10-701: Introduction to Machine Learning

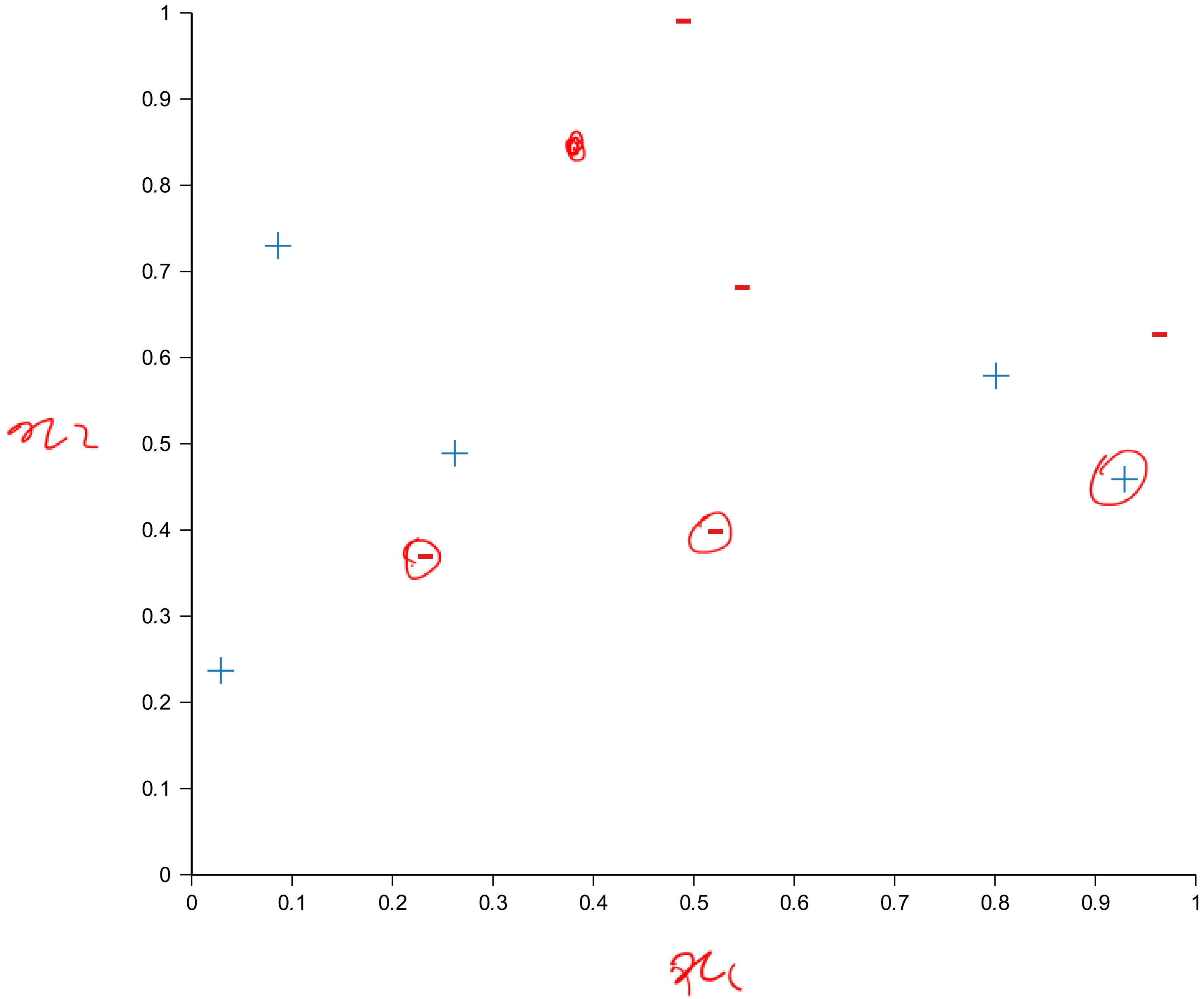
Lecture 4 – Linear Regression

Hoda Heidari

9/3/2025

* Slides adopted from F24 offering of 10701 by Henry Chai.

Nearest Neighbor: Example



Generalization of Nearest Neighbor (Cover and Hart, 1967)

- Claim: under certain conditions, as $n \rightarrow \infty$, with high probability, the true error rate of the nearest neighbor model $\leq 2 \ *$ the Bayes error rate (the optimal classifier)
- Proof (cont.):
- $err(h) = \mathbb{E}_{x'}[\mathbb{1}(h(\boldsymbol{x}') \neq y')] = P\{h(\boldsymbol{x}') \neq y'\}$

$$= P\{h(\boldsymbol{x}') = 1, y' = 0\} + P\{h(\boldsymbol{x}') = 0, y' = 1\}$$

$$= \pi\left(\boldsymbol{x}^{\left(\hat{\imath}(\boldsymbol{x}')\right)}\right)(1 - \pi(\boldsymbol{x}')) + \left(1 - \pi\left(\boldsymbol{x}^{\left(\hat{\imath}(\boldsymbol{x}')\right)}\right)\right)\pi(\boldsymbol{x}')$$

$$\rightarrow \pi(\boldsymbol{x}')(1 - \pi(\boldsymbol{x}')) + (1 - \pi(\boldsymbol{x}'))\pi(\boldsymbol{x}')$$

$$= 2\pi(\boldsymbol{x}')(1 - \pi(\boldsymbol{x}'))$$

$$\leq 2\min\Big(\pi(\boldsymbol{x}'), (1 - \pi(\boldsymbol{x}'))\Big) = 2err(h^*) \ \blacksquare$$

Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964

k-Nearest Neighbors (kNN)

- Why limit ourselves to just one neighbor?
- Classify a point as the most common label among the labels of the k nearest training points
- Tie-breaking (in case of even k and/or more than 2 classes)

k-Nearest Neighbors (kNN)

- Why limit ourselves to just one neighbor?
- Classify a point as the most common label among the labels of the k nearest training points
- Tie-breaking (in case of even k and/or more than 2 classes)
  - Weight votes by distance
  - Remove furthest neighbor
  - Add next closest neighbor
  - Use a different distance metric

