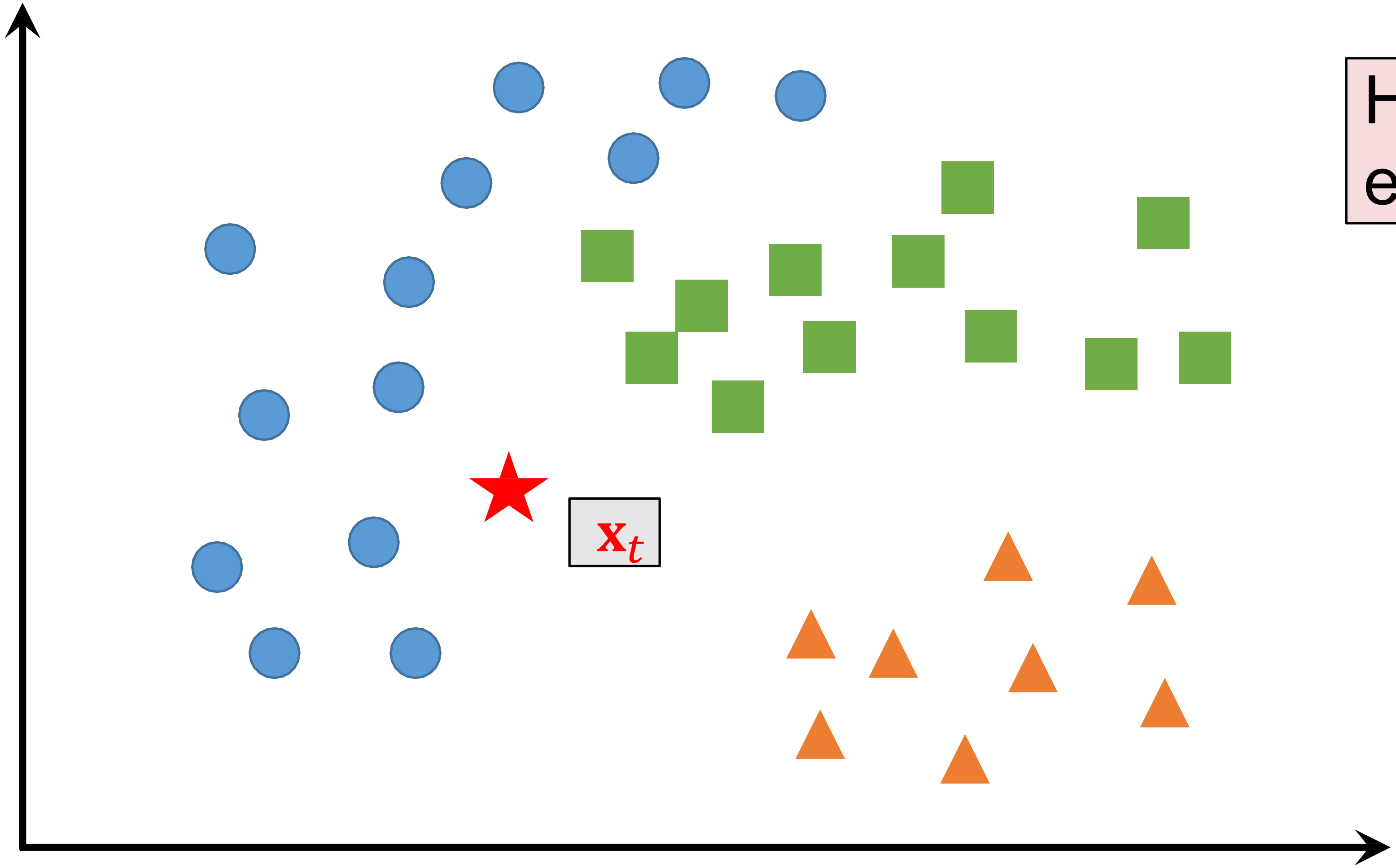
Nearest Neighbor Classifier
& Cross Validation
& Bias-Variance Tradeoff

***k*-Nearest Neighbor (*k*NN)**

Multi-Class Classification

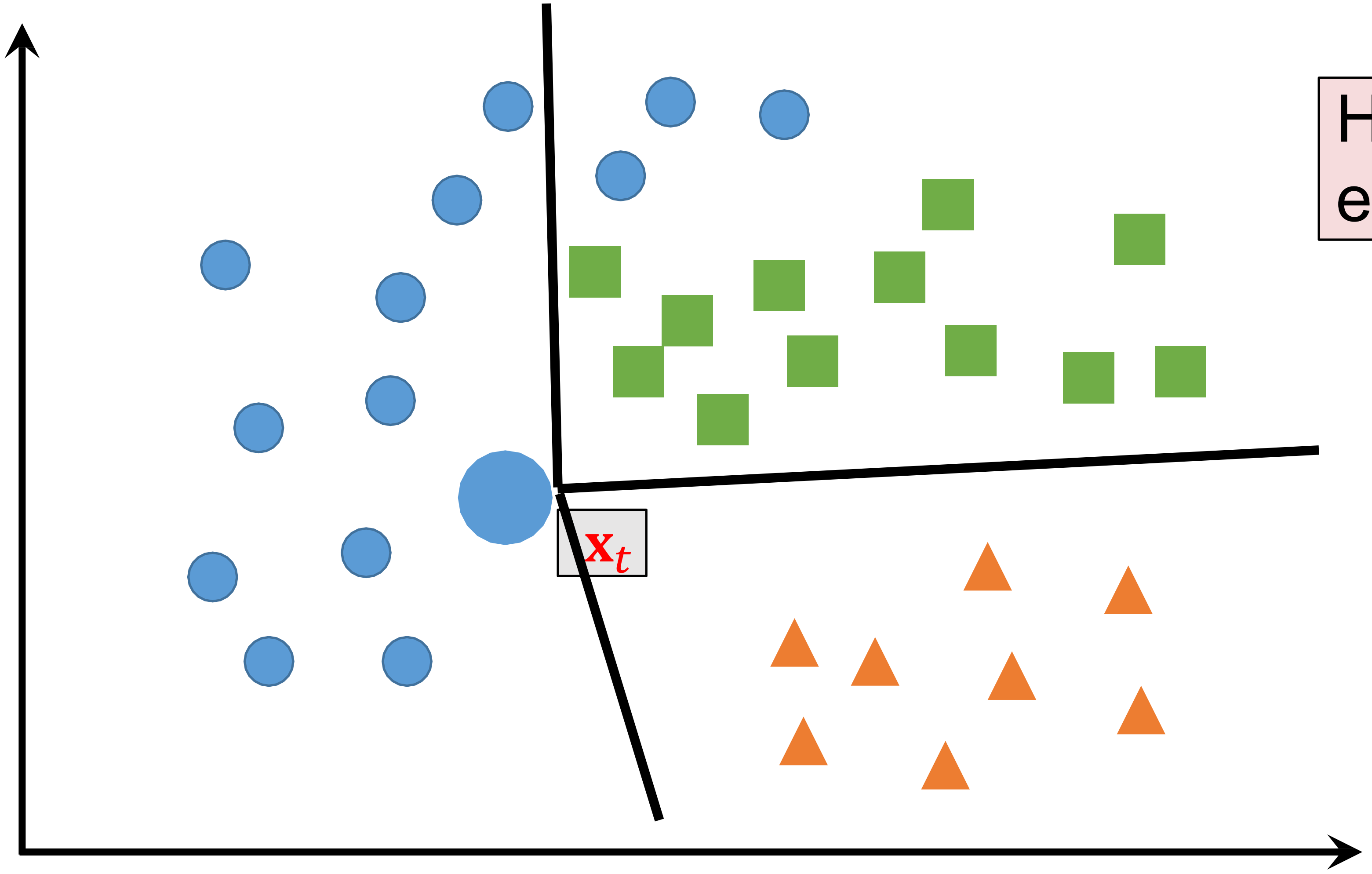
Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.



How to classify a test example $\mathbf{x}_t$?

Linear Models: Perceptron, SVM

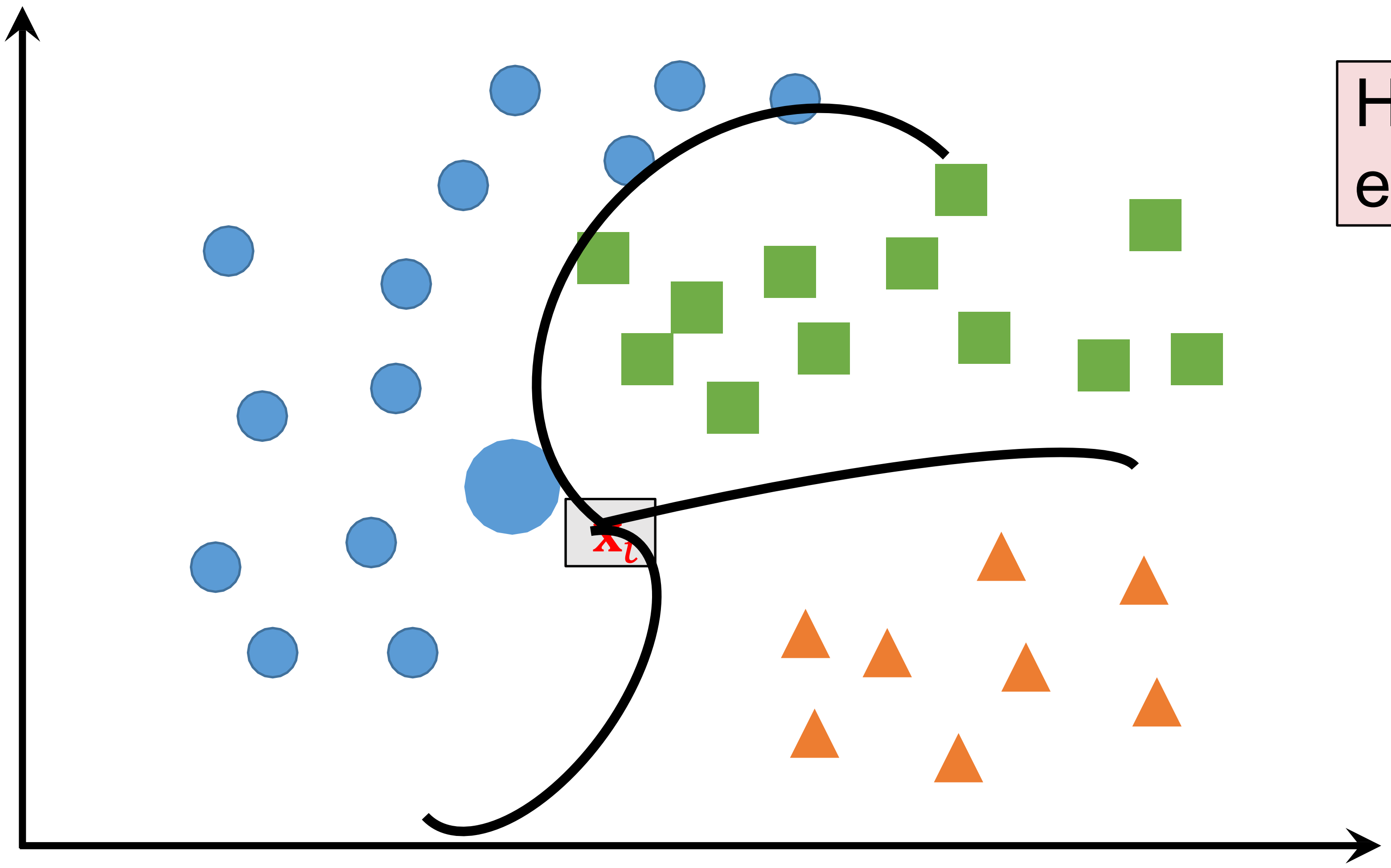
Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.



How to classify a test example $\mathbf{x}_t$?

Nonlinear Models: Deep Neural Net

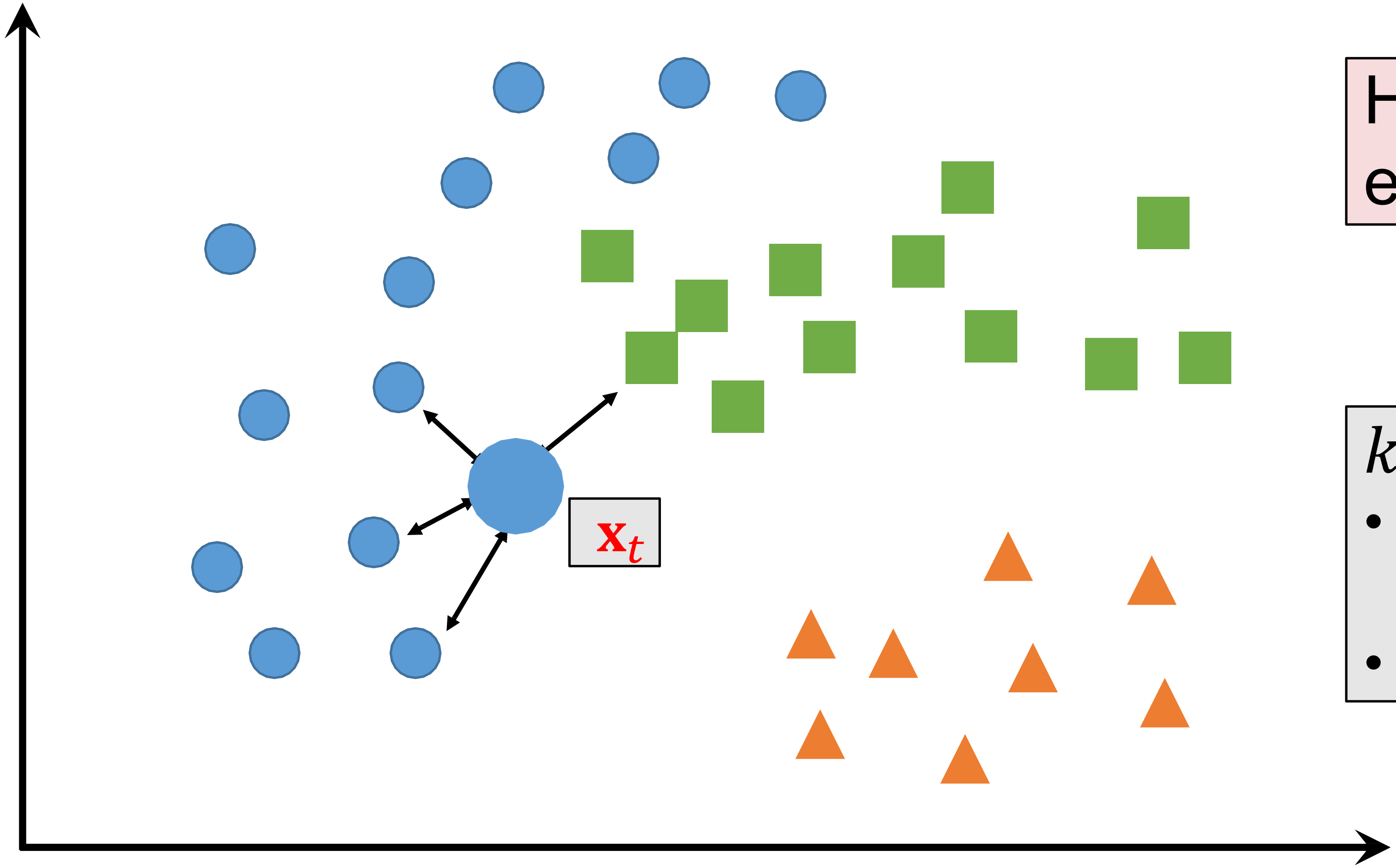
Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.



