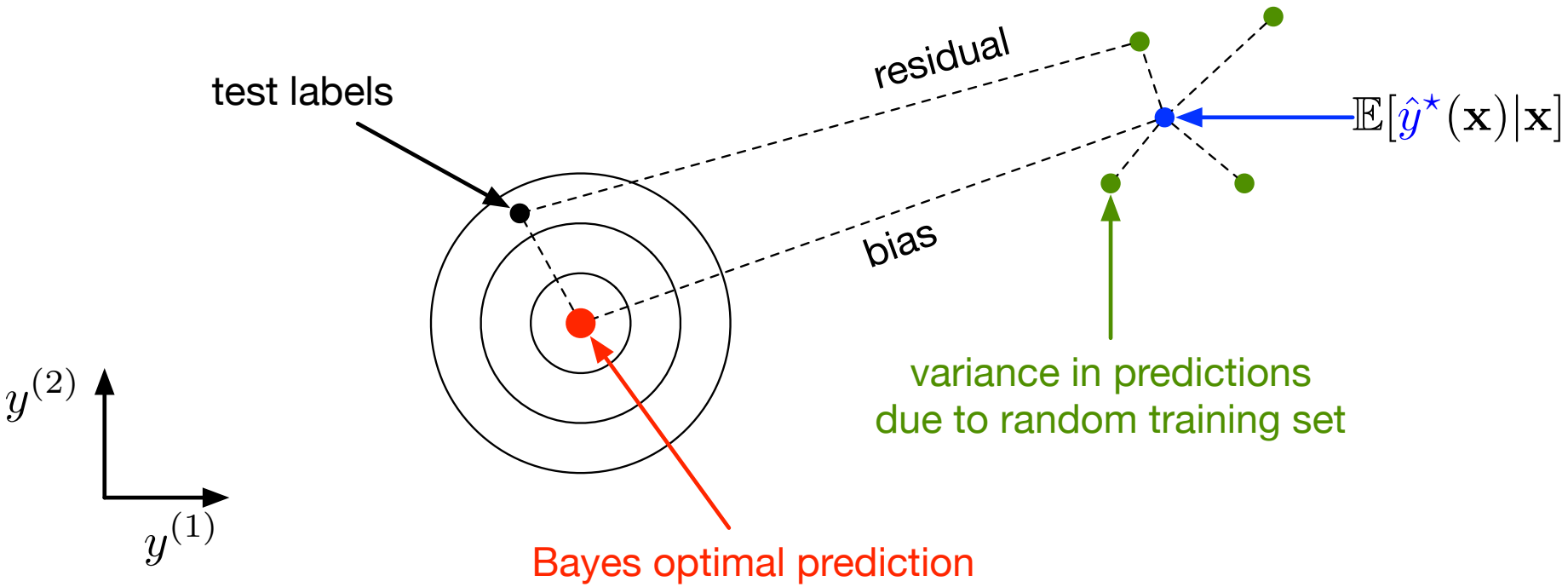
Bias/Variance Decomposition: Another Visualization

- The Bayes error is an irreducible error that comes from the randomness in $p_{\text{data}}(t \mid \mathbf{x})$.



