기계학습 (2022년도 2학기)

Ensemble I

전북대학교 컴퓨터공학부

Overview

- We've seen two particular learning algorithms: k-NN and decision trees
- Next two lectures: **combine multiple models into an ensemble** which performs better than the individual members
  - A key idea in ML more broadly
  - Generic class of techniques that can be applied to almost any learning algorithms...
  - ... 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 predict new examples
  - E.g., (possibly weighted) majority vote
- Intuitions:
  - Individuals often make mistakes, but the "majority" is less likely to make mistakes.
  - Individuals often have partial knowledge, but a committee can pool expertise to make better decisions.
- For this to be nontrivial, the learned hypotheses must differ somehow, e.g.
  - Different algorithm
  - Different choices of hyperparameters
  - Trained on different data
  - Trained with different weighting of the training examples
- Ensembles are usually easy to implement. The hard part is deciding what kind of ensemble you want, based on your goals.

Agenda

- 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

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

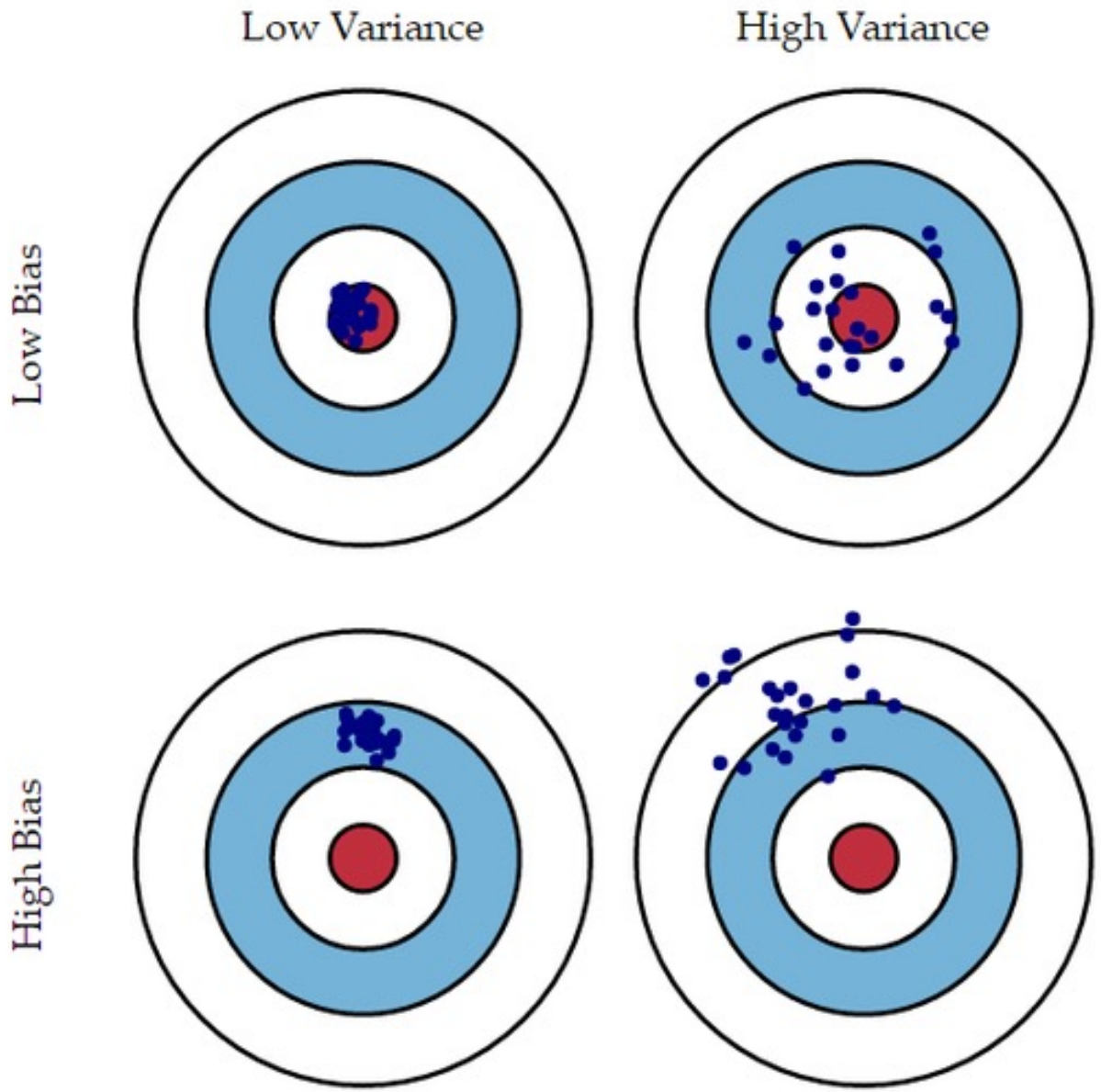
  - Averaging the 0-1 loss over the training set gives the **training error rate**, and averaging over the test set gives the **test error rate**.