How to classify a test example $\mathbf{x}_t$?

K-Nearest Neighbor (kNN) Classifier

Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.



How to classify a test example $\mathbf{x}_t$?

kNN classifier
- Find the k nearest neighbors (NNs) of $\mathbf{x}_t$
- Let the kNNs vote

K-Nearest Neighbor (kNN) Classifier

Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.

k-Nearest Neighbor (kNN) classifier

- Find the k nearest neighbors of $\mathbf{x}_t$
- Let the NNs vote

Question: How to measure similarity?

- Cosine similarity: $\mathrm{sim}(\mathbf{x}, \mathbf{x}_t) = \frac{\mathbf{x}^T \mathbf{x}_t}{\|\mathbf{x}\|_2 \|\mathbf{x}_t\|_2}$
- Gaussian kernel: $\mathrm{sim}(\mathbf{x}, \mathbf{x}_t) = \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{x}_t\|_2^2\right)$
- Laplacian kernel: $\mathrm{sim}(\mathbf{x}, \mathbf{x}_t) = \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{x}_t\|_1\right)$

K-Nearest Neighbor (kNN) Classifier

Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.

k-Nearest Neighbor (kNN) classifier

- Find the k nearest neighbors of $\mathbf{x}_t$
- Let the NNs vote

Question: How to find the k nearest neighbors?

- Naïve algorithm
  - Compute all the similarities $\mathrm{sim}(\mathbf{x}_1, \mathbf{x}_t), \cdots, \mathrm{sim}(\mathbf{x}_N, \mathbf{x}_t)$
  - Sort the scores and find the top k
  - Time complexity O(Nd)
- More efficient algorithms? (to be discussed later)

K-Nearest Neighbor (kNN) Classifier

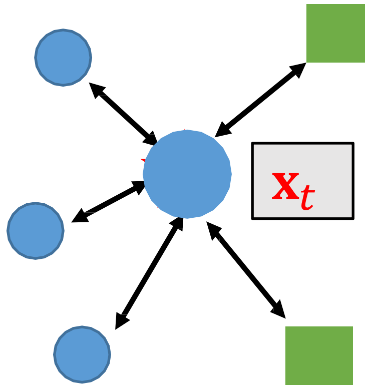
Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.

k-Nearest Neighbor (kNN) classifier

- Find the k nearest neighbors of $\mathbf{x}_t$
- Let the NNs vote

Question: How to vote?

- Option 1: Every neighbor has the same weight



K-Nearest Neighbor (kNN) Classifier

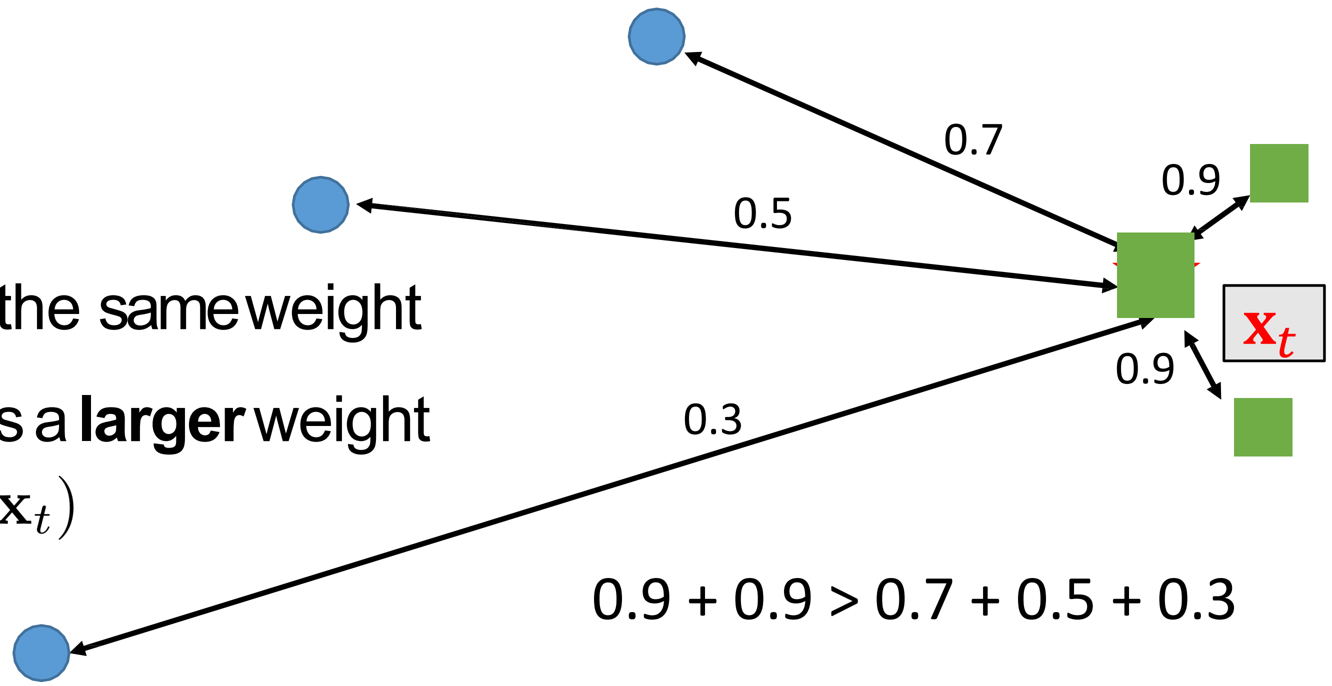
Input: Training examples $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \dots, t_N \in \mathbb{N}$.

k-Nearest Neighbor (kNN) classifier

- Find the k nearest neighbors of $\mathbf{x}_t$
- Let the NNs vote

Question: How to vote?

- Option 1: Every neighbor has the same weight
- Option 2: Nearer neighbor has a **larger** weight
  - e.g., $\mathrm{weight}_n = \mathrm{sim}(\mathbf{x}_n, \mathbf{x}_t)$



0.9 + 0.9 > 0.7 + 0.5 + 0.3

KNN: Naïve Algorithm

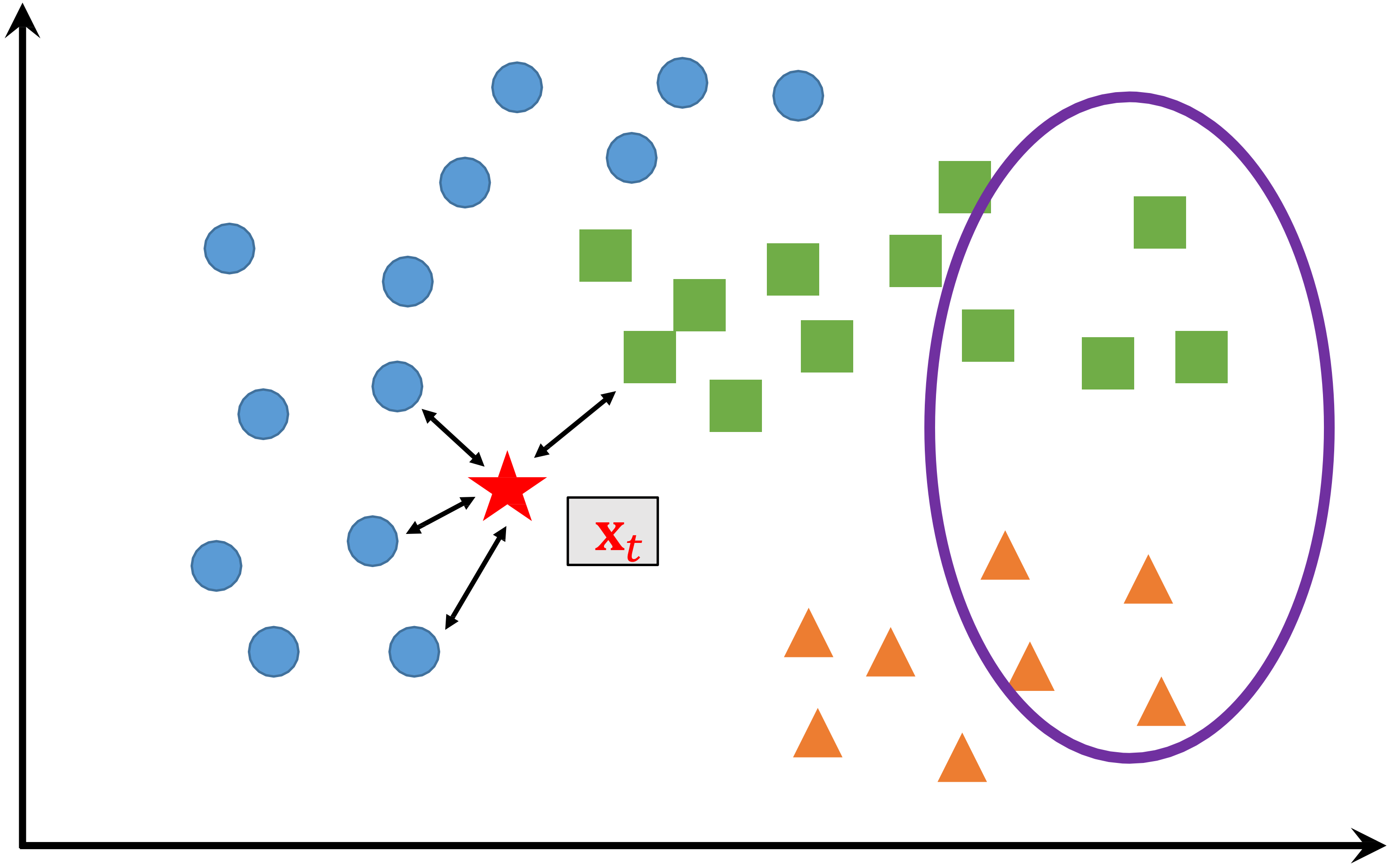
Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.

Algorithm: find the k nearest neighbors to $\mathbf{x}_t$

- Naïve algorithm
  - Compute all the similarities $\mathrm{sim}(\mathbf{x}_1, \mathbf{x}_t), \cdots, \mathrm{sim}(\mathbf{x}_N, \mathbf{x}_t)$ and find the top k
- NO training at all
- Test: for each query, O(Nd) time complexity

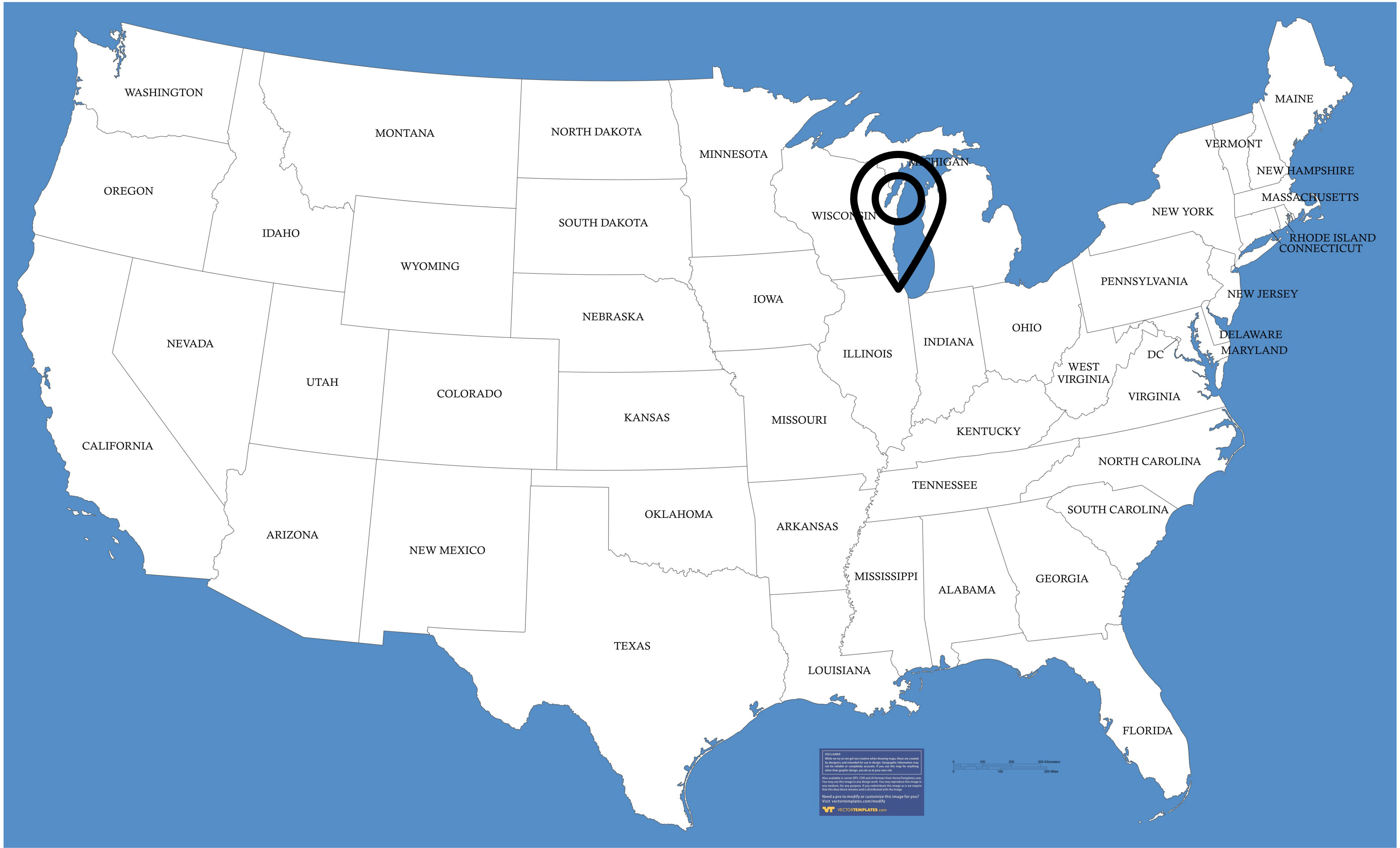
KNN: Efficient Algorithm

Input: Training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ and labels $t_1, \ldots, t_N \in \mathbb{N}$.



Does we need to look at this region?