kNN on Fisher Iris Data

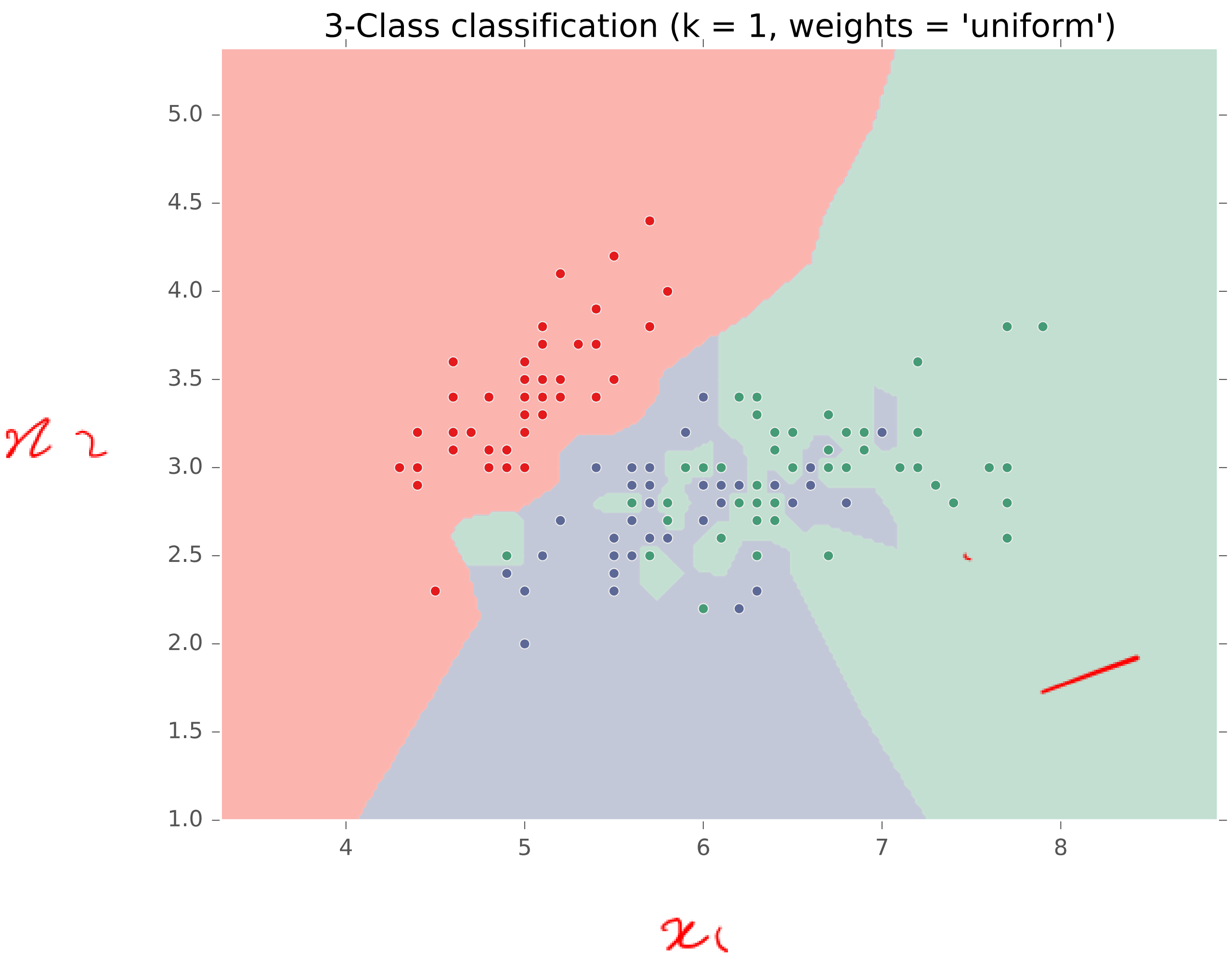


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

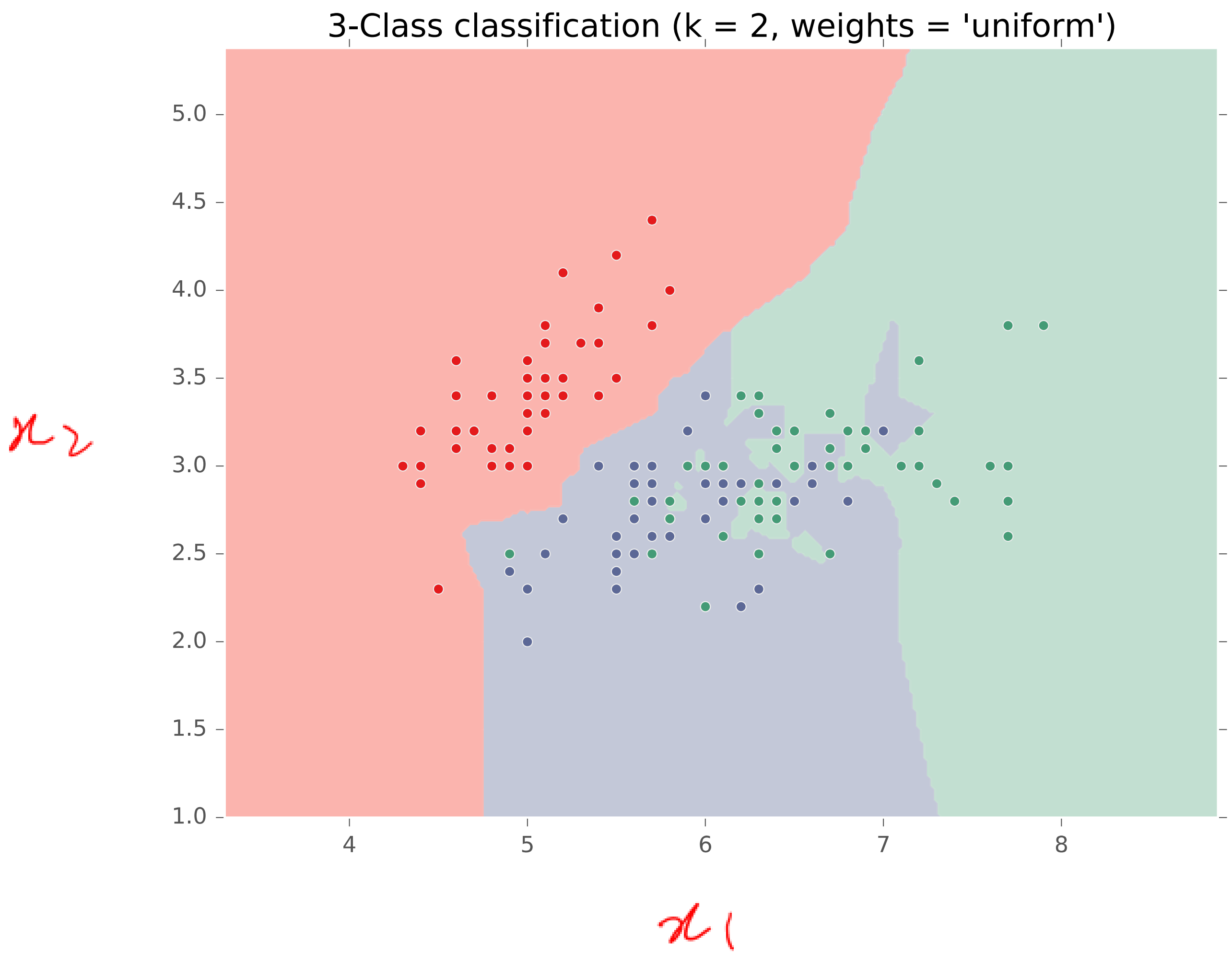


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

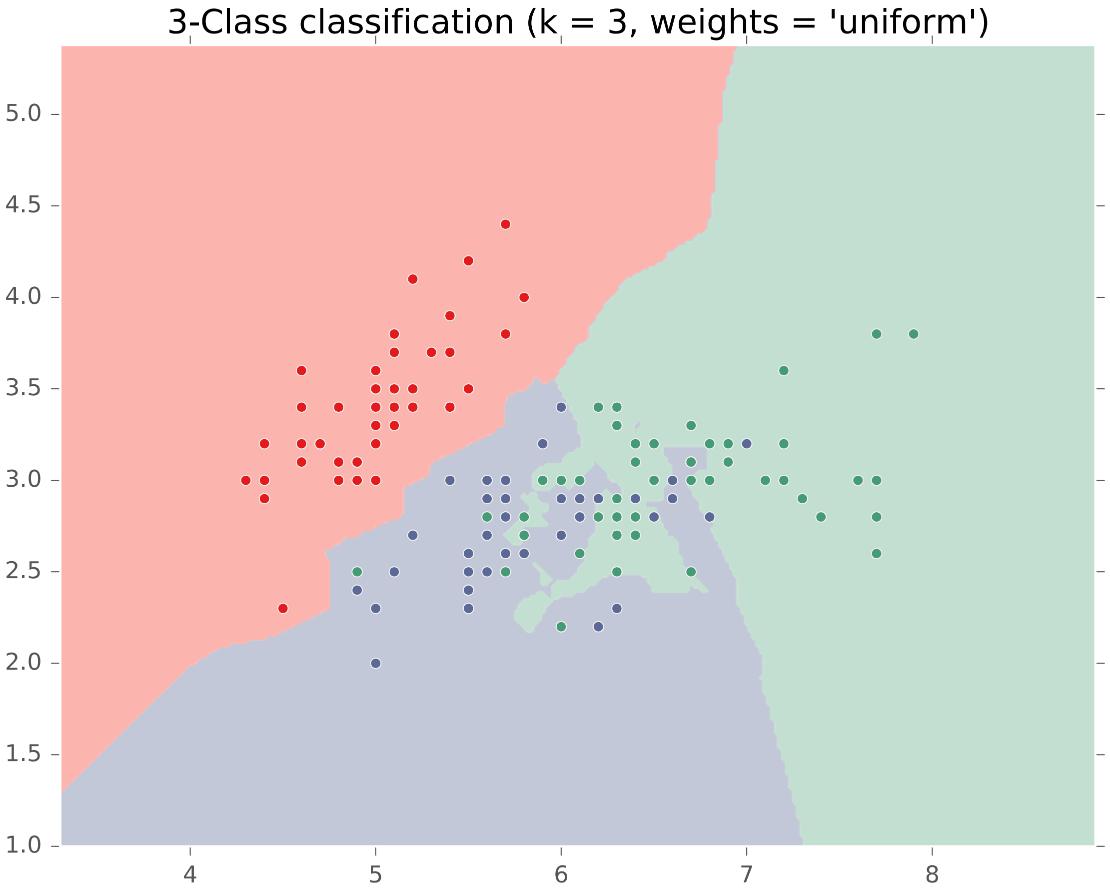


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

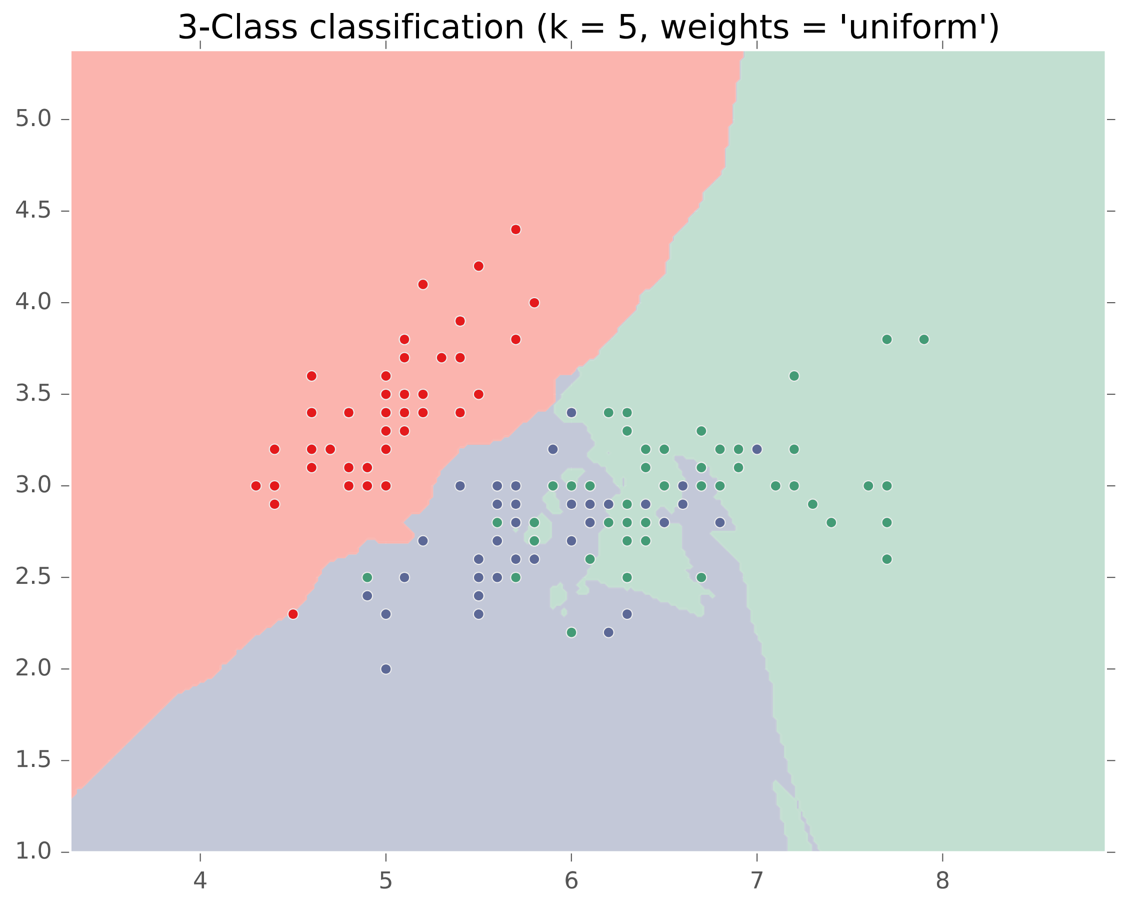


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