- Example: **squared error loss** for regression

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

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

Bias and Variance



Training set의 변화에 따라 기계학습 알고리즘의 성능이 어떻게 달라지는지를 정량적으로 평가하기 위한 개념

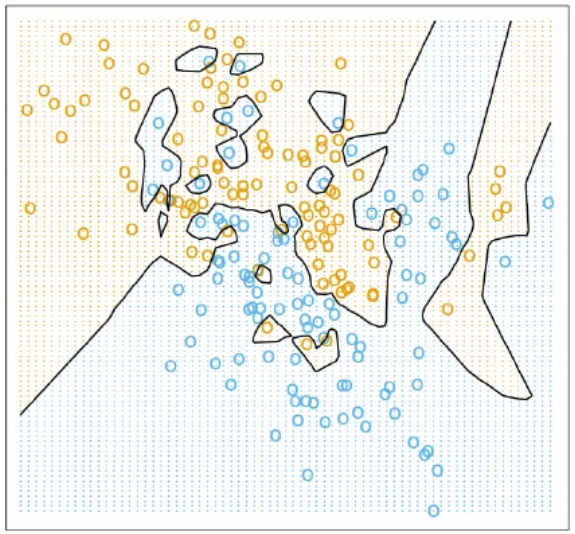
주어진 test 입력 하나에 대해

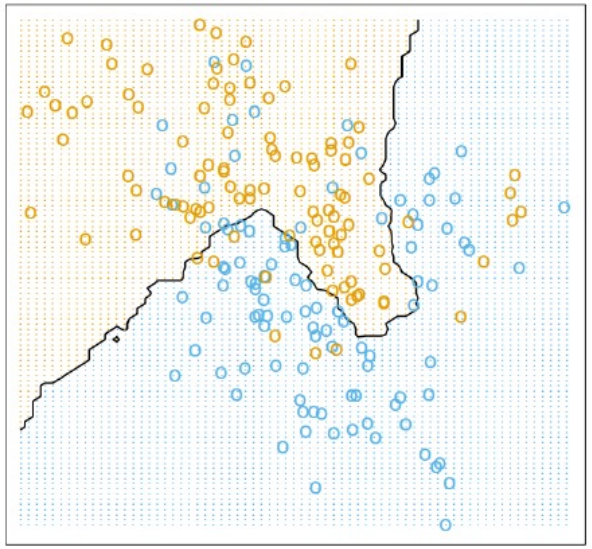
Bias: (unknown) true function과 hypothesis (또는 model) h(x)들의 예측의 평균

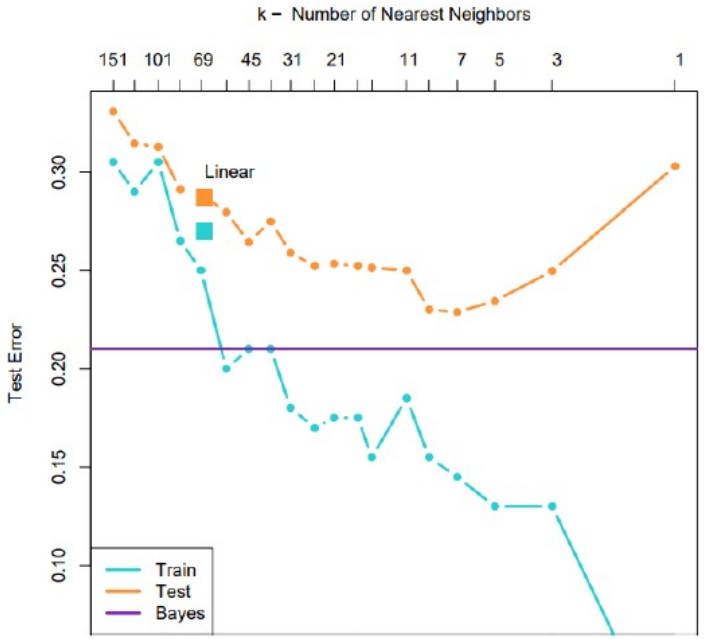
Variance: hypothesis가 training set에 따라 동일한 입력 x에 대해 변화하는 정도 (즉 training set이 달라지면서 x에 대한 예측값의 변화량)