Question: find your nearest post office (given longitude & latitude)



Question: find your nearest post office (given longitude & latitude)

Data: N = 30,000 post offices' latitude and longitude

- Post office 1: $(lat_1, \ lon_1)$
- Post office 2: $(lat_2, \ lon_2)$
- Post office 3: $(lat_3, \ lon_3)$

⋮

- Post office N: $(lat_N, \ lon_N)$

Query: your own latitude and longitude:

(41.8781° N, 87.6298° W)

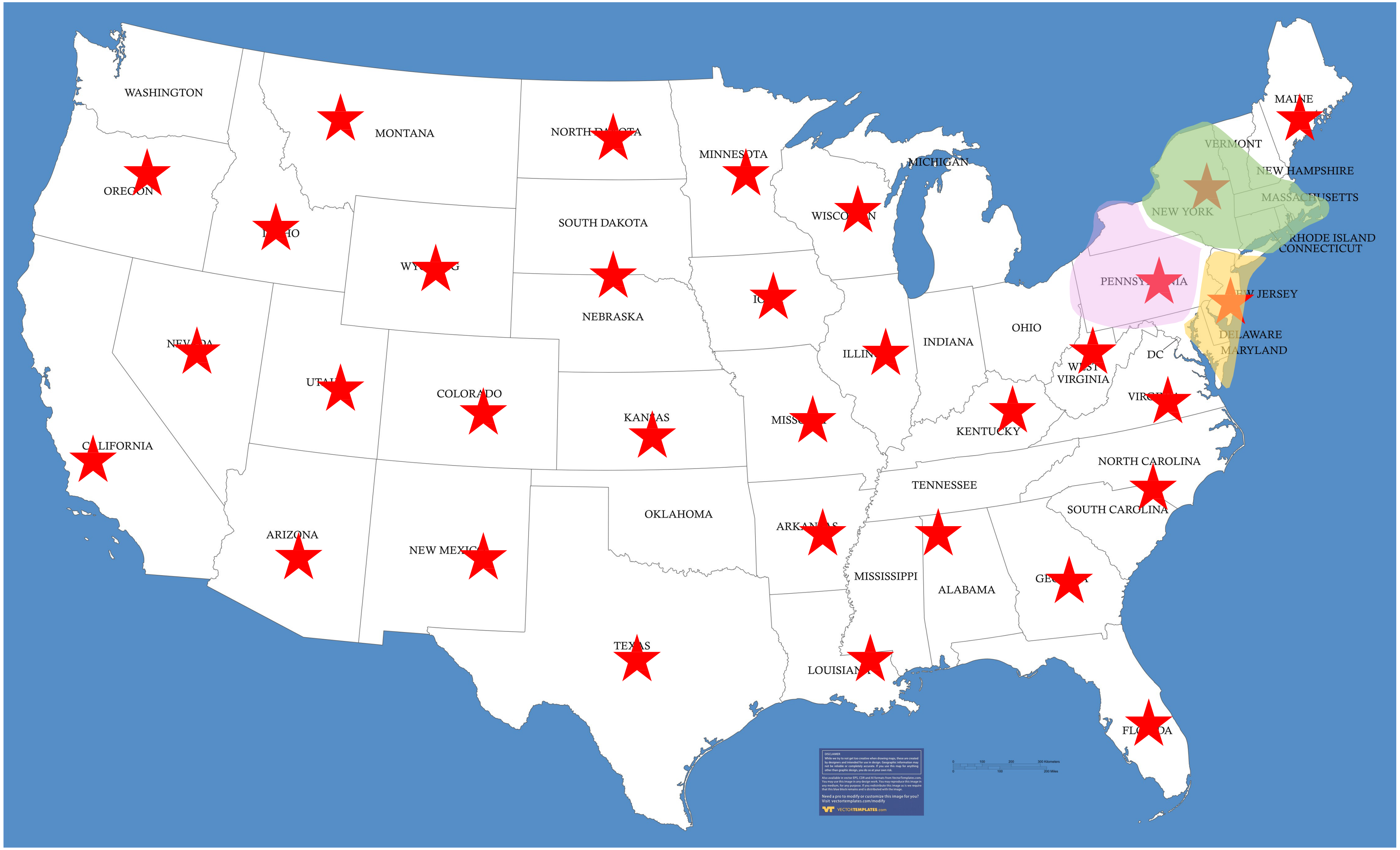
Question: Which one is your nearest post office?

Vector Quantization for KNN

Training

1. Vector quantization (build landmarks)

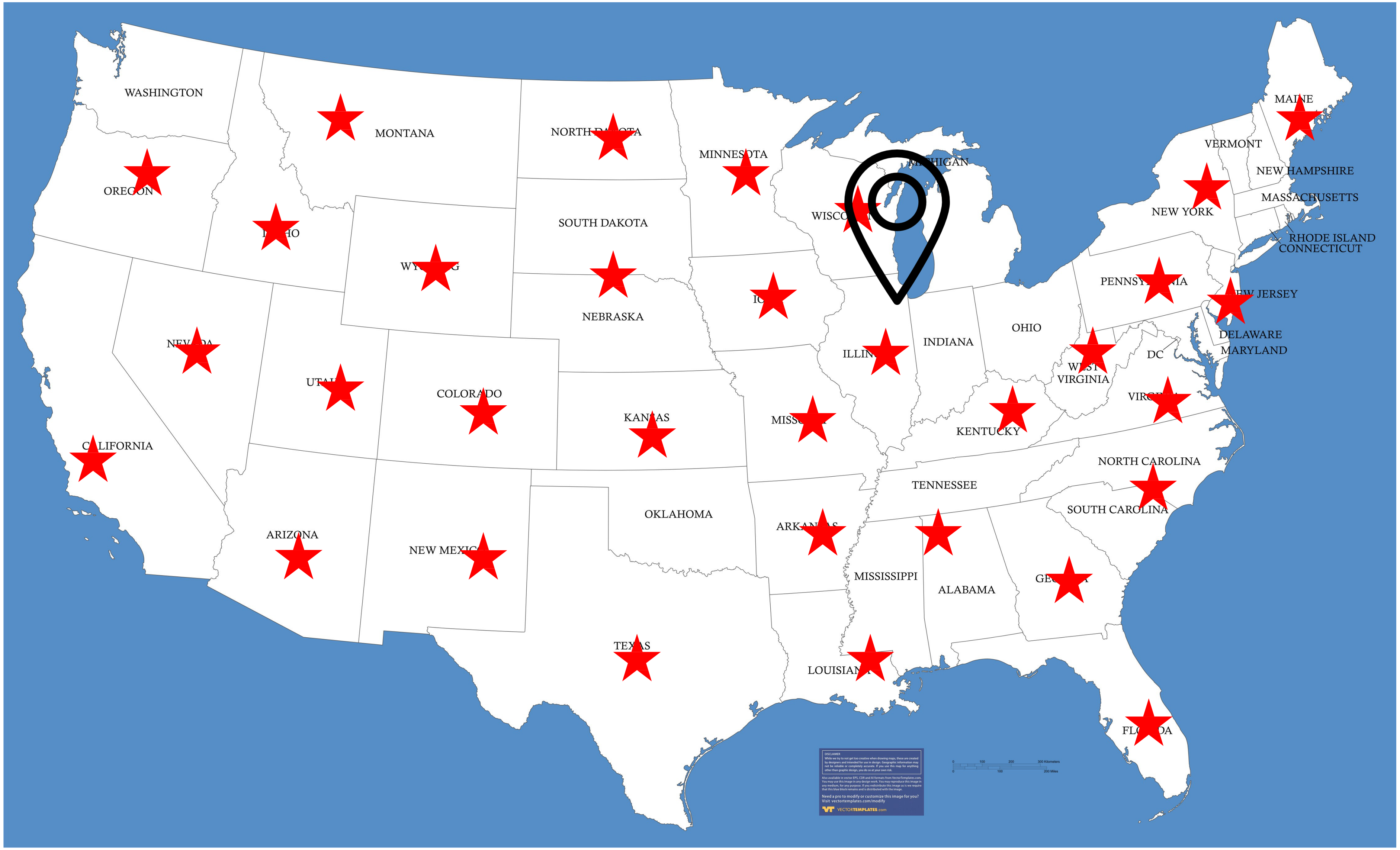
Vector Quantization for KNN



Training

1. Vector quantization (build landmarks)
2. Assign each post office to its nearest landmarks

Vector Quantization for KNN



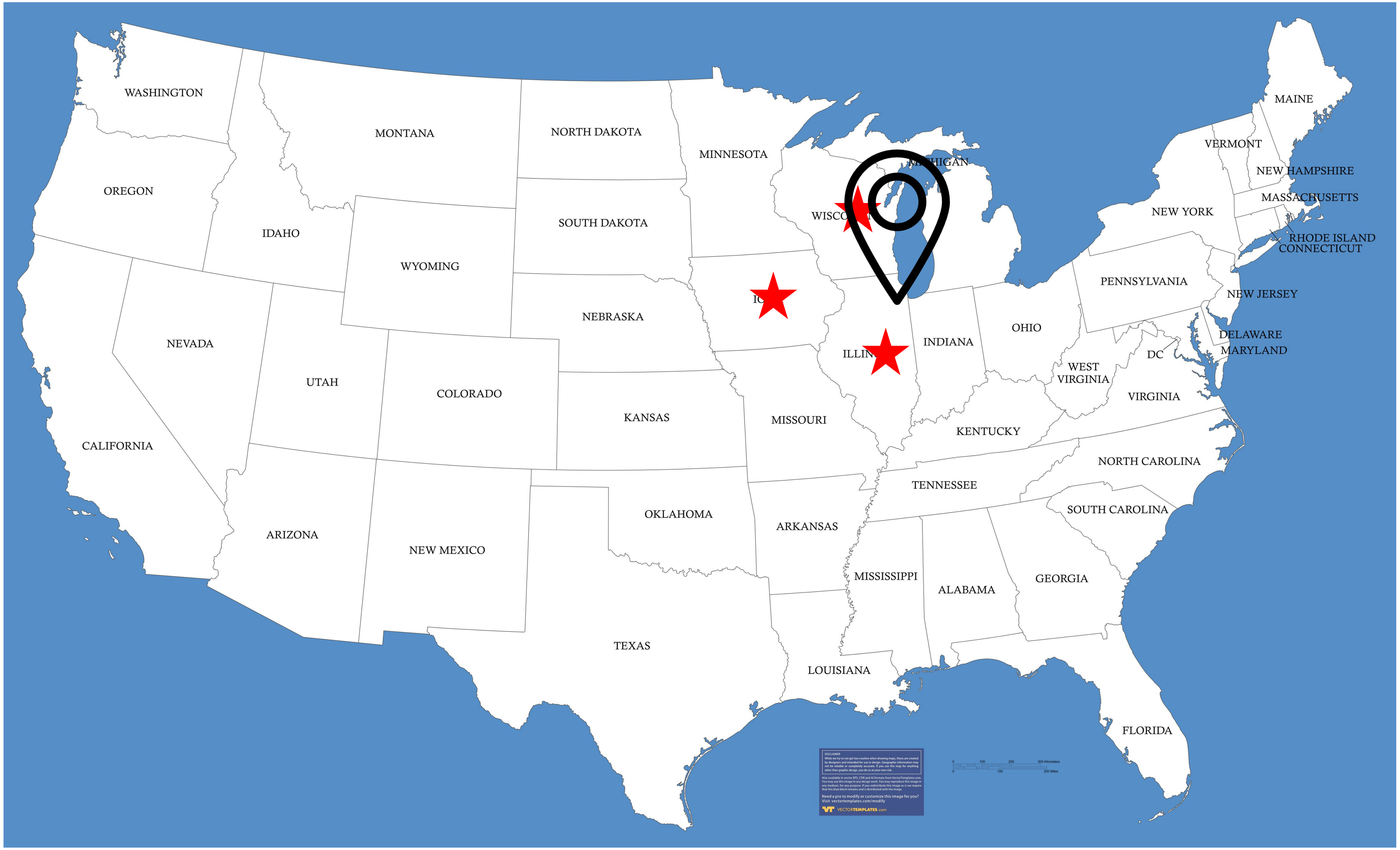
Training

1. Vector quantization (build landmarks)
2. Assign each post office to its nearest landmarks

Testing

1. Compare your location with all the landmarks and find the nearest landmarks

Vector Quantization for KNN



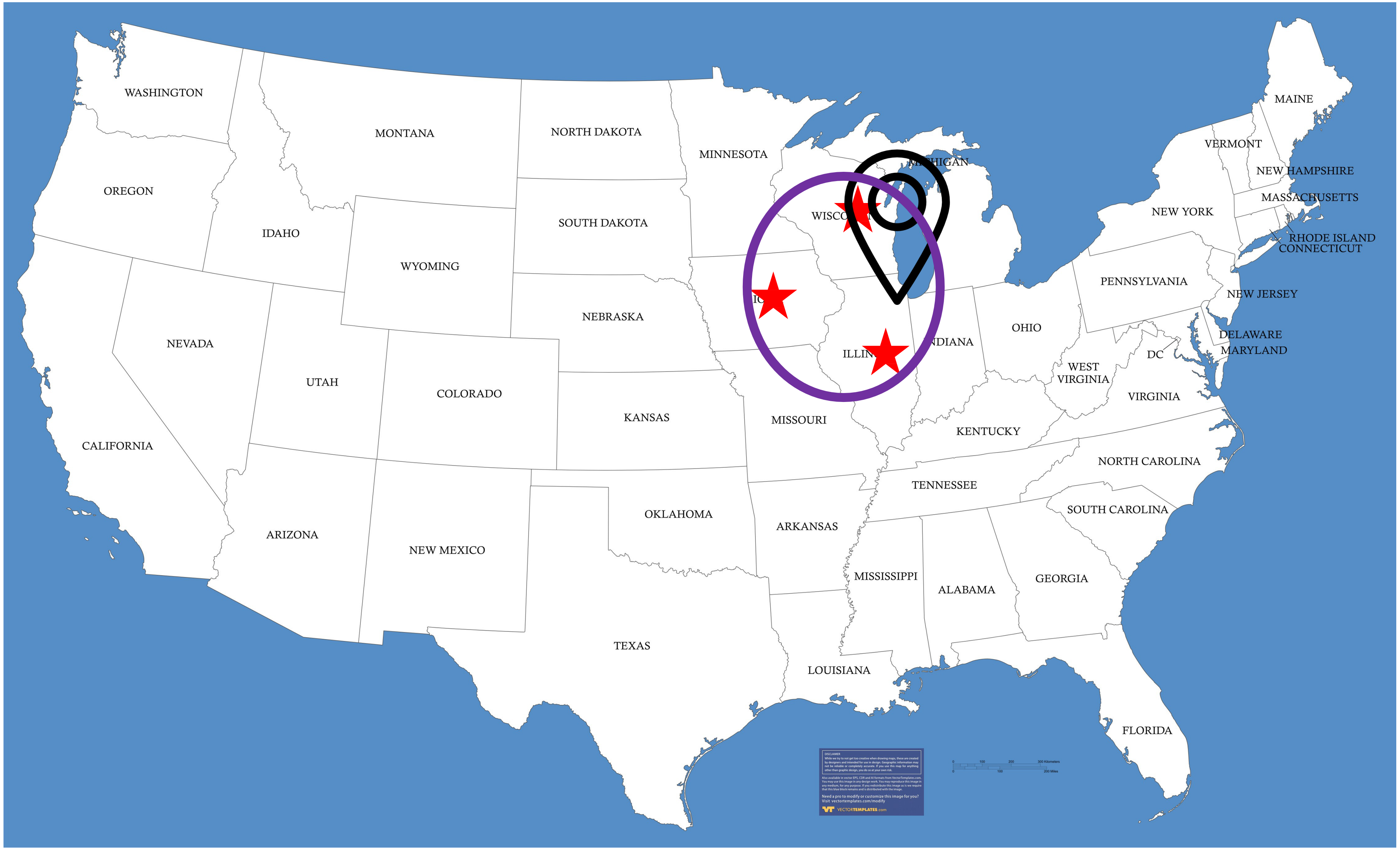
Training

1. Vector quantization (build landmarks)
2. Assign each post office to its nearest landmarks

Testing

1. Compare your location with all the landmarks and find the nearest landmarks

Vector Quantization for KNN



Training:

1. Vector quantization (build landmarks)
2. Assign each post office to its nearest landmarks

Testing

1. Compare your location with all the landmarks and find the nearest landmarks
2. Compare with the postal offices assigned to the landmarks

KNN: Efficient Algorithms

- Vector Quantization
  - Clustering based method
- KD-tree
- Locality sensitive hashing
- More resources
  - KNN Search (Wikipedia)