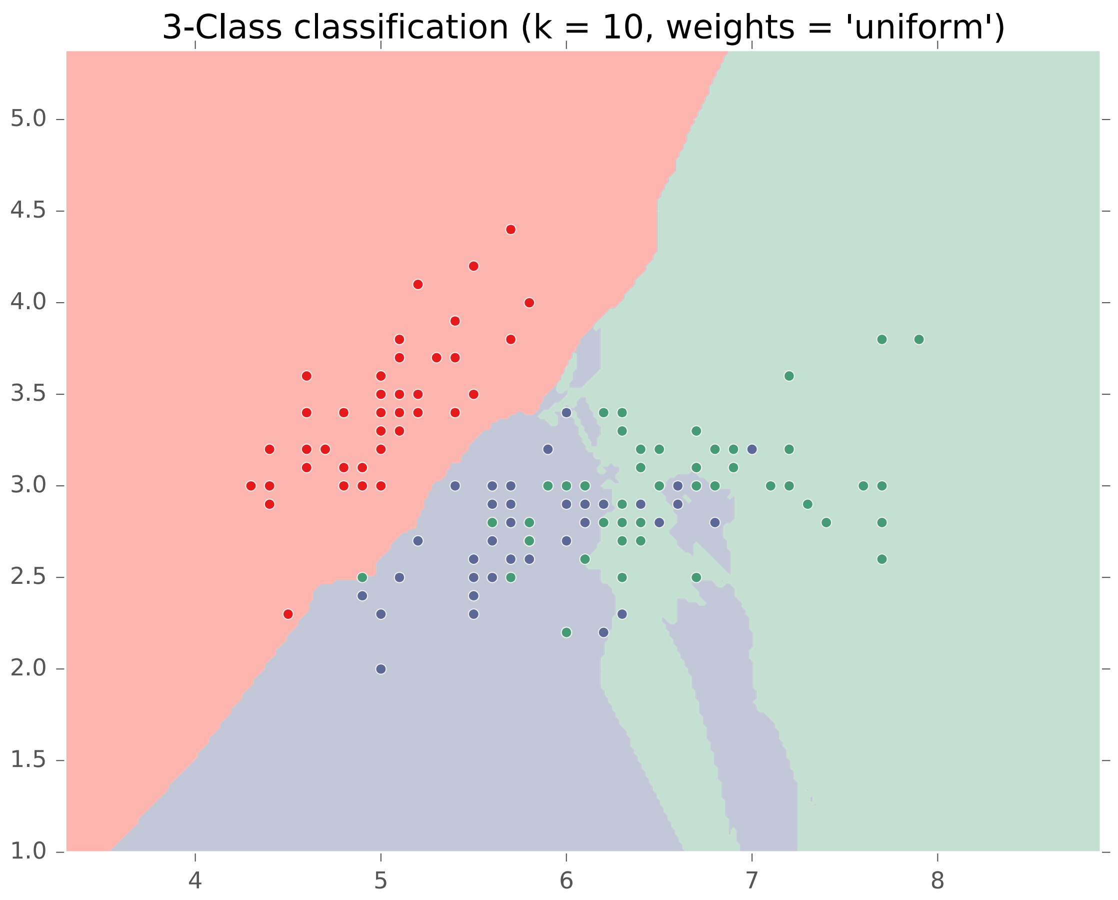


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

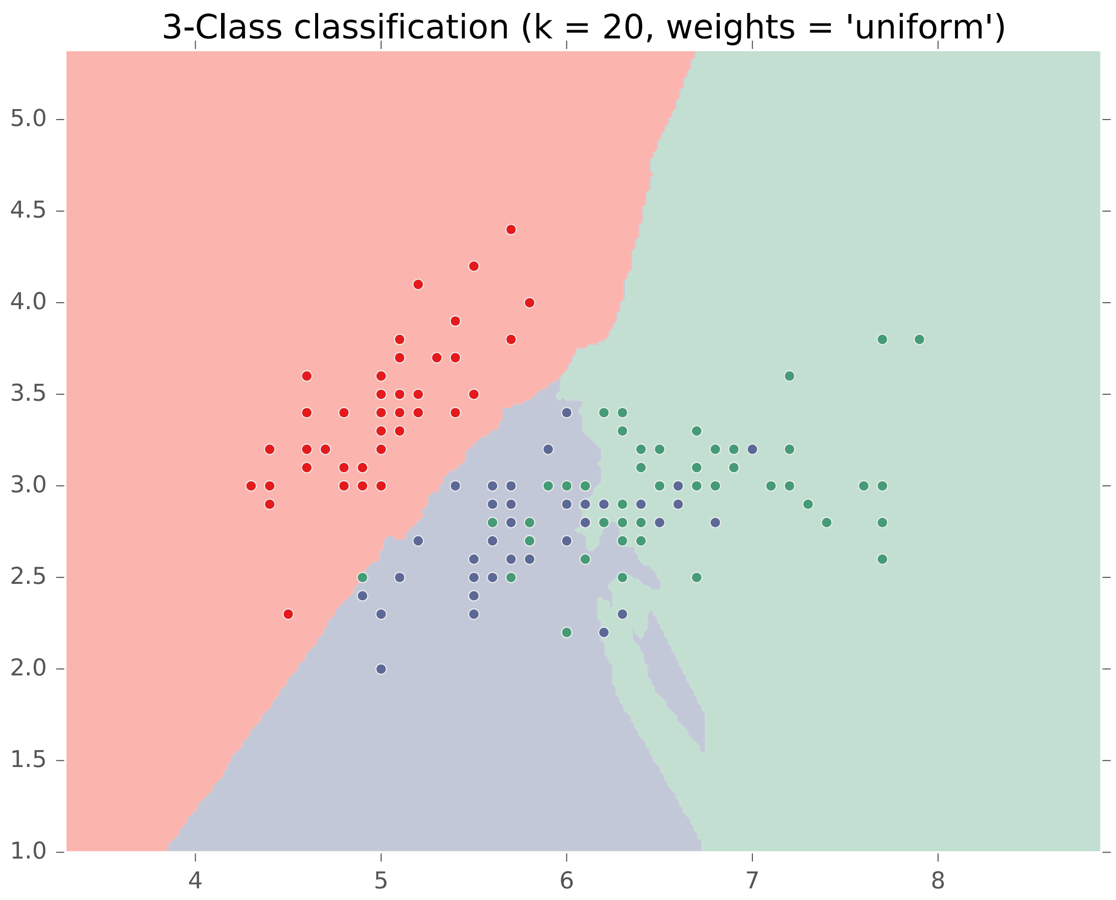


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

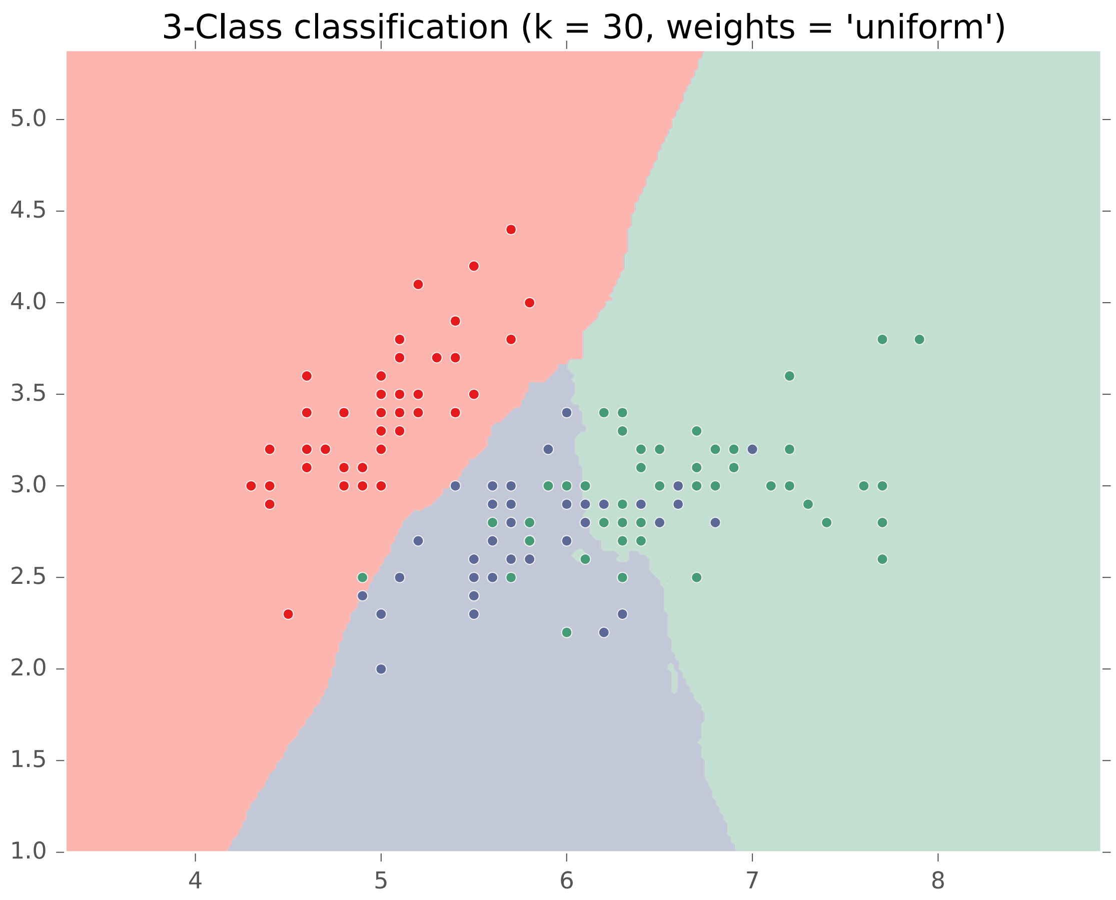


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

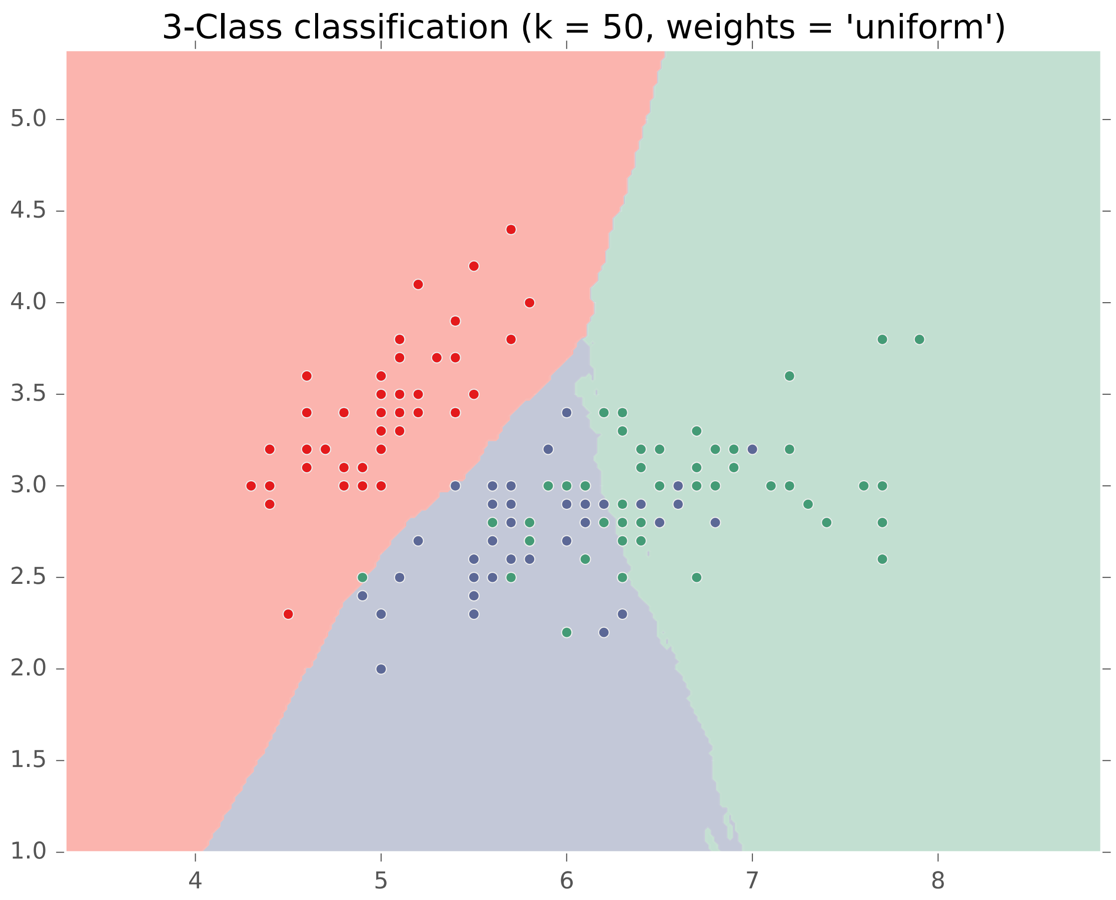


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

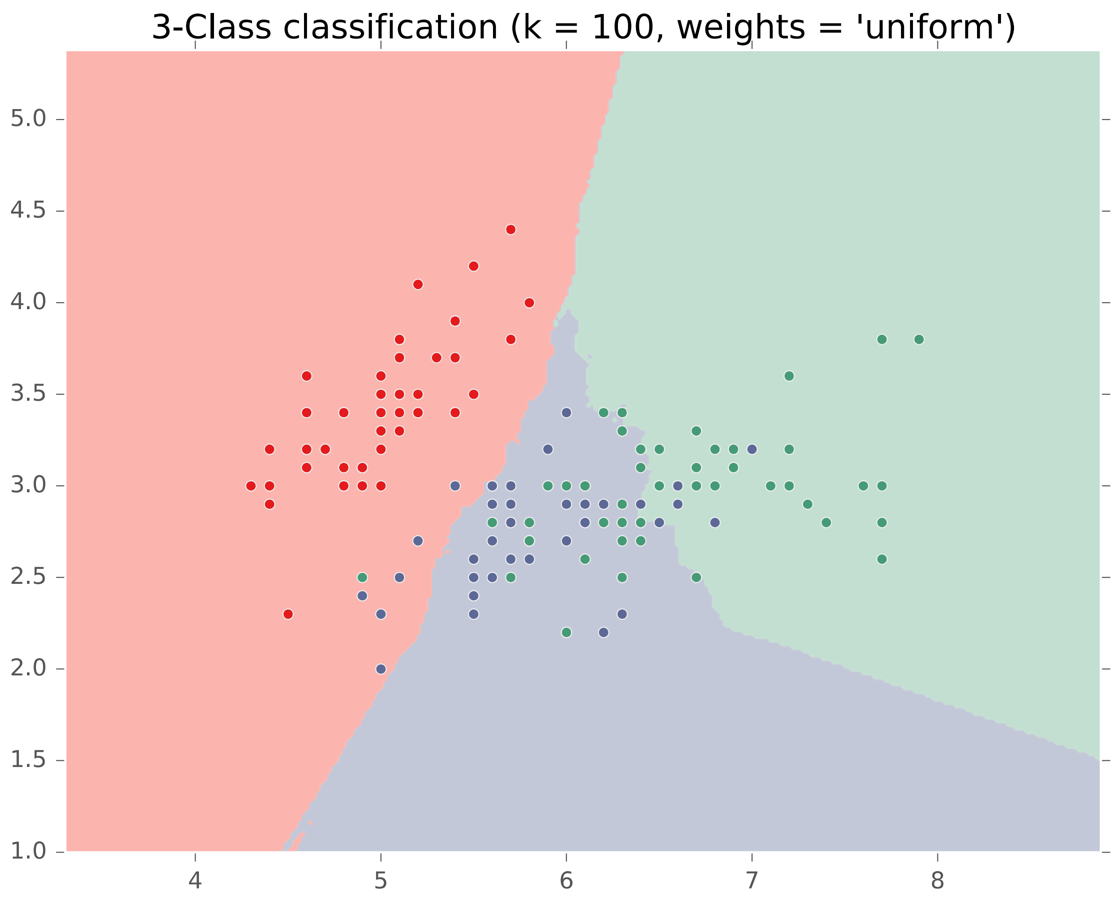