Bias-Variance Decomposition

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



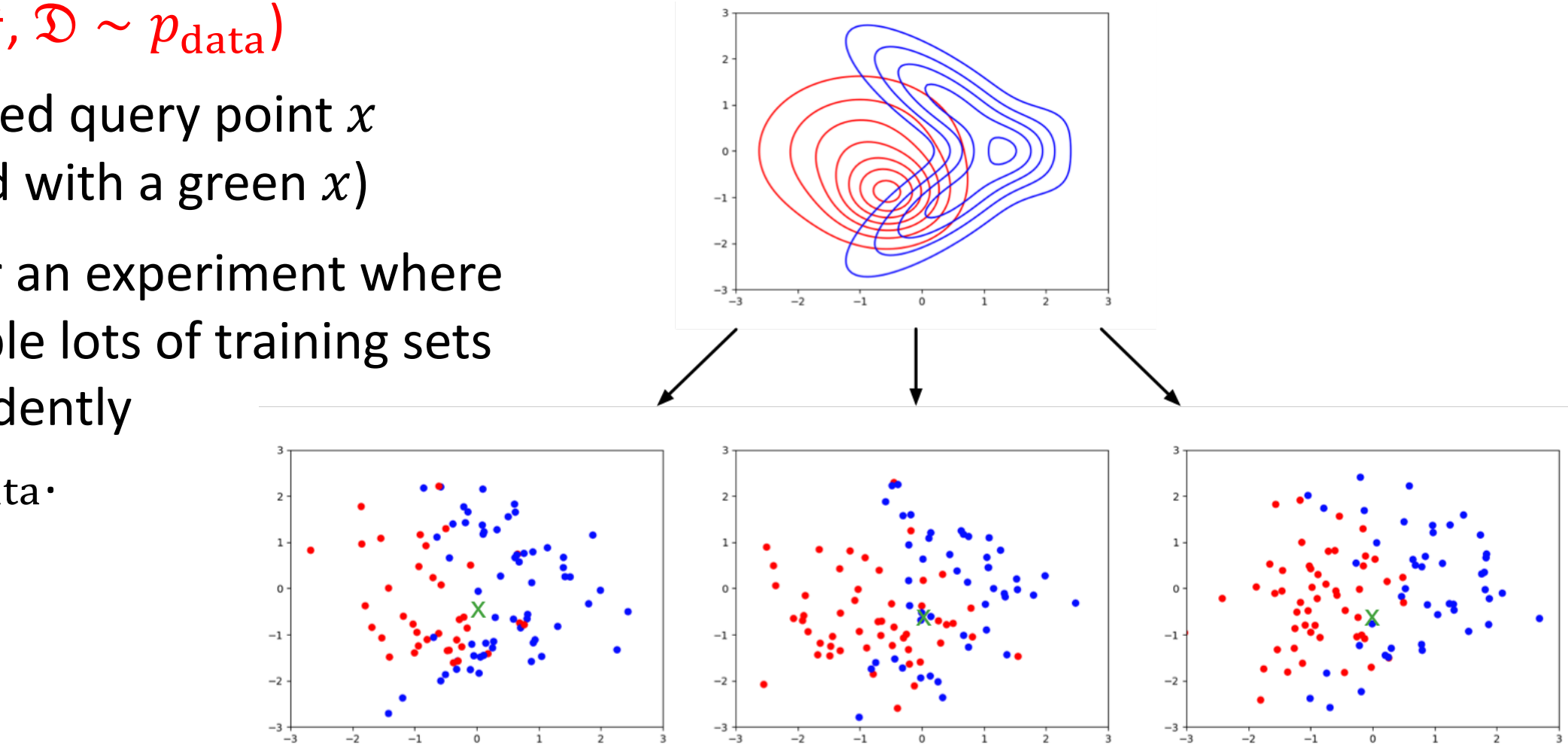




- We can quantify this effect in terms of the **bias/variance decomposition**.
- Bias and variance of what?

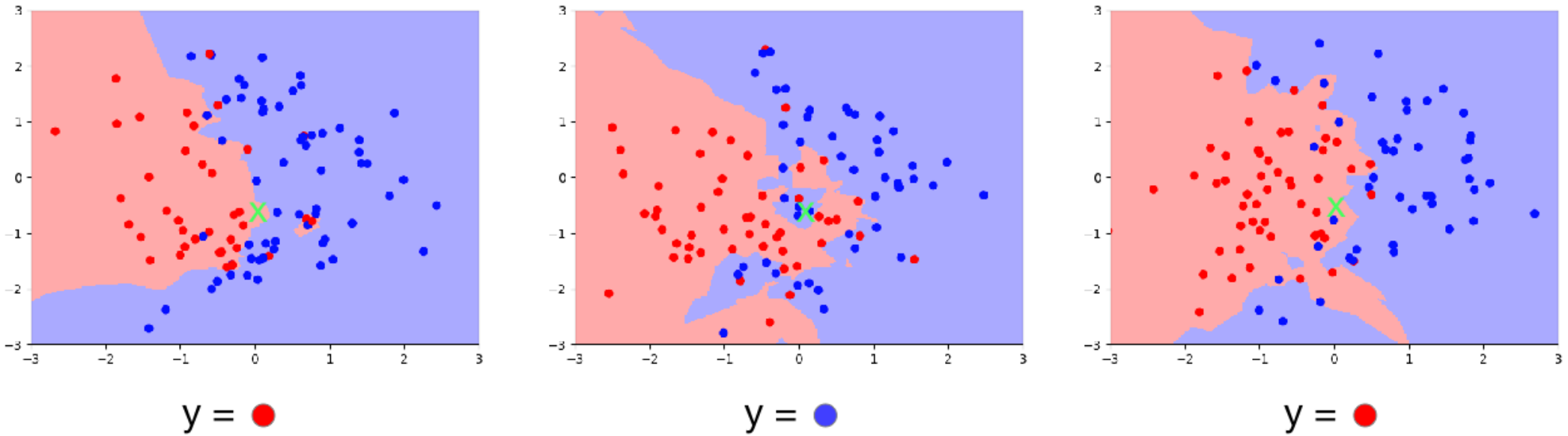
Bias-Variance Decomposition: Basic Setup