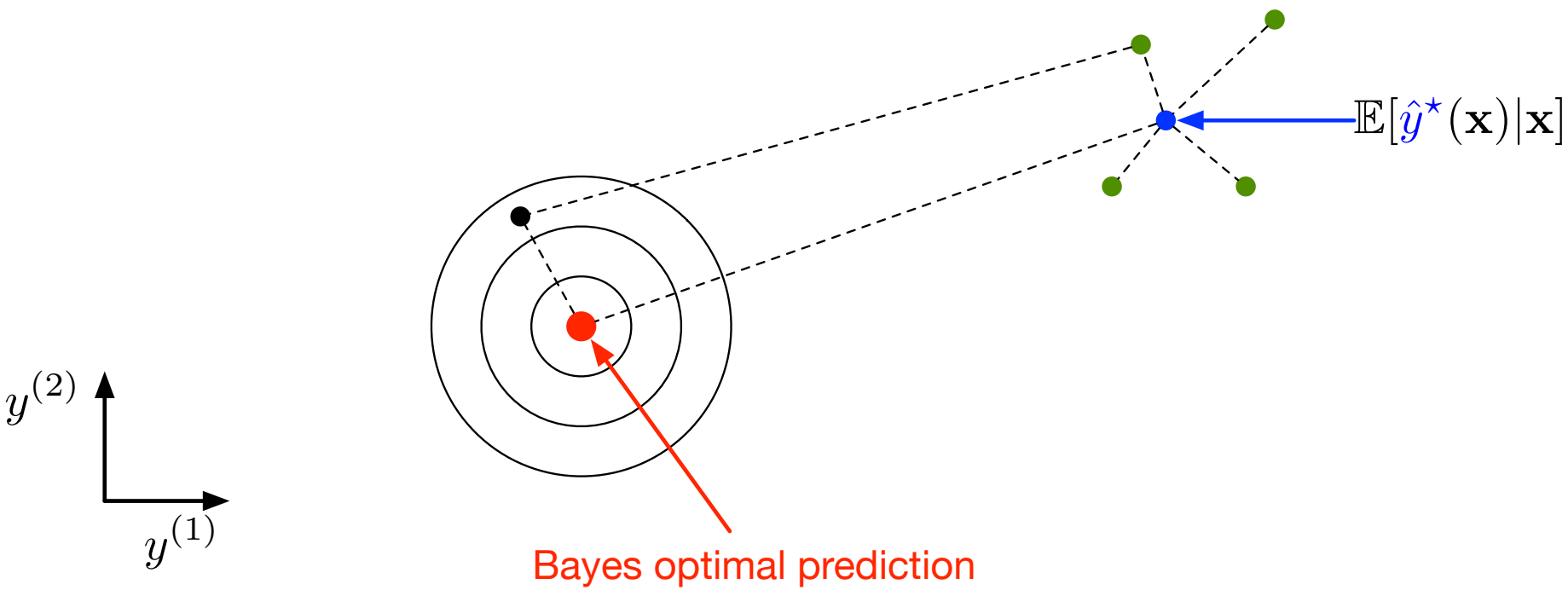
Bias/Variance Decomposition: Another Visualization

- Selecting a predictor $\hat{y}^\star \in \mathcal{H}$ from a training set comes with bias and variance.



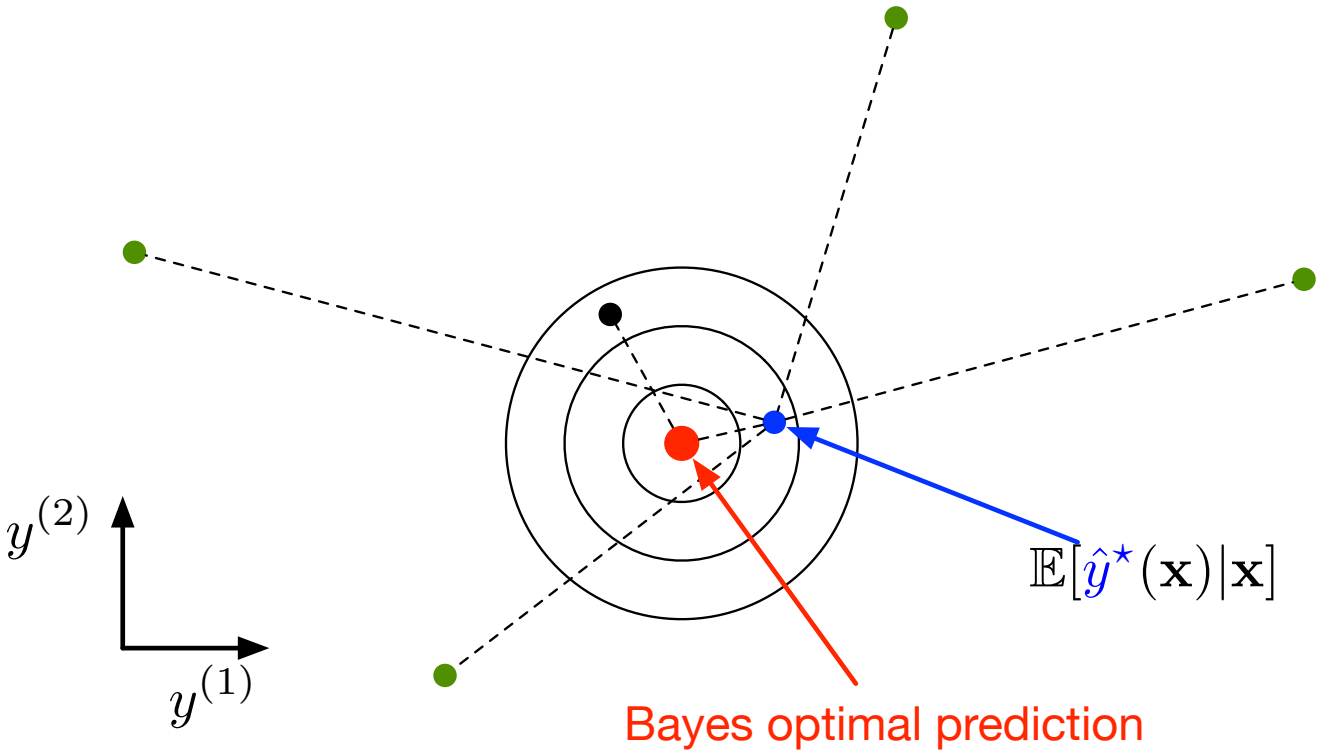
Bias/Variance Decomposition: Another Visualization