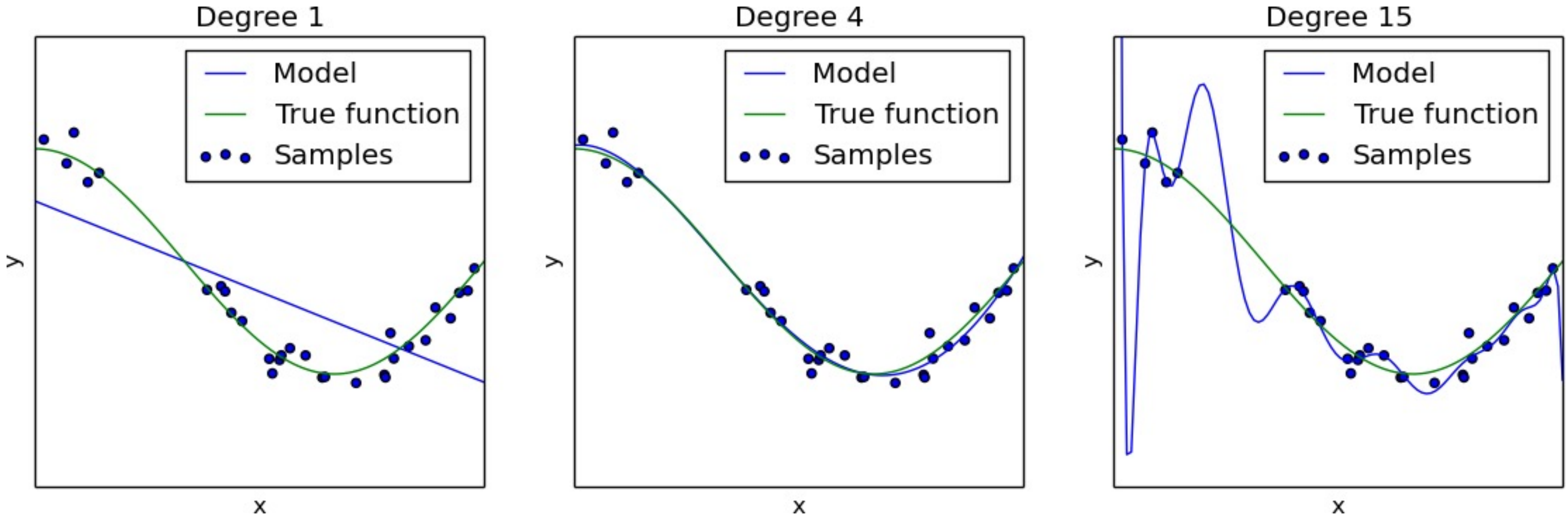
Hyperparameter Tuning: Cross Validation

Hyperparameters

- Parameters that cannot be directly learnt from the model
  - Polynomial regression degree: *p* $f(\mathbf{x};\mathbf{w}) = \sum_{j=0}^{p} w_j x^j$
  - *Regularized linear regression: λ* $\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \|\mathbf{w}\|_p^p$
  - *Gaussian/RBF kernel SVM: σ* $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / 2\sigma^2\big)$
  - *(Stochastic) gradient descent: α* $\mathbf{x}_{(t+1)} = \mathbf{x}_{(t)} - \alpha \mathbf{g}_{(t)}$
  - *#Layers, #hidden neurons, batch size in deep neural networks*
  - *K-nearest neighbor: k*

How to learn good hyperparameters?

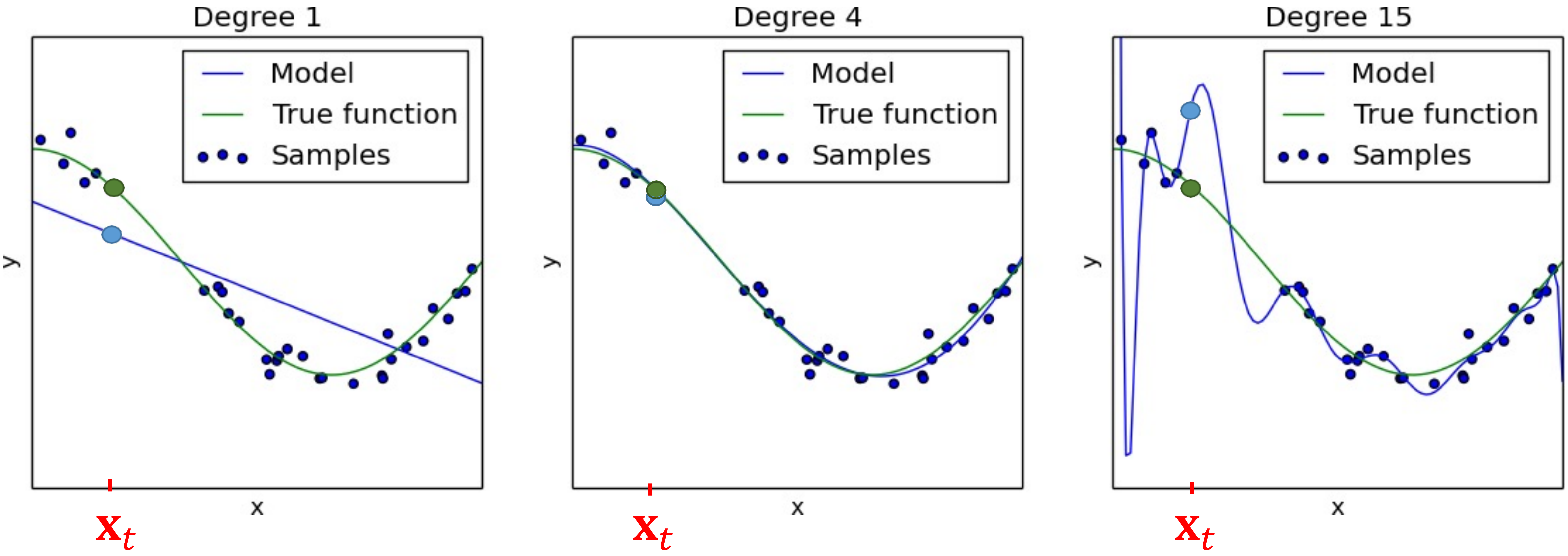
Polynomial Regression



Underfitting

Overfitting

Polynomial Regression

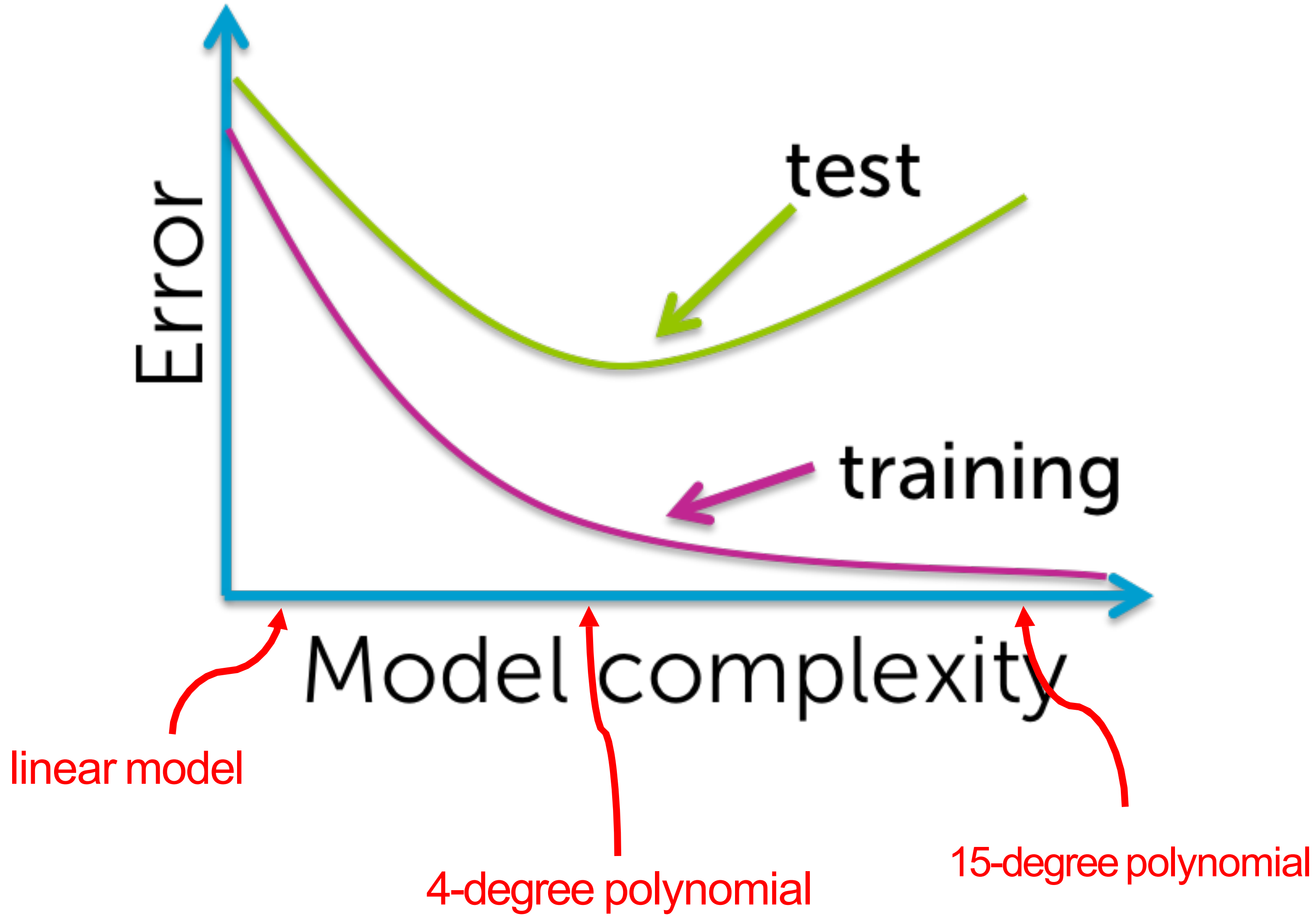


BAD

GOOD

BAD

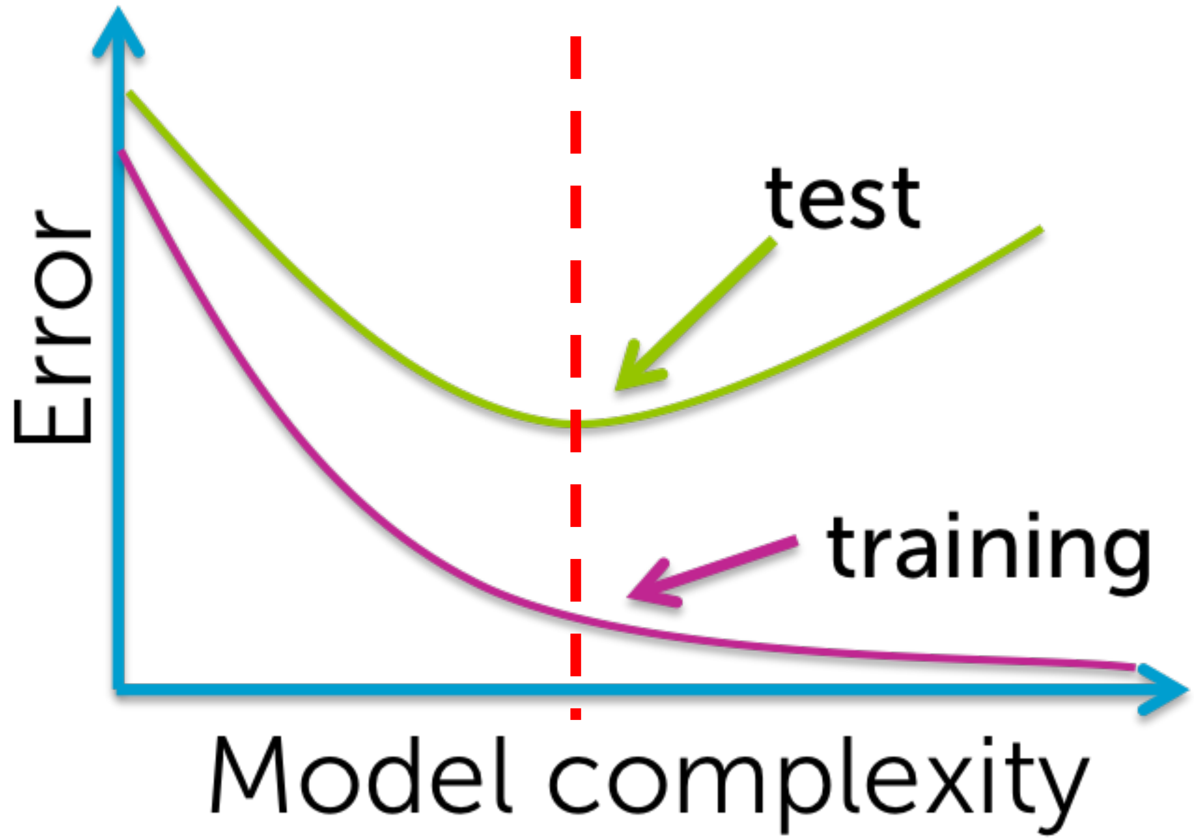
Training Error vs Testing Error



Hyperparameter Tuning

Question: For the polynomial regression model, how to determine the degree p?

Answer: The degree p leads to the smallest testerror



Hyper-Parameter Tuning

Training Set		Test Set
Train a degree-1 polynomial regression	→	Test MSE = 23.2
Train a degree-2 polynomial regression	→	Test MSE = 19.0
Train a degree-3 polynomial regression	→	Test MSE = 16.7
Train a degree-4 polynomial regression	→	Test MSE = 12.2
Train a degree-5 polynomial regression	→	Test MSE = 14.8
Train a degree-6 polynomial regression	→	Test MSE = 25.1
Train a degree-7 polynomial regression	→	Test MSE = 39.4
Train a degree-8 polynomial regression	→	Test MSE = 53.0

Hyperparameter Tuning

Training Set		Test Set
Train a degree-1 polynomial regression	→	Test MSE = 23.2
Train a degree-2 polynomial regression	→	Test MSE = 19.0
Train a degree-3 polynomial regression	→	Test MSE = 16.7
Train a $\boxed{\text{degree-4 polynomial}}$ regression	→	$\boxed{\text{Test MSE = 12.2}}$
Train a degree-5 polynomial regression	→	Test MSE = 14.8
Train a degree-6 polynomial regression	→	Test MSE = 25.1
Train a degree-7 polynomial regression	→	Test MSE = 39.4
Train a degree-8 polynomial regression	→	Test MSE = 53.0

Hyperparameter Tuning

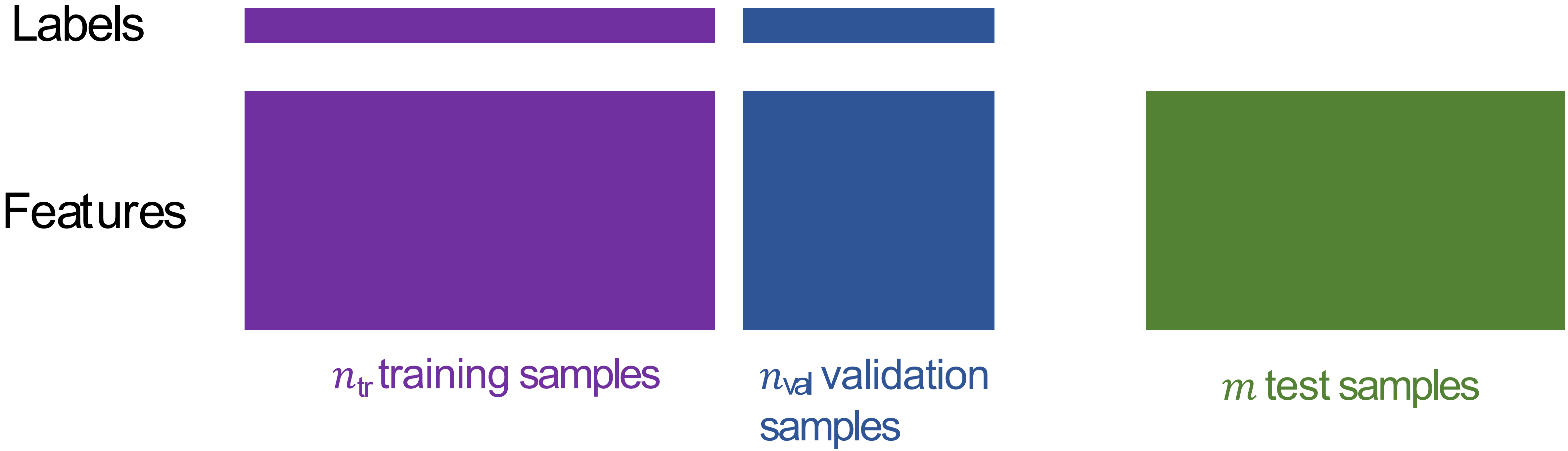
Training Set		Test Set
Train a degree-1 polynomial regression	⟶	Test MSE = 23.2
Train a degree-2 polynomial regression	⟶	Test MSE = 19.0
Train a degree-3 polynomial regression	⟶	Test MSE = 16.7
Train a degree-4 polynomial regression	⟶	Test MSE = 12.2
Train a degree-5 polynomial regression	⟶	Test MSE = 14.8
Train a degree-6 polynomial regression	⟶	Test MSE = 25.1
Train a degree-7 polynomial regression	⟶	Test MSE = 39.4
Train a degree-8 polynomial regression	⟶	Test MSE = 53.0

Wrong! The test labels are unavailable!