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



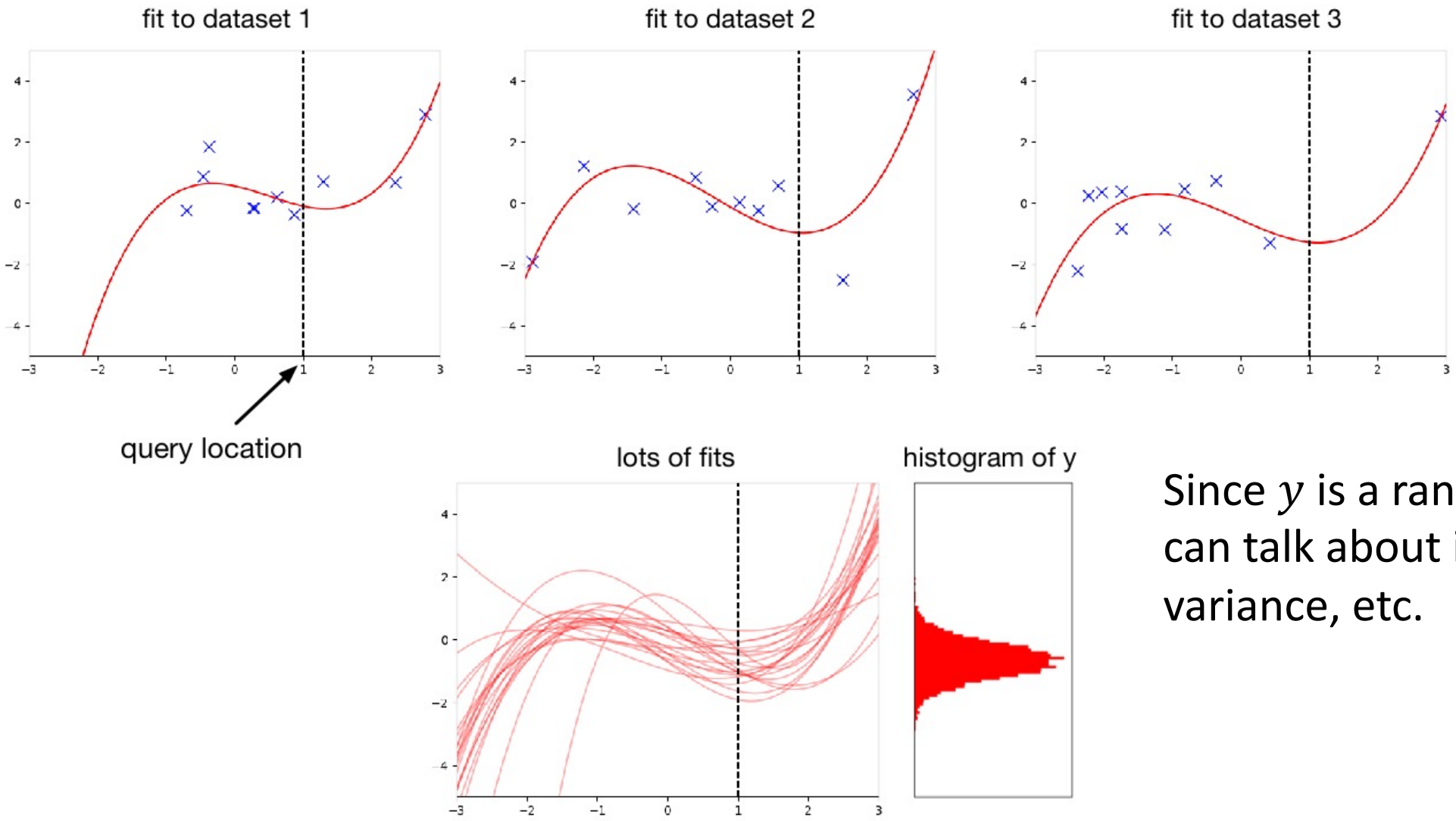
Bias-Variance Decomposition: Basic Setup

- Let's run our learning algorithm on each training set, and compute its prediction y at the query point $\mathbf{x}$.
- 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

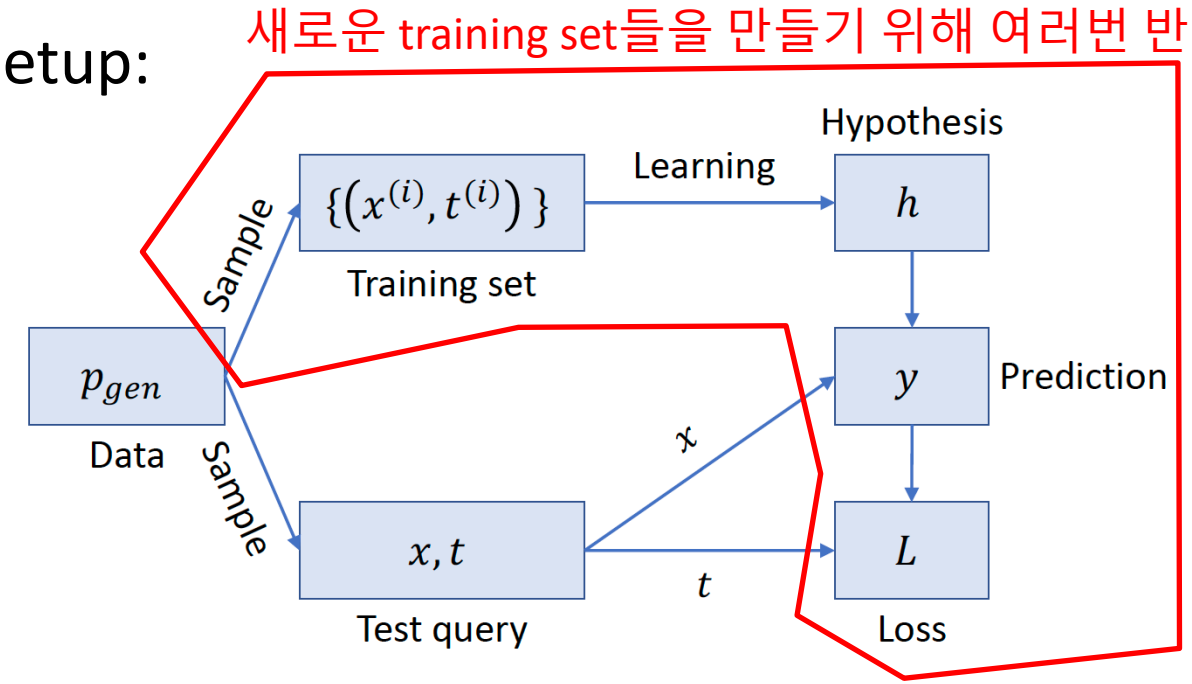
- Here is the analogous setup for regression:



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

Bias-Variance Decomposition: Basic Setup

- Recap of basic setup:



- Notice: y is independent of t. (why? y는 training set에서 훈련된 h로부터 예측)
- This gives a distribution over the loss at $\mathbf{x}$, with expectation $\mathbb{E}[L(y, t) \mid \mathbf{x}]$
  (독립적으로 sampling된 training set들로부터 개별적으로 훈련된 h들의 $\mathbf{x}$에 대한 평균 loss)
- For each query point $\mathbf{x}$, the expected loss is different. We are interested in minimizing the expectation of the losses 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 $p(t|\mathrm{x})$. What value y should we predict?
  - Here, we are treating t as a random variable and choosing y.