- An overly simple model (e.g. k-NN with large k) might have
  - high bias (too simplistic to capture the structure in the data)
  - low variance (there's enough data to get a stable estimate of the decision boundary)



Bias/Variance Decomposition: Another Visualization

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



Validation

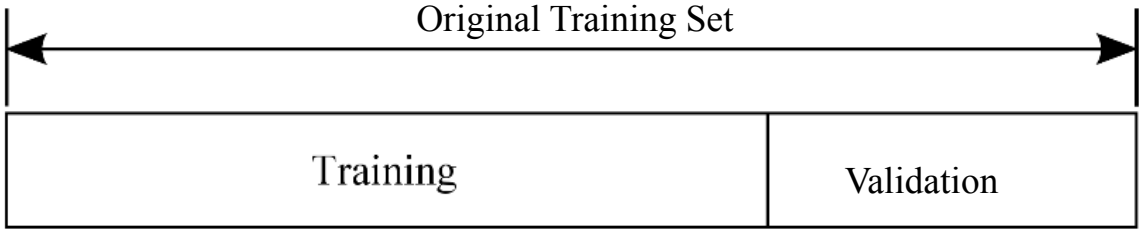
- Before we move on to bagging, it's a good time to mention validation.
- We may want to assess how likely a learning algorithm is to generalize before picking one and reporting the final test error.
- In other words, until now we've been picking predictors that optimize the training loss, but we want a technique for picking predictors that are likely to generalize as well.

Validation

- For example, we may want to assess the following types of choices:
  1. Hyper-parameters of the learning algorithm that lead to better generalization. Often there are parameters that cannot be fit on the training set, e.g., k in k-NN, because the training set would give meaningless answers about the best setting, i.e., $k = 1$ is always gives optimal training set loss for k-NN.
  2. Picking predictors that generalize better. E.g., should we use a decision tree or k-NN if we want to generalize?
- We make these choices using validation to avoid measuring test loss (then the test set would no longer be unseen data!).
- Suppose we are trying to estimate the generalization of two learning algorithms, e.g., a decision tree and a k-NN model.

Hold-out validation

- The most common method of validation is to hold-out a subset of the training set and use it to assess how likely we are to generalize to unseen data.