Even if you have the test labels, never do this!

Cross Validation

Cross Validation



Cross Validation

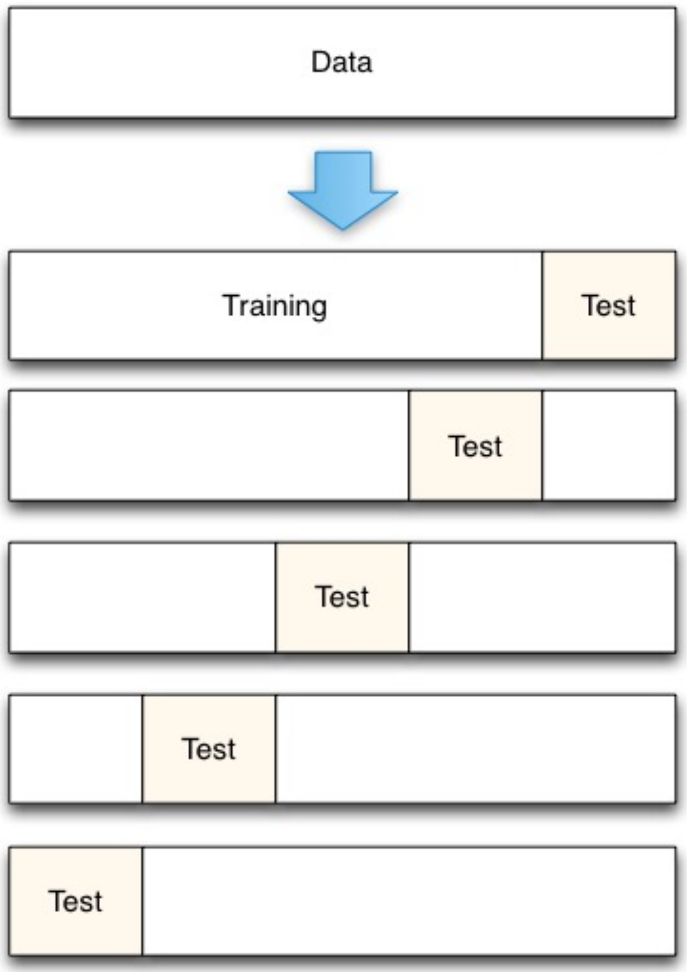
Training Set		Test Set
Train a degree-1 polynomial regression	→	Test MSE = [illegible]3.2
Train a degree-2 polynomial regression	→	Test MSE = [illegible]9.0
Train a degree-3 polynomial regression	→	Test MSE = 16.7
Train a degree-4 polynomial regression	→	Test MSE = 12.2
Train a degree-5 polynomial regression	→	Test MSE = 14.8
Train a degree-6 polynomial regression	→	Test MSE = 25.1
Train a degree-7 polynomial regression	→	Test MSE = [illegible]9.4
Train a degree-8 polynomial regression	→	Test MSE = 5[illegible].0

Cross Validation

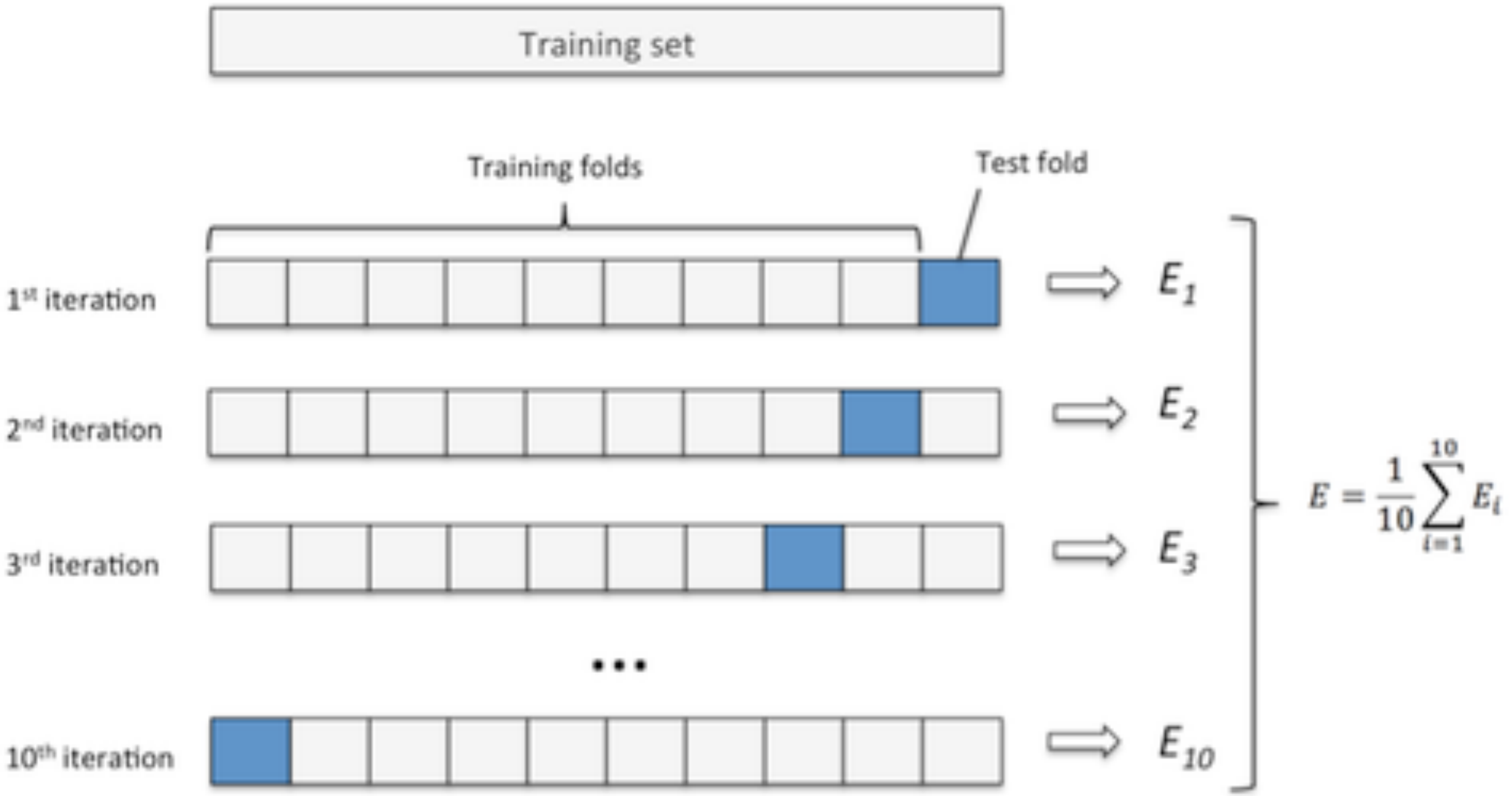
Training Set		Validation Set
Train a degree-1 polynomial regression	→	Valid. MSE = 23.1
Train a degree-2 polynomial regression	→	Valid. MSE = 19.2
Train a degree-3 polynomial regression	→	Valid. MSE = 16.3
Train a degree-4 polynomial regression	→	$\boxed{\text{Valid. MSE = 12.5}}$
Train a degree-5 polynomial regression	→	Valid. MSE = 14.4
Train a degree-6 polynomial regression	→	Valid. MSE = 25.0
Train a degree-7 polynomial regression	→	Valid. MSE = 39.1
Train a degree-8 polynomial regression	→	Valid. MSE = 53.5

k-Fold Cross-Validation

1. Propose a grid of hyperparameters
   - E.g. $p \in \{1, 2, 3, 4, 5\}$.
2. Randomly partition the training samples to k parts
   - k -1 parts for training
   - The remaining 1 part for test
3. Compute the averaged errors of the k repeats
   - Called the validation error
4. Choose the hyper-parameter p that leads to the smallest validation error



Example: 10-Fold Cross-Validation



Validation error

Example: 10-Fold Cross-Validation

Hyperparameter	Validation error	Test error
p=1	23.1	MSE = 12.2
p=2	19.2	
p=3	16.3	
p=4	12.5	
p=5	14.4	
…	…	

Bias-Variance Trade-Off

Bias **and Variance**

- Every learning algorithm has assumptions about the model hypothesis space
  - Linear
  - SVM with RBF kernel
  - A three-layer neural network with ReLU activations

- Bias
  - ***True error*** (loss) of the ***best*** classifier in the hypothesis space

- Underfitting: Large bias
  - Hypothesis space is simple
  - Classifiers from hypothesis space cannot represent the target function

Bias and *Variance*

- Performance of a classifier is dependent on the specific training set
  - Model will change if slightly changing the training set
- Variance
  - Describes how much the best classifier depends on the training set
- Overfitting: Large variance
  - Hypothesis space is complex
  - Classifiers are very flexible and unconstrained