Claim: $y_* = \mathbb{E}[t \mid \mathbf{x}]$ is the best possible prediction.

Proof:

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

Bayes Optimality

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

(목표: 위 식의 최소값일때 y 값 구하기)

- 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**.

  (예: Gaussian noise를 가지는 true function $t = f(x) + \epsilon, \epsilon \sim N(0, \sigma)$)
  - 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**.

Bayes Optimality

- 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):

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

Bayes Optimality

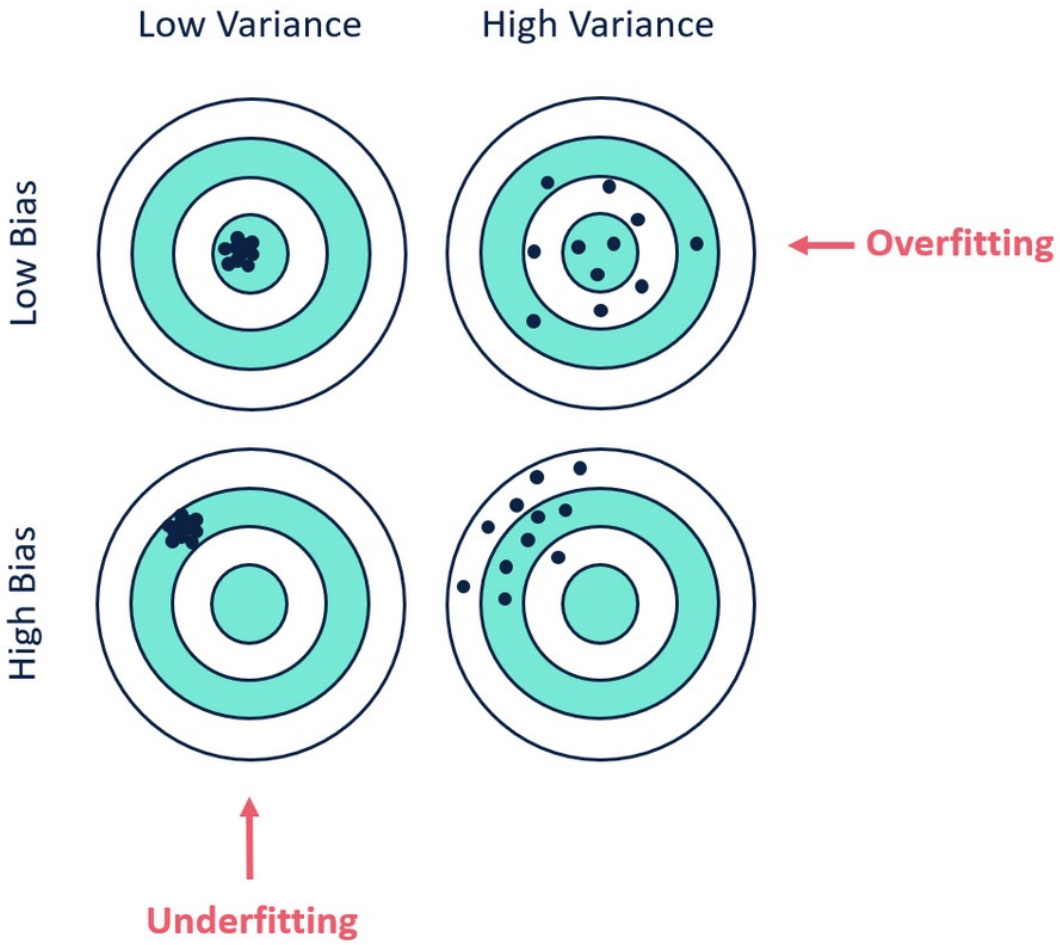
$$\mathbb{E}[(y - t)^2] = \underbrace{(y_* - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\text{Var}(y)}_{\text{variance}} + \underbrace{\text{Var}(t)}_{\text{Bayes error}}$$

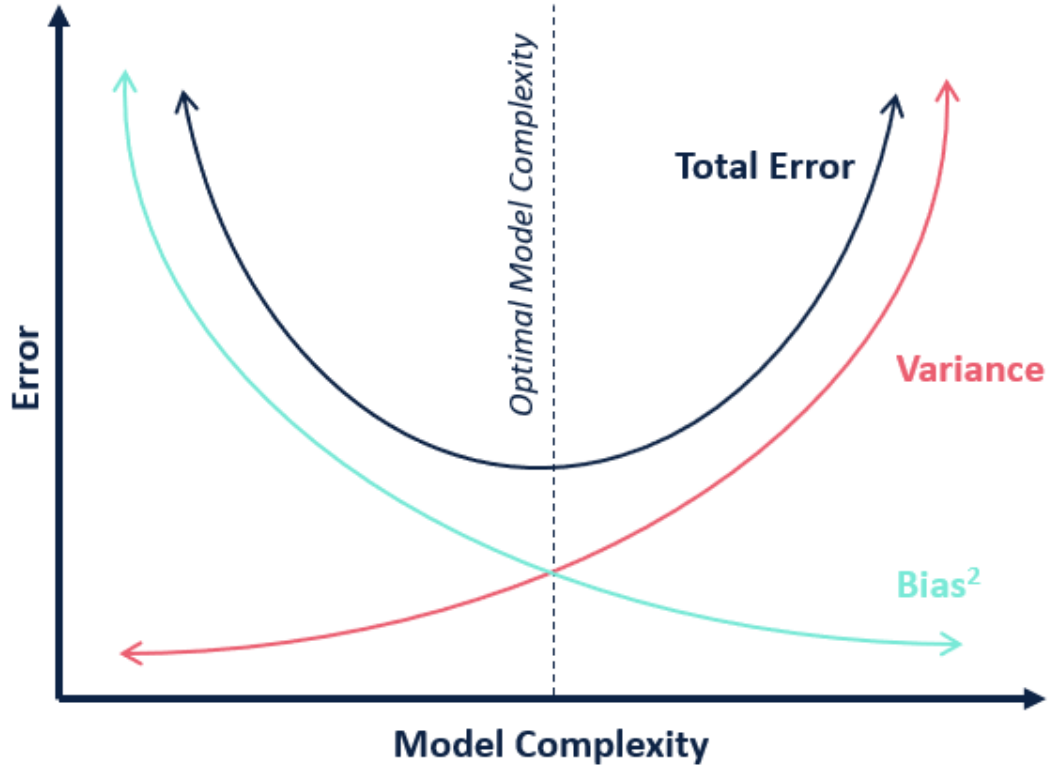
$$\textit{total error} = \textit{irreducible error} + \underbrace{\textit{error due to bias} + \textit{error due to variance}}_{\textit{reducible error}}$$