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

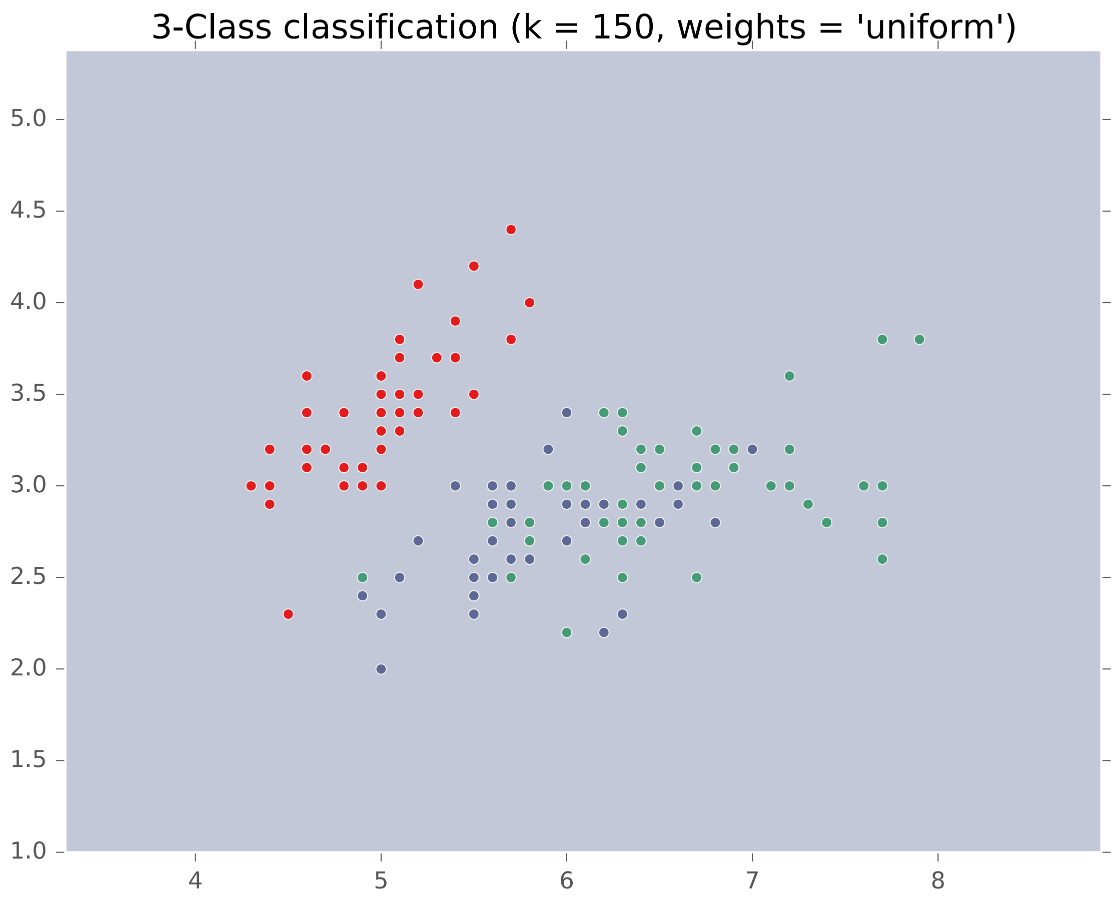


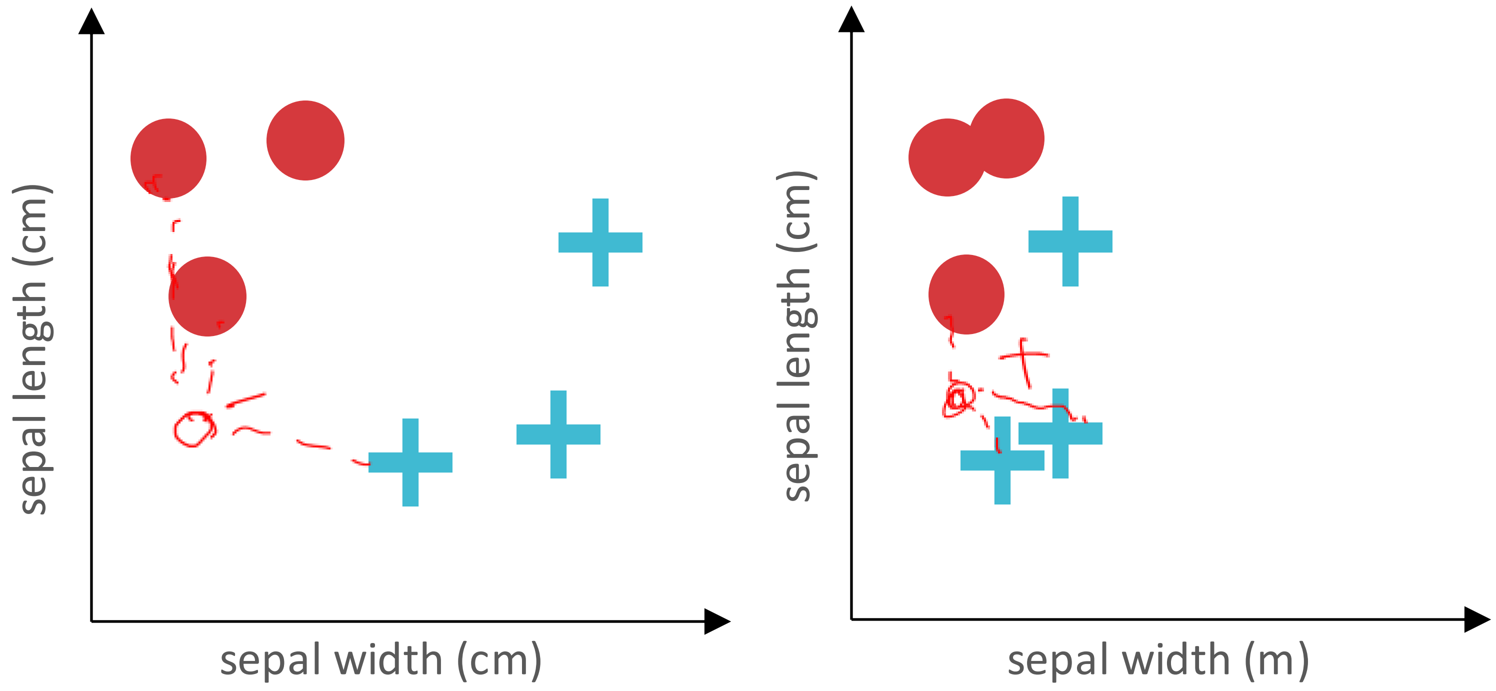
Figure courtesy of Matt Gormley

kNN: Inductive Bias

- What is the inductive bias of a kNN model that uses the Euclidean distance metric?

kNN: Inductive Bias

- What is the inductive bias of a kNN model that uses the Euclidean distance metric?
- Similar points should have similar labels and *all features are equivalently important for determining similarity*



- Feature scale can dramatically influence results!

Figure courtesy of Matt Gormley

Setting k

- When $k = 1$:
  - many, complicated decision boundaries
  - may ***overfit***
- When $k = N$:
  - no decision boundaries; always predicts the most common label in the training data
  - may ***underfit***
- k controls the complexity of the hypothesis space $\Longrightarrow k$ affects how well the learned model will generalize

Setting k

- Theorem:
  - If k is some function of N s.t. $k(N) \to \infty$ and $\frac{k(N)}{N} \to 0$ as $N \to \infty$ …
  - … then (under certain assumptions) the true error of a kNN model $\to$ the Bayes error rate
- Heuristics:
  - $k = \lfloor \sqrt{N} \rfloor$
- This is fundamentally a question of **model selection**: each value of k corresponds to a different model/hypothesis class.