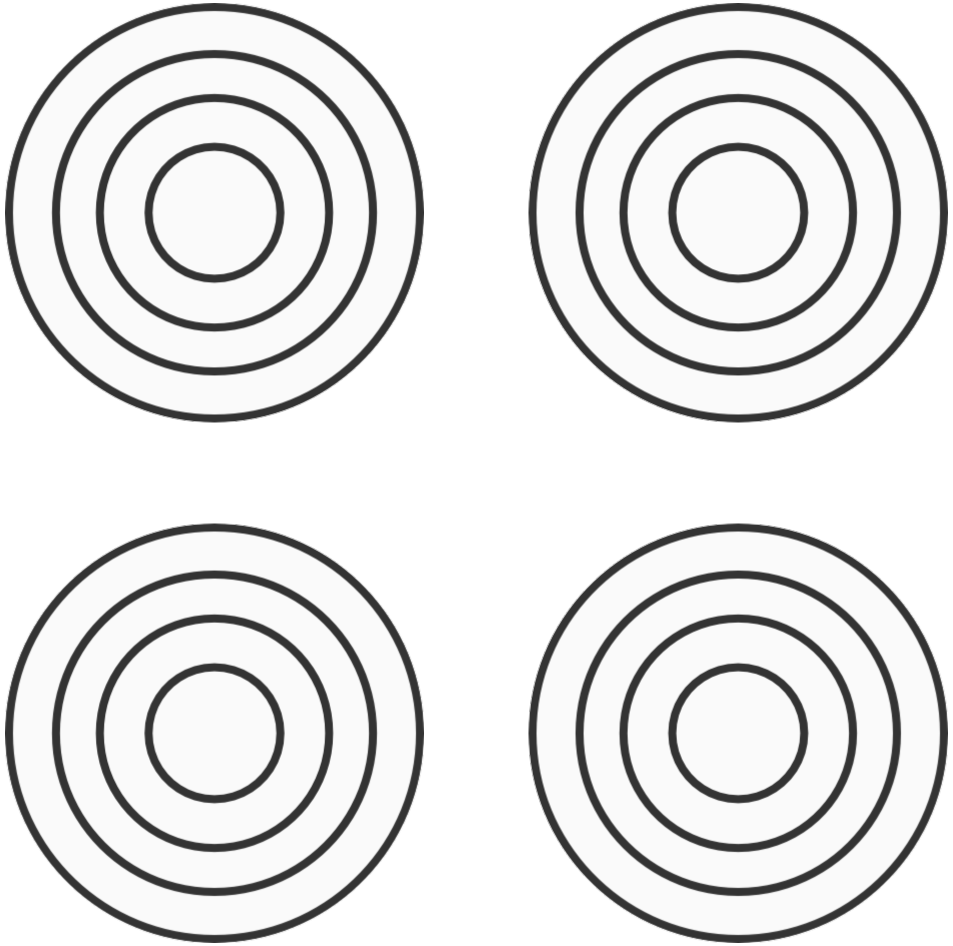
Let’s Play Darts

Suppose the optimal model is the center

Each dot is a learnt model

High bias

Low bias



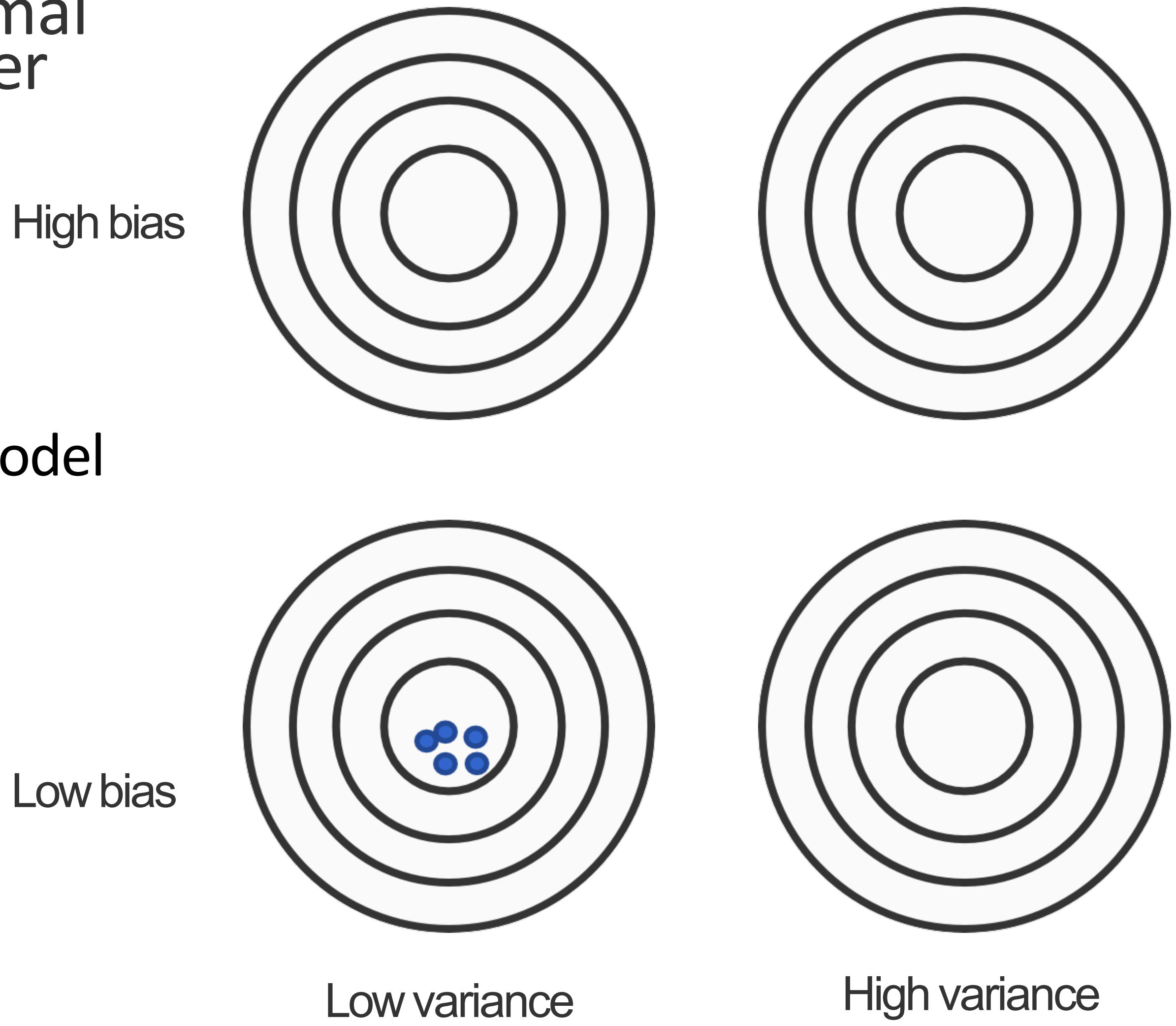
Low variance

High variance

Let’s Play Darts

Suppose the optimal model is the center

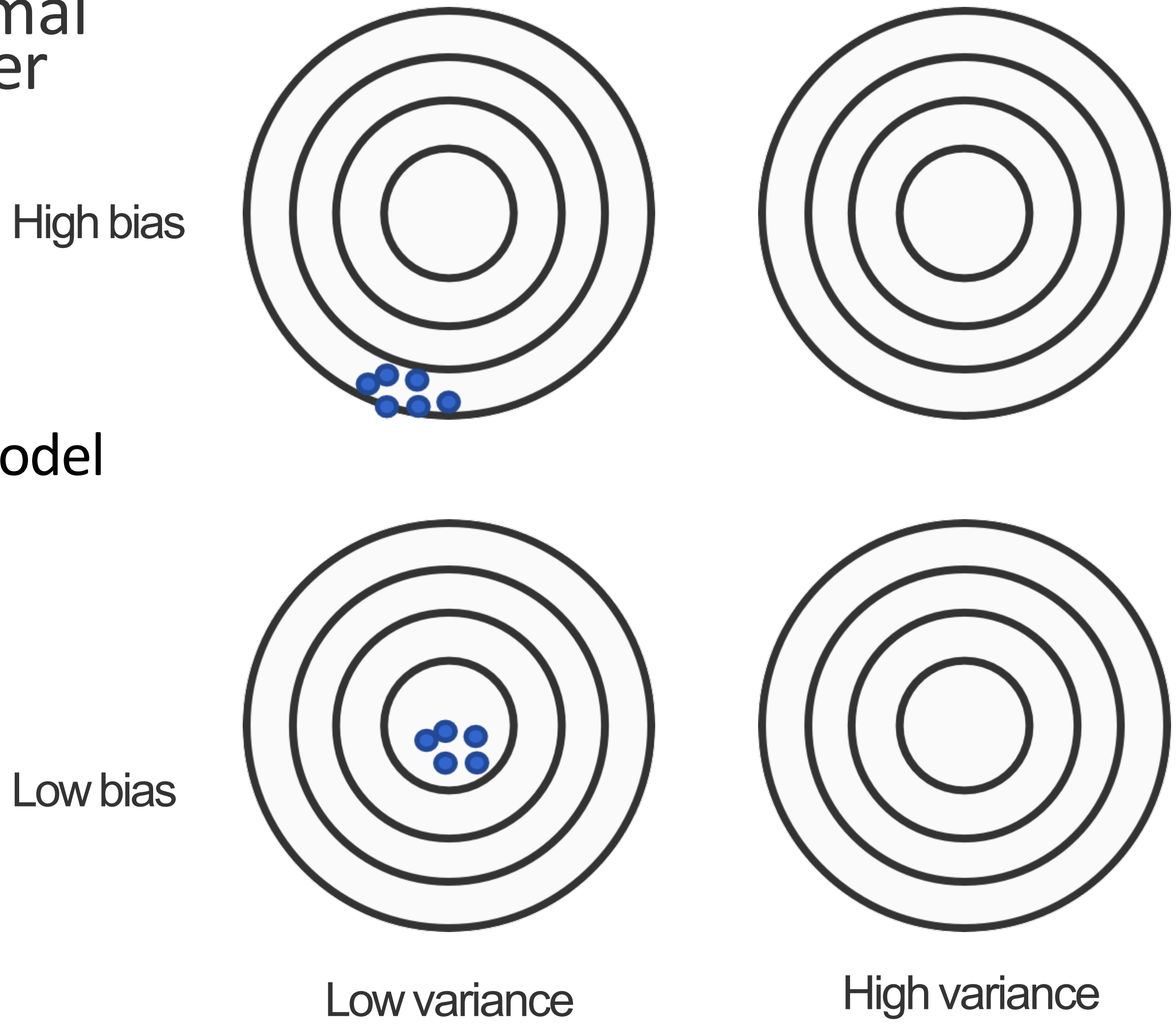
Each dot is a learnt model



Let’s Play Darts

Suppose the optimal model is the center

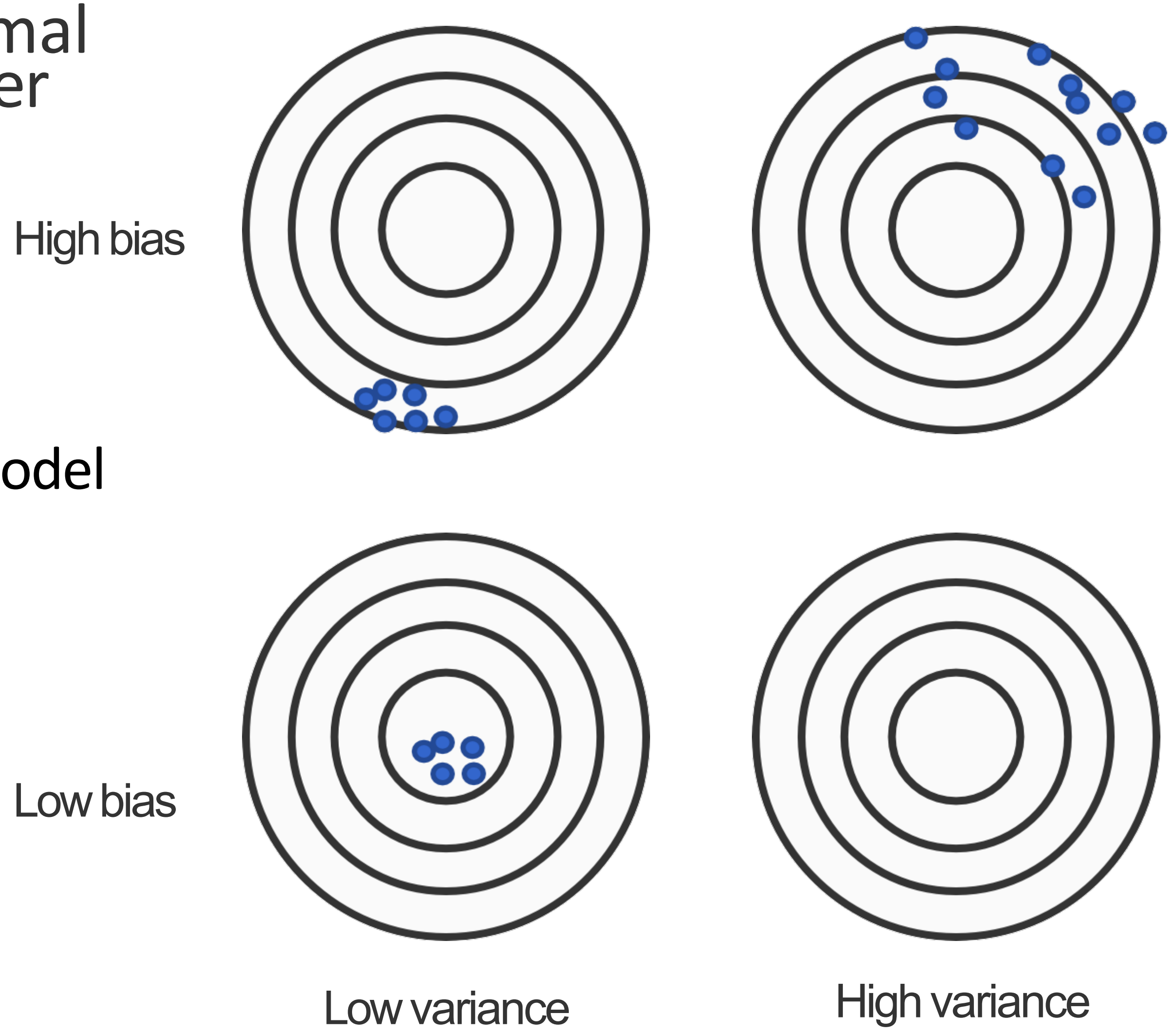
Each dot is a learnt model



Let's Play Darts

Suppose the optimal model is the center

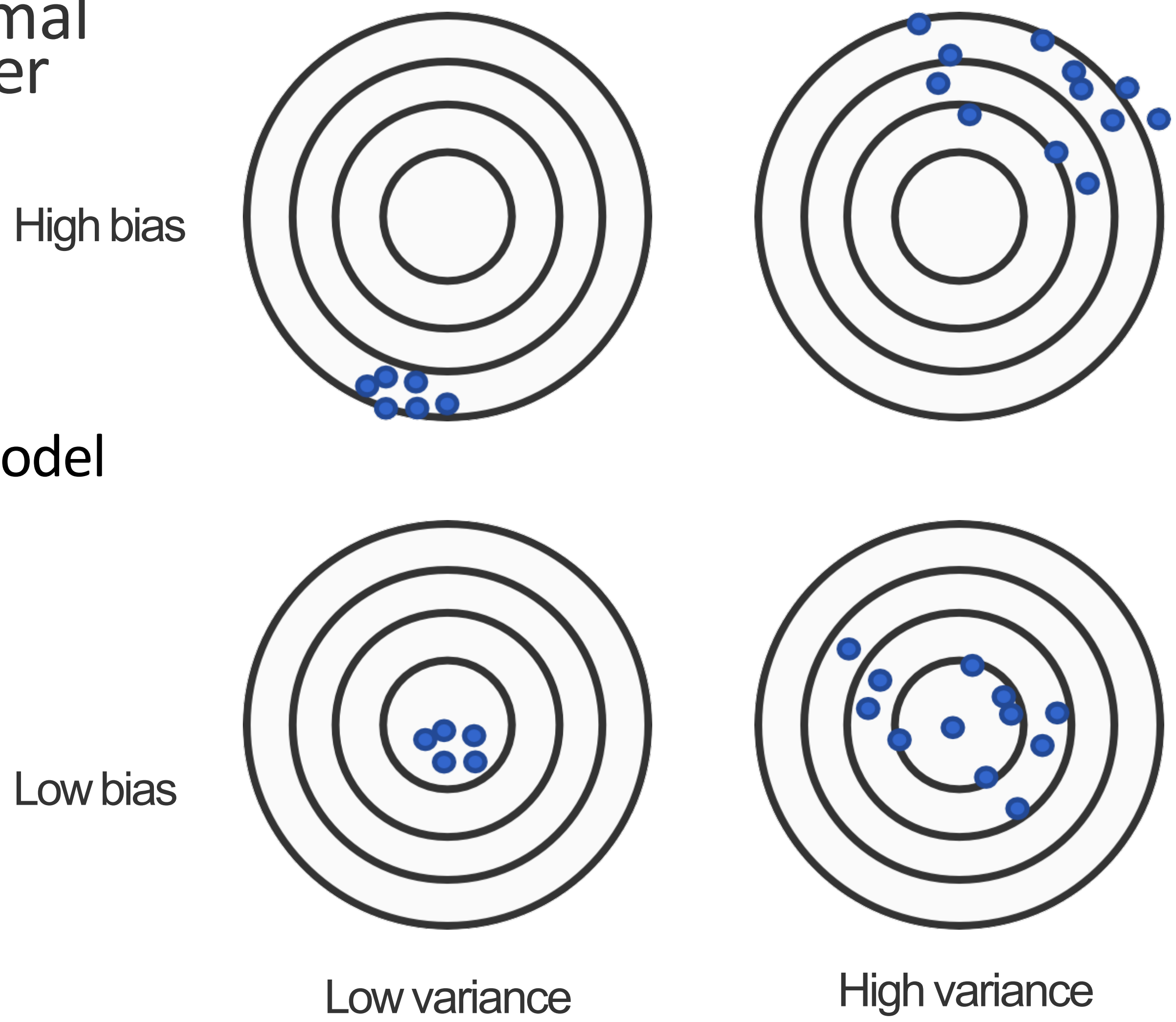
Each dot is a learnt model



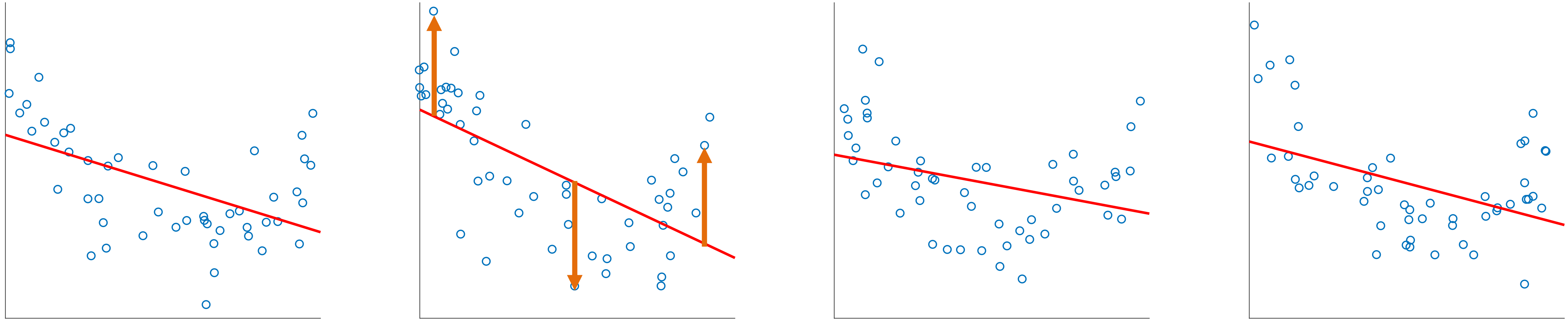
Let's Play Darts

Suppose the optimal model is the center

Each dot is a learnt model

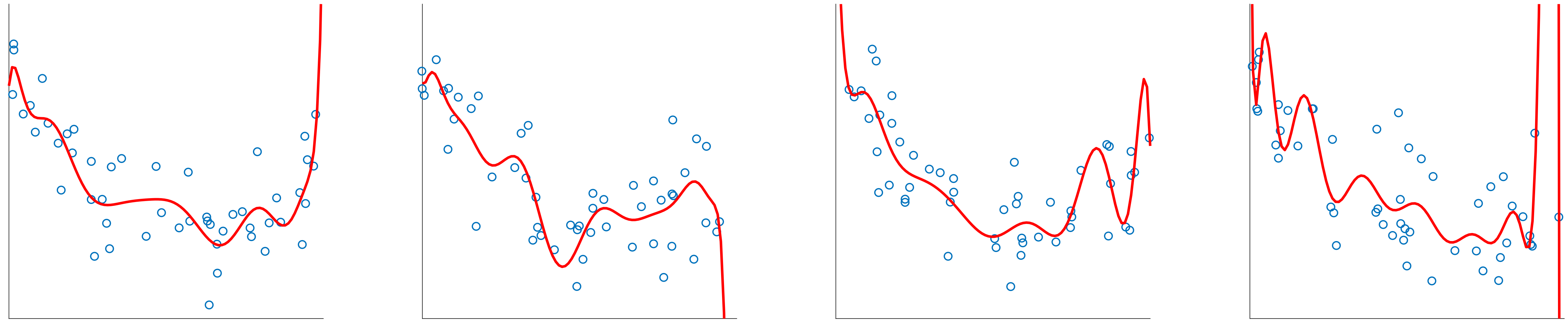


Bias in ML Models



Regardless of (size of) training sample, model will produce consistent (large) errors