- 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

Bayes Optimality

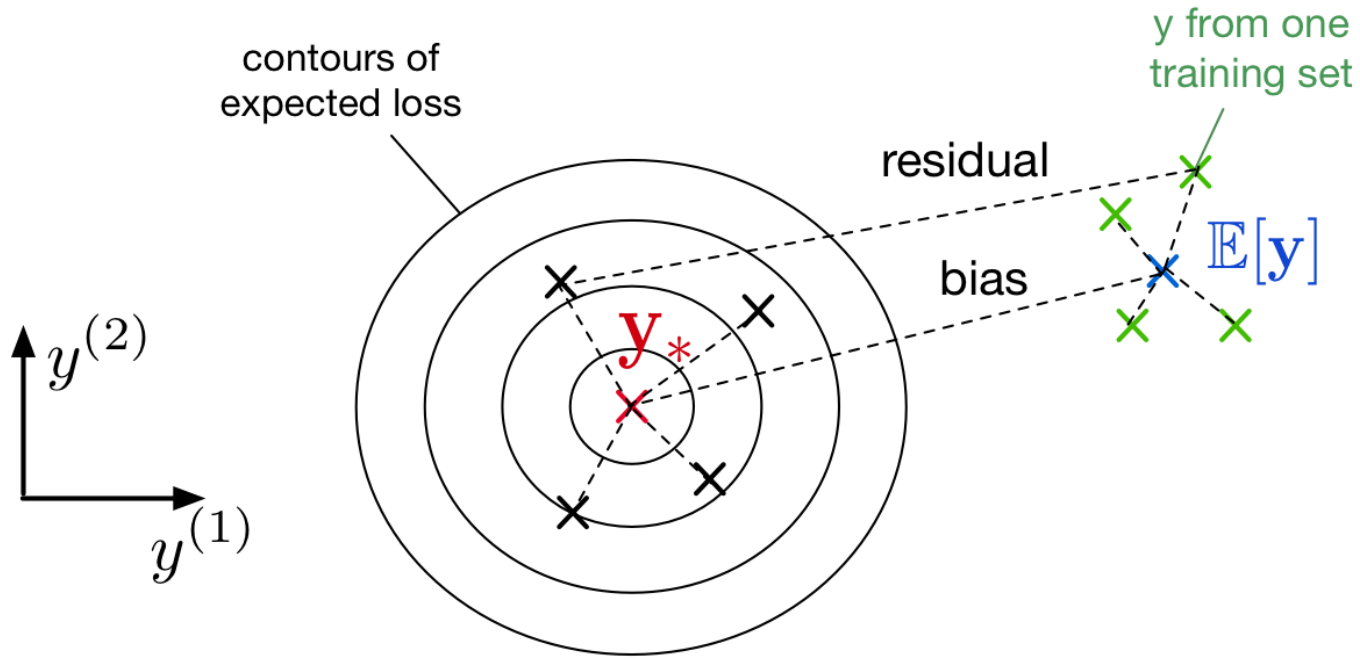
- Even though this analysis only applies to squared error, we often loosely use “bias" and “variance" as synonyms for “underfitting" and “overfitting"