Model Selection

- A **model or hypothesis class** is a (typically infinite) set of classifiers that a learning algorithm searches through to find the best one
- **Model parameters** are the numeric values or structure that are selected by the learning algorithm
- **Hyperparameters** are the tunable aspects of the model that are not selected by the learning algorithm

Example: Decision Trees

- **Model/hypothesis class** = set of all possible trees, potentially narrowed down according to the hyperparameters (e.g., max depth)
- **Model parameters** = structure of a specific tree e.g., splits, split order, predictions at leaf nodes, …
- **Hyperparameters** = splitting criterion, max-depth, tie-breaking procedures, etc…

Model Selection

- A **model or hypothesis class** is a (typically infinite) set of classifiers that a learning algorithm searches through to find the best one
- **Model parameters** are the numeric values or structure that are selected by the learning algorithm
- **Hyperparameters** are the tunable aspects of the model that are not selected by the learning algorithm

Example: kNN

- **Model/hypothesis class** = set of all possible nearest neighbors classifiers
- **Model parameters** = none! kNN is a "non-parametric model"
- **Hyperparameters** = k

Model Selection with Test Sets

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$, suppose we have multiple candidate model/hypothesis spaces:

$$\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$$

- Learn a classifier from each space using only $\mathcal{D}_{train}$:

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \ldots, h_M \in \mathcal{H}_M$$

- Evaluate each one using $\mathcal{D}_{test}$ and choose the one with lowest test error:

$$\hat{m} = \underset{m \in \{1,\ldots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{test})$$

Model Selection with Test Sets?

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$, suppose we have multiple candidate <u>model/hypothesis spaces</u>:

$$\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_M$$

- Learn a classifier from each space using only $\mathcal{D}_{train}$:

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \dots, h_M \in \mathcal{H}_M$$

- Evaluate each one using $\mathcal{D}_{test}$ and choose the one with lowest test error:

$$\hat{m} = \operatorname*{argmin}_{m \in \{1,\dots,M\}} err(h_m, \mathcal{D}_{test})$$

- Is $err(h_{\hat{m}}, \mathcal{D}_{test})$ a good estimate of $err(h_{\hat{m}})$?

Model Selection with Validation Sets

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$, suppose we have multiple candidate model/hypothesis spaces: :

$$\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$$

- Learn a classifier from each space using only $\mathcal{D}_{train}$:

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \ldots, h_M \in \mathcal{H}_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\ldots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

- Now $err(h_{\hat{m}}, \mathcal{D}_{test})$ is a good estimate of $err(h_{\hat{m}})$!

Hyperparameter Optimization with Validation Sets

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$, suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only $\mathcal{D}_{train}$:

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\dots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

- Now $err(h_{\hat{m}}, \mathcal{D}_{test})$ is a good estimate of $err(h_{\hat{m}})$!

Pro tip: train your final model using *both* training and validation datasets

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$, suppose we have multiple candidate <u>hyperparameter settings</u>:

$$\theta_1, \theta_2, \ldots, \theta_M$$

- Learn a classifier for each setting using only $\mathcal{D}_{train}$:

$$h_1, h_2, \ldots, h_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\ldots,M\}}{\operatorname{argmin}} \; err(h_m, \boxed{\mathcal{D}_{val}})$$

- Train a new model on $\boxed{\mathcal{D}_{train} \cup \mathcal{D}_{val}}$ using $\theta_{\hat{m}}$, $h_{\hat{m}}^{+}$
- $err(h_{\hat{m}}^{+}, \mathcal{D}_{test})$ is still a good estimate of $err(h_{\hat{m}}^{+})$!

Setting k for kNN with Validation Sets

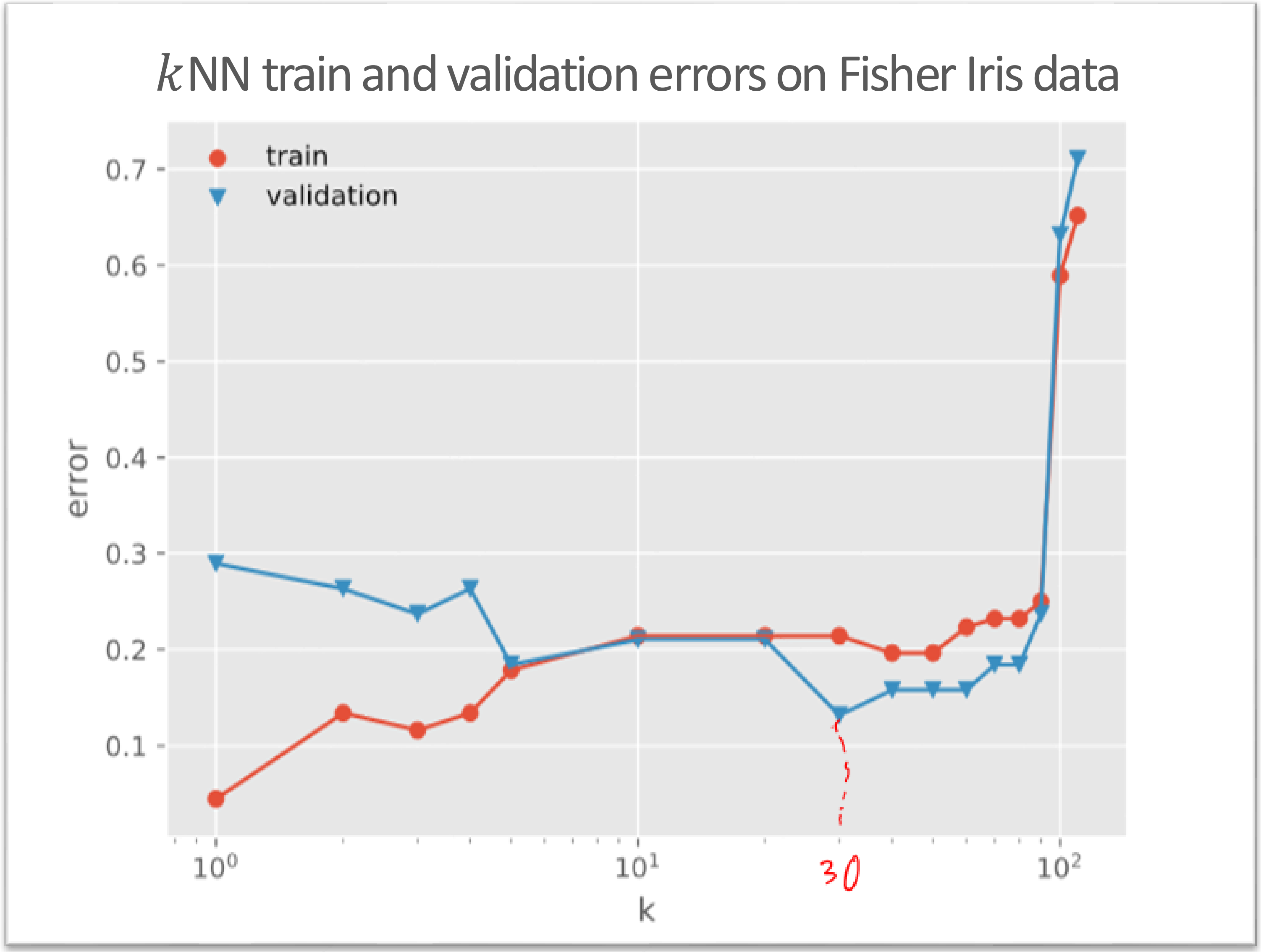


Figure courtesy of Matt Gormley

How should we partition our dataset?