Variance in ML Models



Different samples of training data yield different model fits

Formalizing Bias and Variance

Given data set $\boldsymbol{\mathcal{D}} = \{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_N}, y_N)\}$ ~ $p(\boldsymbol{\mathcal{D}})$

Model f built from the data set $\boldsymbol{\mathcal{D}}$

Prediction of a testing example $\boldsymbol{x}$ is given by $f(\boldsymbol{x}; \boldsymbol{\mathcal{D}})$

Expected mean squared error of a testing example ($\boldsymbol{x}$, y)

$$\mathrm{MSE}_{\boldsymbol{x}} = \boldsymbol{E}_{\boldsymbol{\mathcal{D}}}\left[\left(y - f(\boldsymbol{x}; \boldsymbol{\mathcal{D}})\right)^{\mathbf{2}}\right]$$

Formalizing Bias and Variance

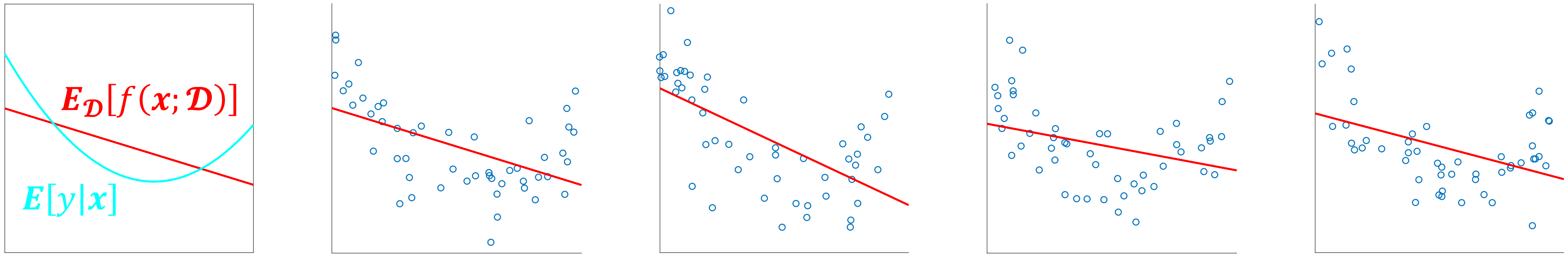
$$\mathrm{MSE}_{\boldsymbol{x}} = \boldsymbol{E}_{\mathcal{D}}\left[\left(y - f(\boldsymbol{x};\mathcal{D})\right)^2\right]$$

$$= \left(\boldsymbol{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})] - \boldsymbol{E}[y|\boldsymbol{x}]\right)^2$$

$$+ \boldsymbol{E}_{\mathcal{D}}\left[\left(f(\boldsymbol{x};\mathcal{D}) - \boldsymbol{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})]\right)^2\right]$$

$$+ \boldsymbol{E}\left[\left(y - \boldsymbol{E}[y|\boldsymbol{x}]\right)^2\right]$$

Bias: difference between average model prediction (across data sets) and the target



Formalizing Bias and Variance

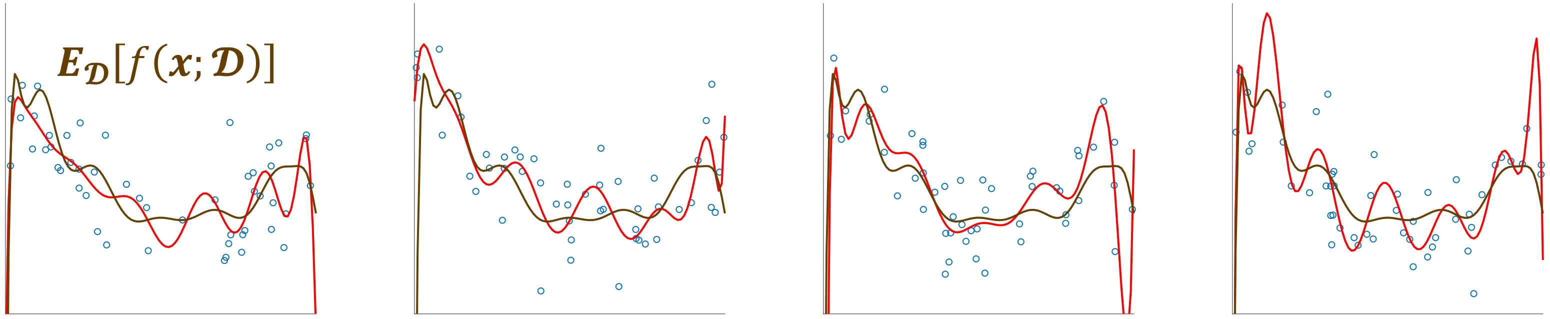
$$\mathrm{MSE}_{\boldsymbol{x}} = \boldsymbol{E}_{\mathcal{D}|\boldsymbol{x}}\left[\left(y - f(\boldsymbol{x};\mathcal{D})\right)^2\right]$$

$$= \left(\boldsymbol{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})] - \boldsymbol{E}[y|\boldsymbol{x}]\right)^2$$

$$+ \boldsymbol{E}_{\mathcal{D}}\left[\left(f(\boldsymbol{x};\mathcal{D}) - \boldsymbol{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})]\right)^2\right]$$

$$+ \boldsymbol{E}\left[\left(y - \boldsymbol{E}[y|\boldsymbol{x}]\right)^2\right]$$

Variance of models (across data sets) for a given point



Formalizing Bias and Variance

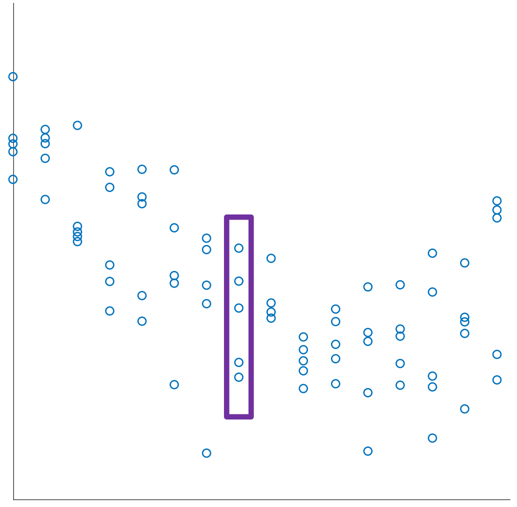
$$\mathrm{MSE}_{\boldsymbol{x}} = \boldsymbol{E}_{\mathcal{D}|\boldsymbol{x}}\left[\left(y - f(\boldsymbol{x};\mathcal{D})\right)^2\right]$$

$$= \left(\boldsymbol{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})] - \boldsymbol{E}[y|\boldsymbol{x}]\right)^2$$

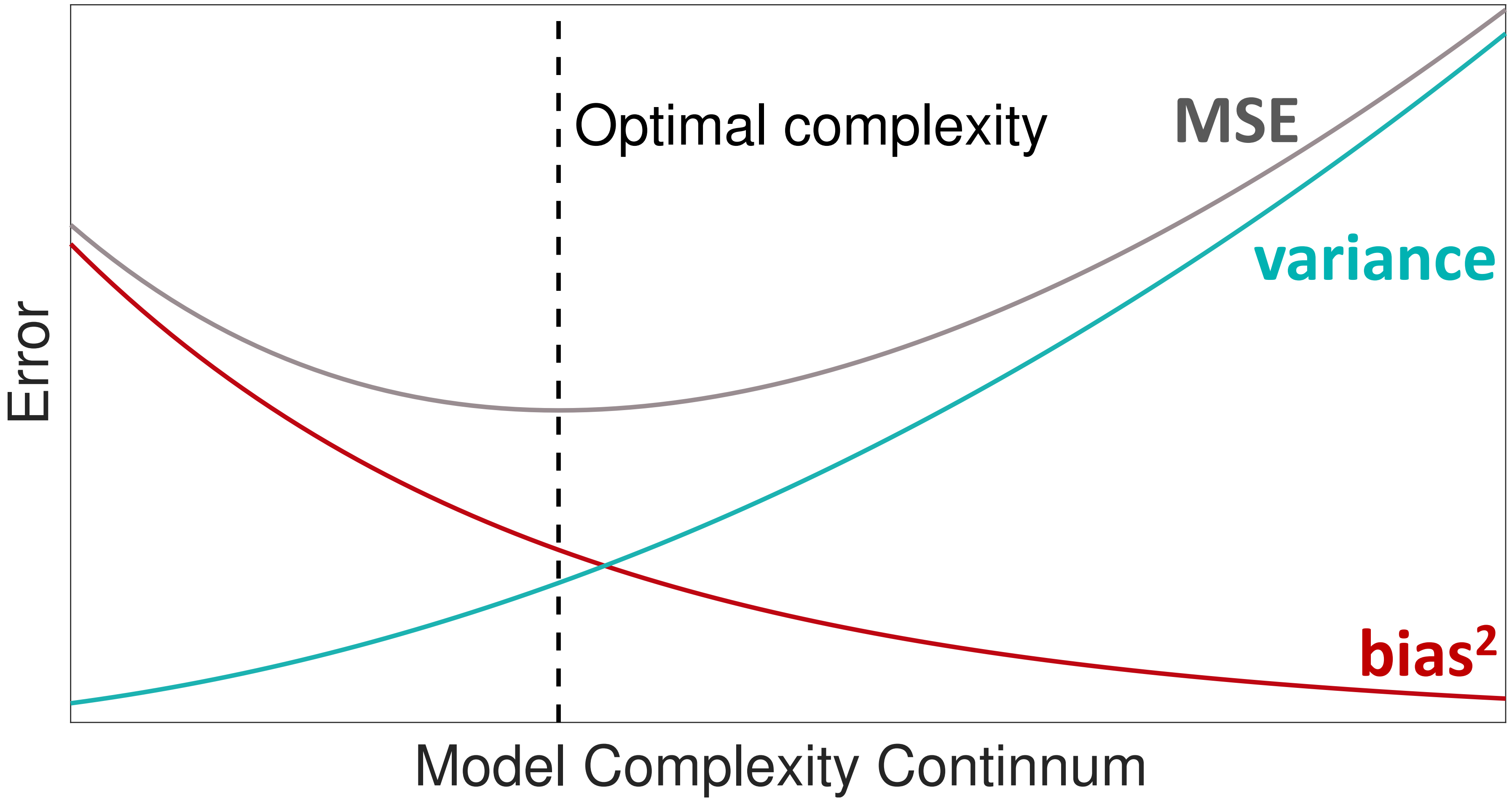
$$+ \boldsymbol{E}_{\mathcal{D}}\left[\left(f(\boldsymbol{x};\mathcal{D}) - \boldsymbol{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})]\right)^2\right]$$

$$+ \boldsymbol{E}\left[\left(y - \boldsymbol{E}[y|\boldsymbol{x}]\right)^2\right]$$