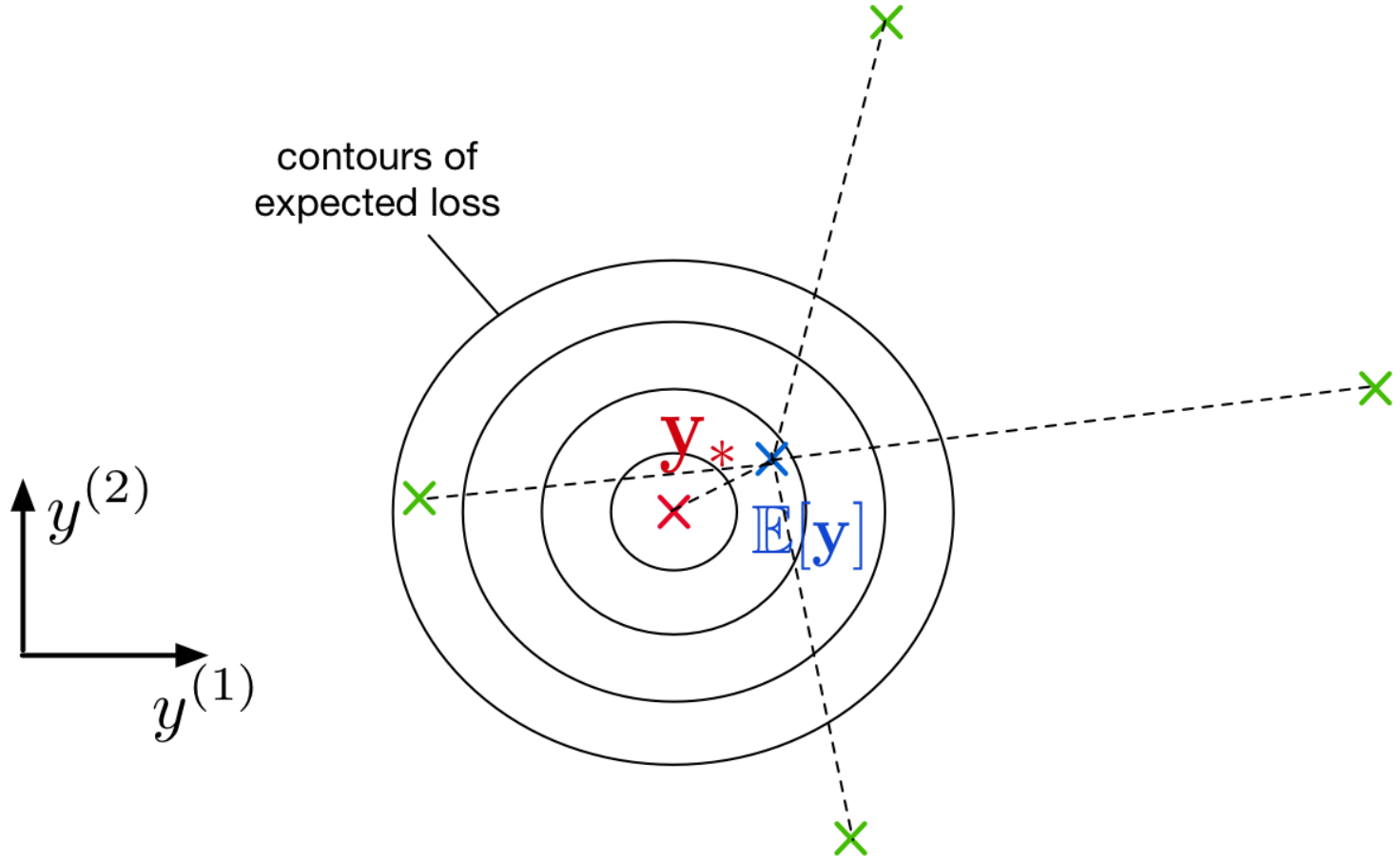
Bias/Variance Decomposition: Another Visualization

- 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. k-NN 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. k-NN 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)



Bagging

- Now, back to bagging!

Bagging: Motivation

- Suppose we could somehow sample m independent training sets from p_{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, we don't have access to the underlying data generating distribution p_{data}.
- It is expensive to independently collect many datasets.
- Solution: **bootstrap aggregation**, or **bagging**.
  - Take a single dataset $\mathfrak{D}$ with *n* examples.
  - Generate *m* new datasets, each by sampling *n* training examples from $\mathfrak{D}$, with replacement.
  - Average the predictions of models trained on each of these datasets.

Bagging: The Idea

- Problem: the datasets are not independent, so we don't get the $\frac{1}{m}$ 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.

(data set을 완전히 독립적으로 sampling하는 것은 어려우므로 다른 종류의 모델들을 사용하여 예측들의 독립성을 높임. 즉 위의 식에서 ρ을 줄임)

Random Forests

- **Random forests** = bagged decision trees, with one extra trick to decorrelate the predictions (전 페이지의 식에서 ρ을 줄임)
- Collection of independently-trained binary decision trees
  - 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

Application: Microsoft Xbox 360 Kinect

- Body part recognition: supervised learning

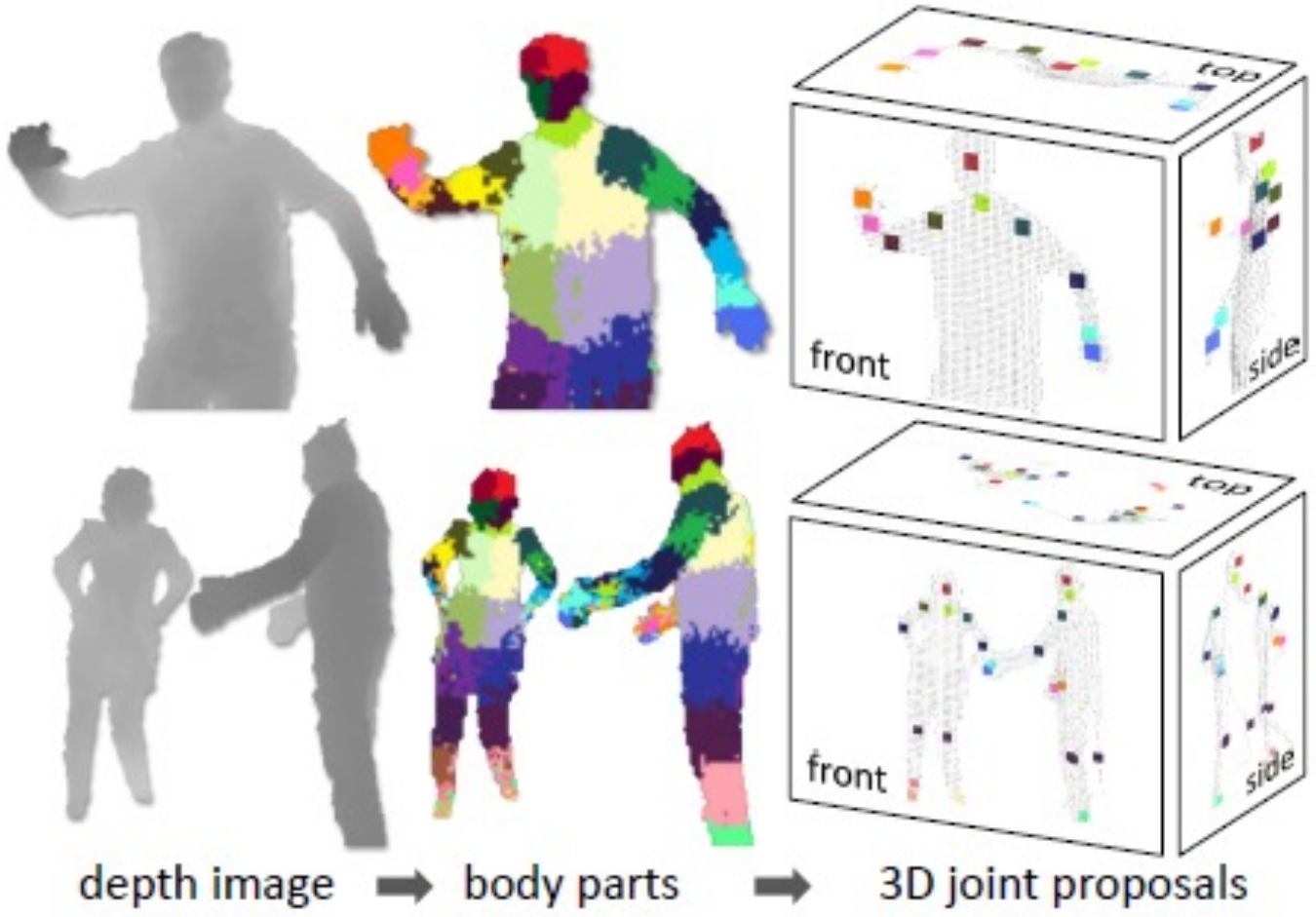
- Kinect (Depth camera)