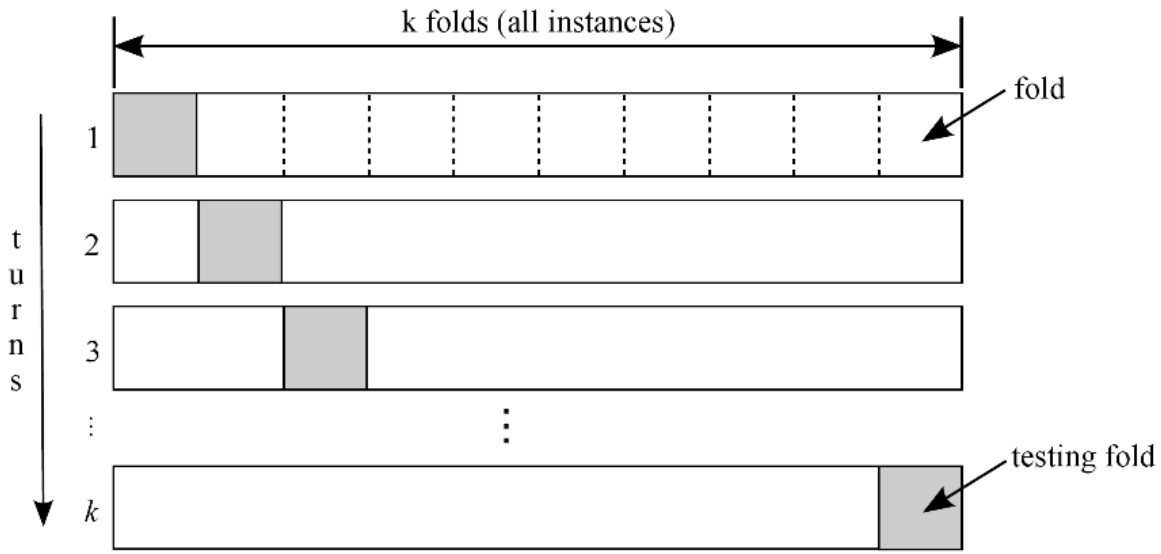
- In our example of deciding between a decision tree and k-NN in terms of generalization, we would fit $\hat{y}^{\star}_{k\text{NN}}$ and $\hat{y}^{\star}_{\text{d-tree}}$ on the training set and measure the average loss on the validation set

$$\hat{\mathcal{R}}[\hat{y}^{\star}_{k\text{NN}}, \mathcal{D}^{valid}] \text{ vs. } \hat{\mathcal{R}}[\hat{y}^{\star}_{\text{d-tree}}, \mathcal{D}^{valid}]$$

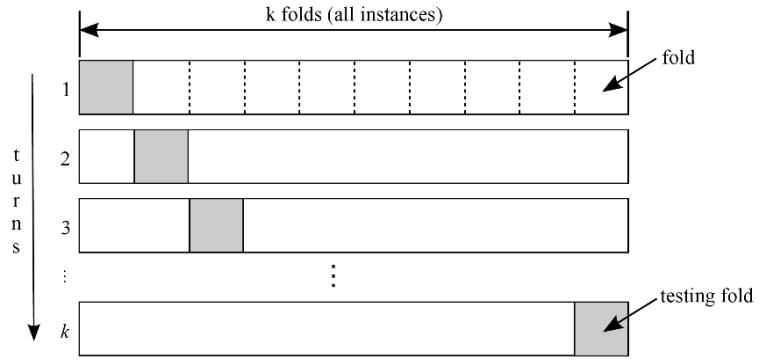
- We pick the predictor $\hat{y}^{\star}_{k\text{NN}}$ vs. $\hat{y}^{\star}_{\text{d-tree}}$ with lowest validation loss.
- Problem: this is usually a waste of data.

K-fold cross validation

- Second most common way: partition training data randomly into K equally sized subsets. For each "turn", use the first $K-1$ subsets (or "folds") as training data and the last subset as validation



K-fold cross validation



- In our running example: fit a new predictor using each learning algorithm on $K-1$ folds for each of the K turns, and measure the validation loss on the held-out fold, averaged over the turns:

$$\frac{1}{K}\sum_{i=1}^{K}\hat{\mathcal{R}}[\hat{y}^{\star}_{k\text{NN},i}, \mathcal{D}_i^{valid}] \text{ vs. } \frac{1}{K}\sum_{i=1}^{K}\hat{\mathcal{R}}[\hat{y}^{\star}_{d-tree,i}, \mathcal{D}_i^{valid}]$$

  where $\hat{y}^{\star}_{A,i}$ is the predictor fit on the training subset of the ith turn using algorithm A and $\mathcal{D}_i^{valid}$ is the validation subset of the ith turn.

- We pick the learning algorithm, e.g., k-NN v. decision tree, with lowest validation loss averaged across the K turns.