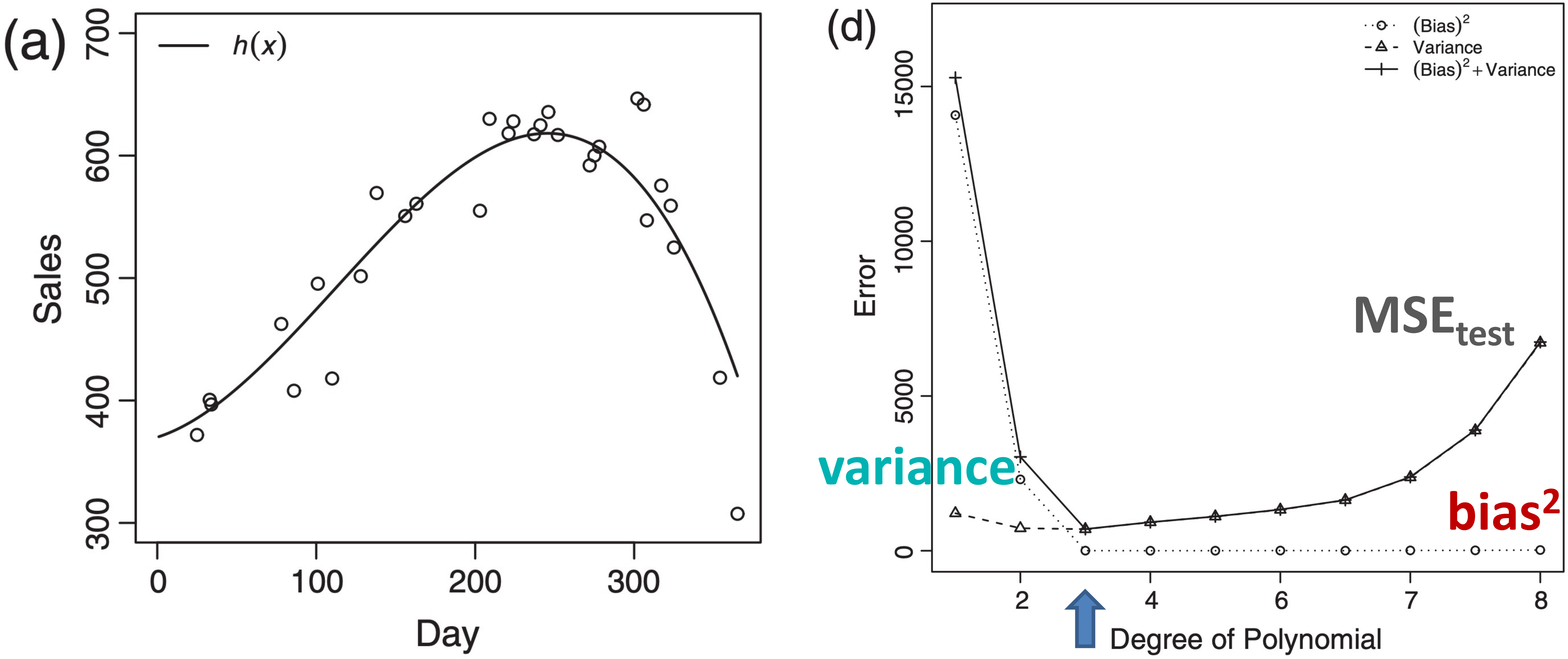
intrinsic noise in data set



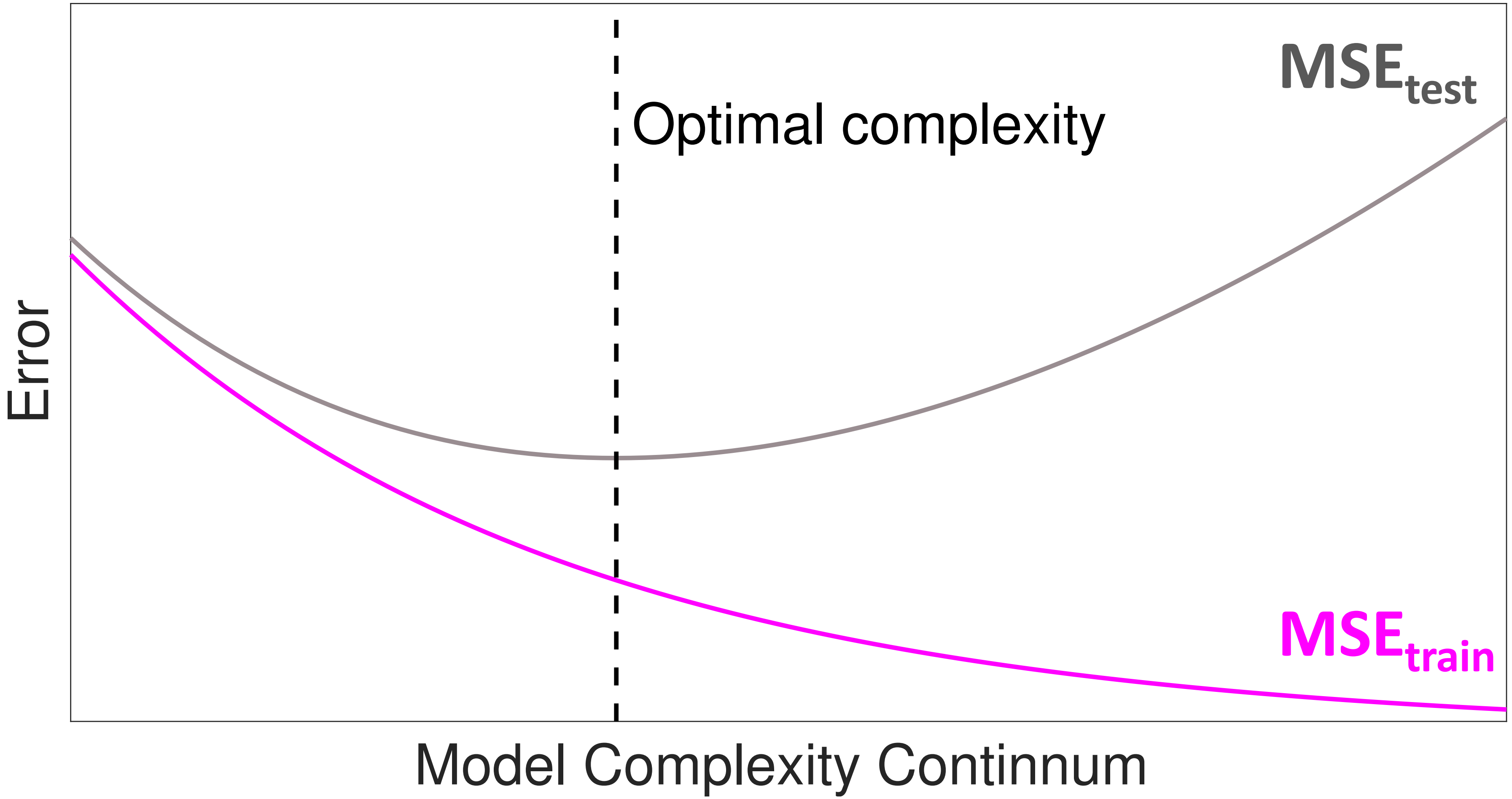
Bias-Variance Tradeoff



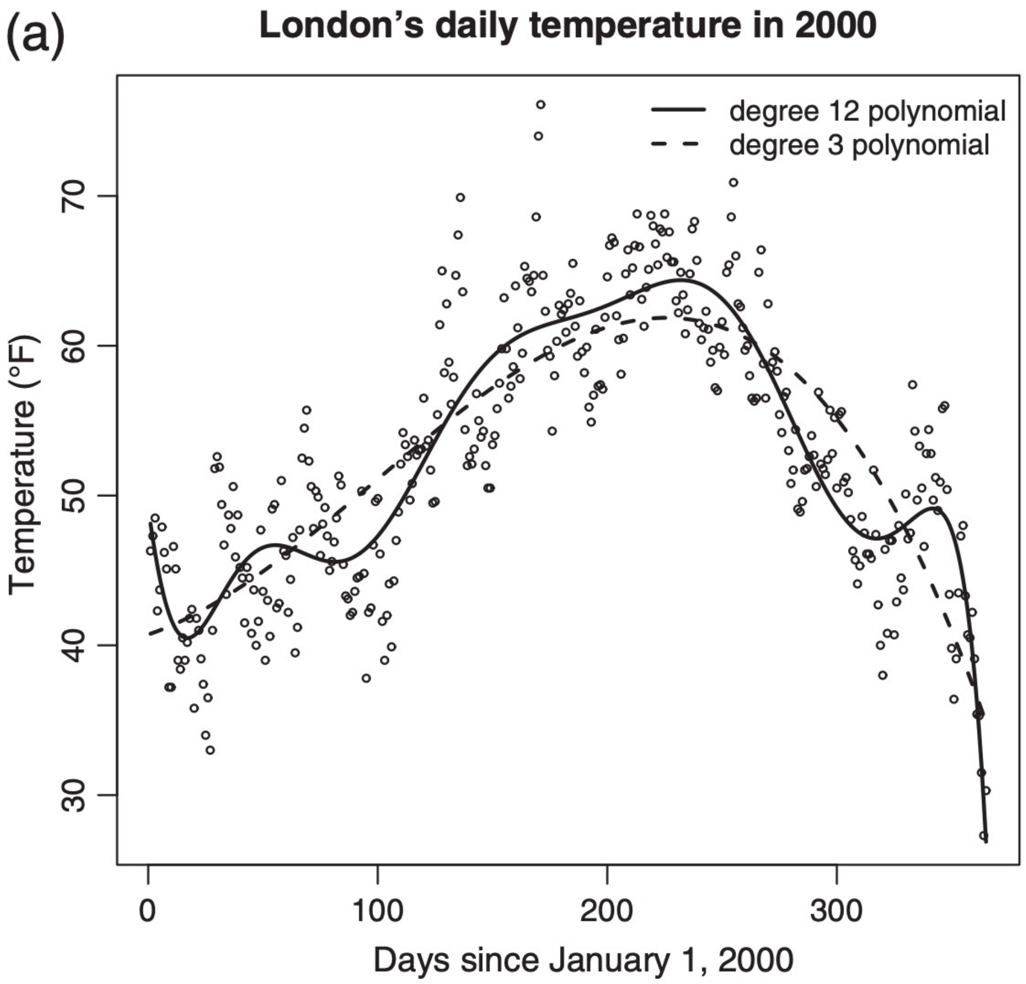
Bias-Variance Tradeoff

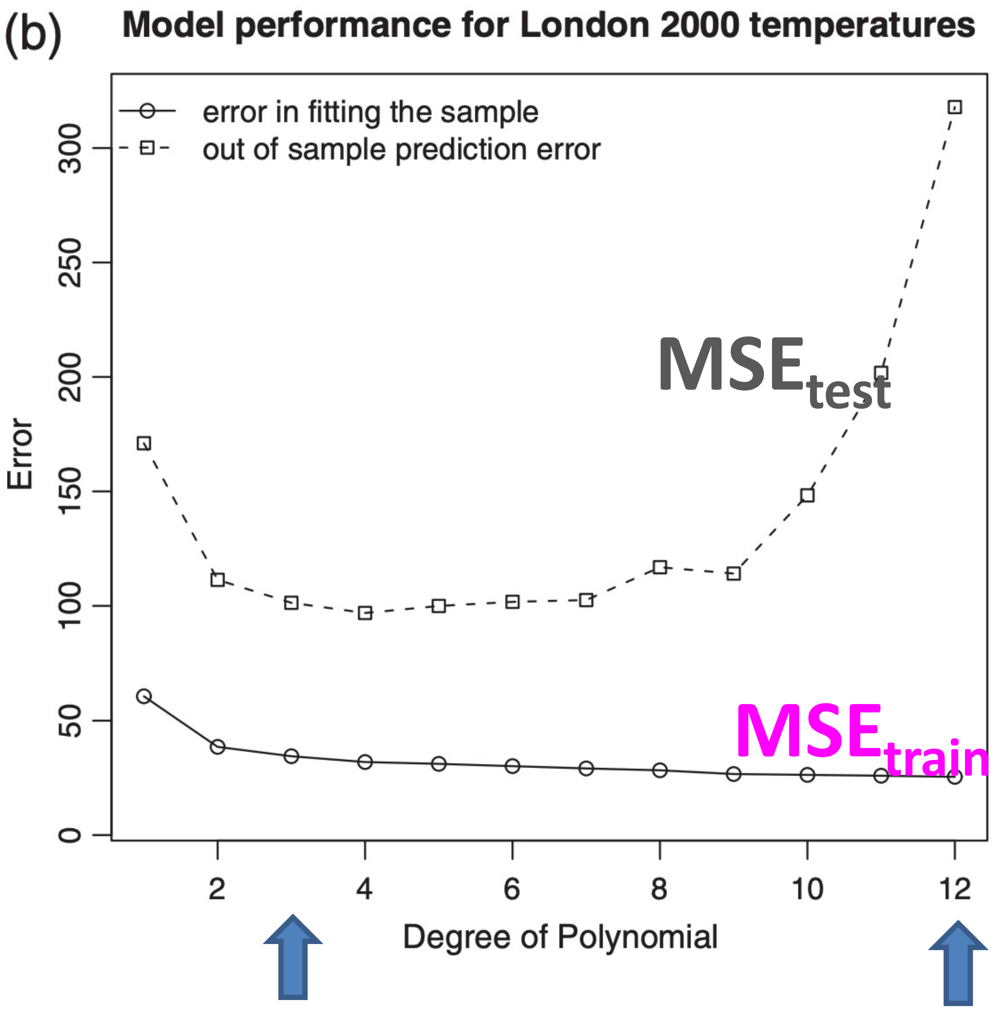


Bias-Variance Tradeoff Is Revealed Via Test Set Not Training Set



Bias-Variance Tradeoff Is Revealed Via Test Set Not Training Set





Summary: Bias-Variance Tradeoff

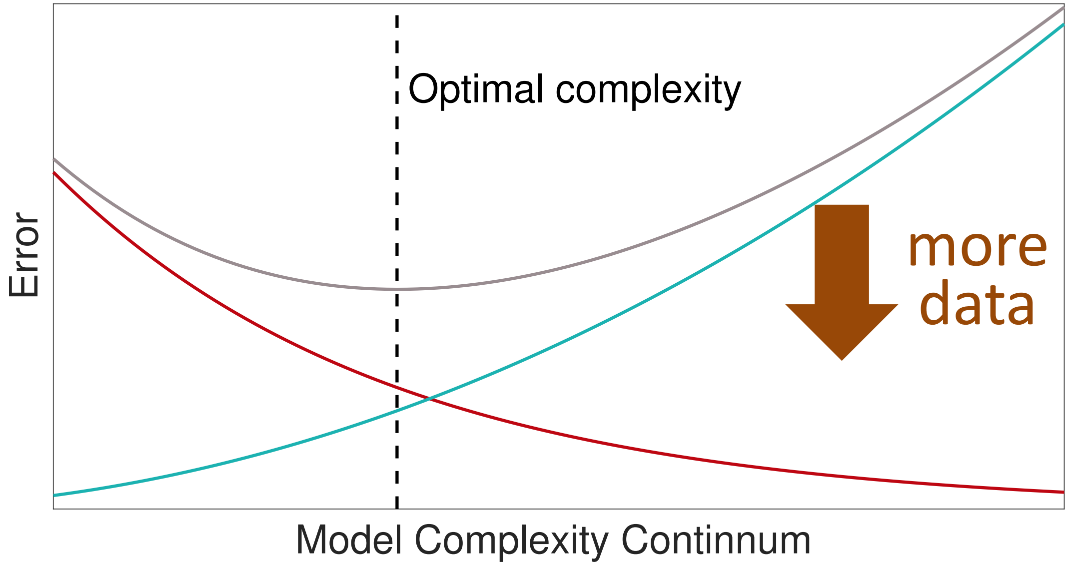
- Error = bias^2 + variance (+ noise)

- High bias $\rightarrow$ both training and test error can be high
  - Arises when the classifier underfits/cannot represent the data

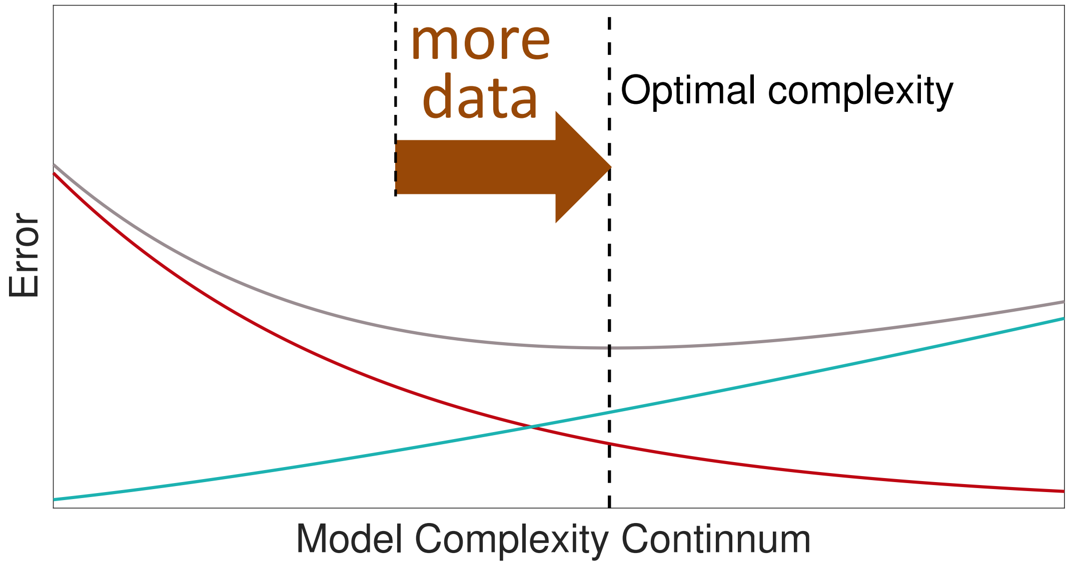
- High variance $\rightarrow$ training error can be low, but test error will be high
  - Arises when the classifier overfits the training set/is very powerful

Bias-Variance Tradeoff Control

- Reduce bias
  - Higher polynomial degrees
  - Deeper models
    - Deep neural nets/decision trees, etc
  - Smaller k in k-nearest neighbors



- Reduce variance
  - Ensemble methods (bagging, boosting)
  - Stronger regularization
    - L1, L2 regularization
  - Larger k in k-nearest neighbors
  - **More training data**



Summary

- K-Nearest Neighbor Classifier
  - No training
  - Time complexity O(Nd), independent of number of classes
  - Efficient algorithms: Vector Quantization(VQ)

- Hyperparameter Tuning
  - K-fold cross validation
  - Split training set into training set + validation set
  - Learn hyperparameter via validation set (**NOT** test set)

- Bias-Variance Tradeoff
  - Test/Generalization error = bias^2 + variance (+ Noise)
  - Simple model: large bias, small variance
  - Complex model: small bias, large variance

Acknowledgement

Some slides are from

Shusen Wang

https://github.com/wangshusen/DeepLearning

Mike Mozer

University of Colorado at Boulder