Infrared image

Gray scale depth map

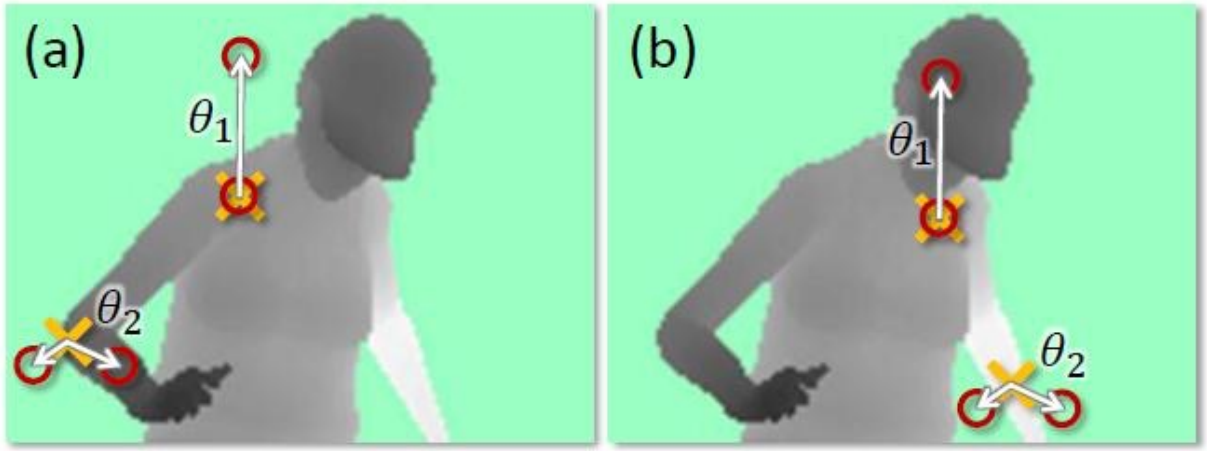
Kinect Body Part Recognition

- Problem: label each pixel with a body part



Kinect Body Part Recognition

- Features: depth differences between pairs of pixels

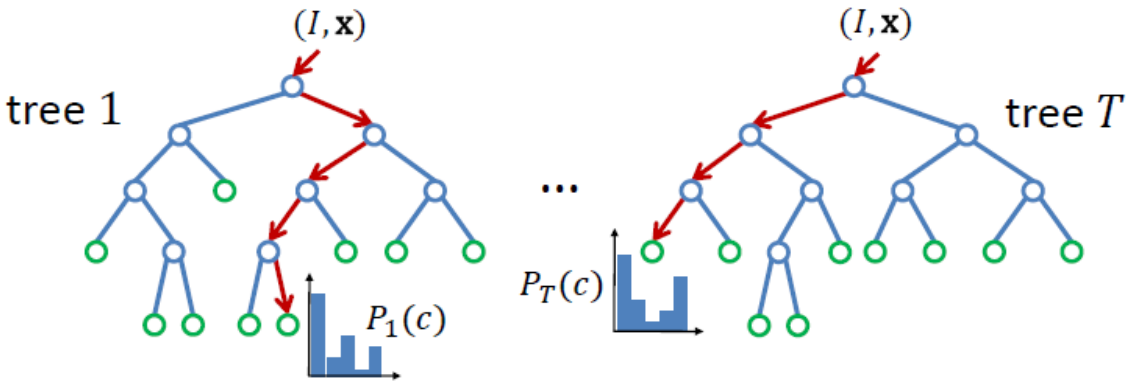


$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

$d_I(\mathbf{x})$ is depth image, $\theta = (\mathbf{u}, \mathbf{v})$ is offset to second pixel

- Classification: forest of decision trees
  - Classify each pixel **x** in image *I* using all decision trees and average the results at the leaves:

$$P(c|I, \mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I, \mathbf{x})$$



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 a correlation between classifiers.
- Random forest solution: Add more randomness.