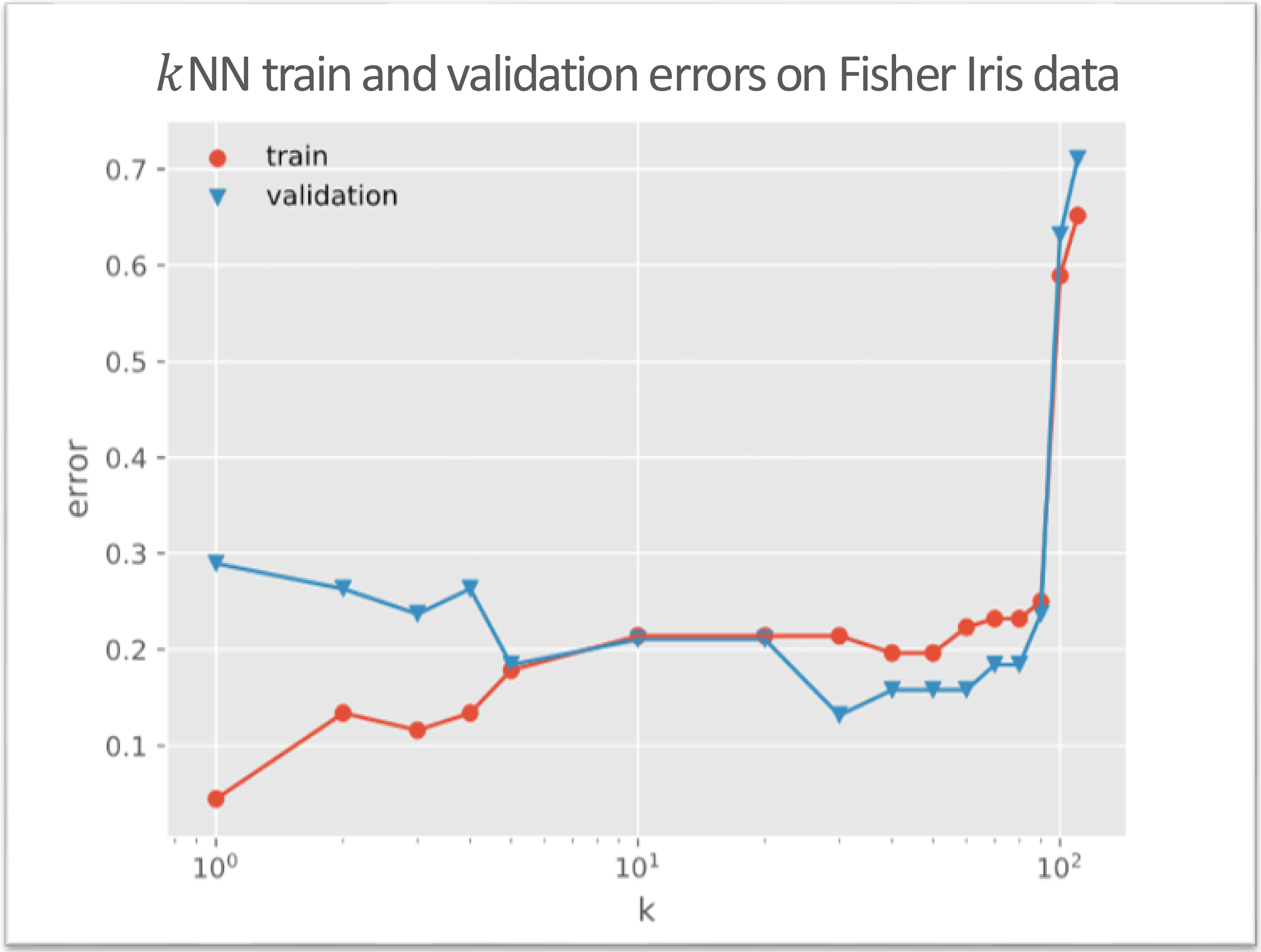
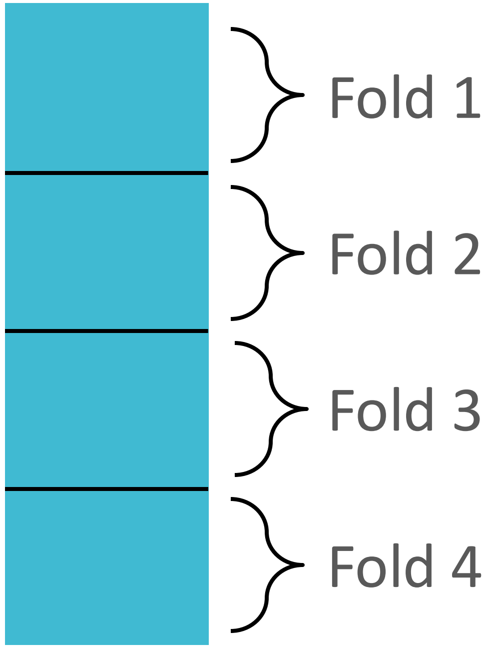


Figure courtesy of Matt Gormley

remaining data when D_{test} is set aside.

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:

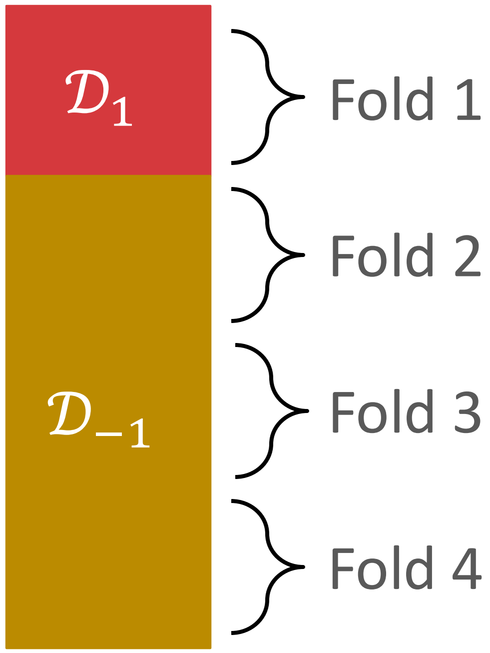


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$
- Use each one as a validation set once:

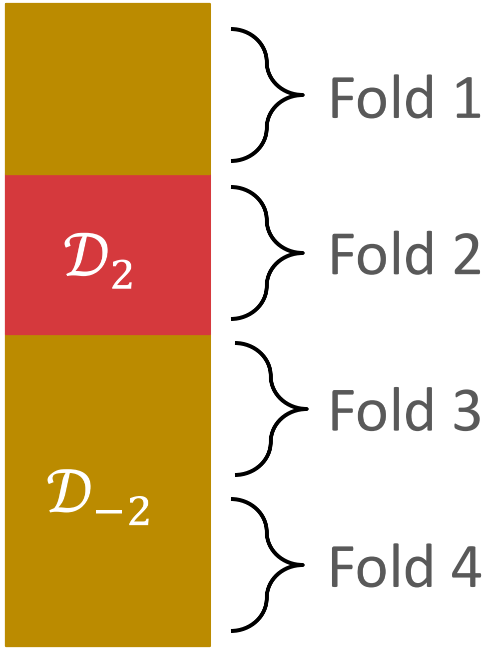


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:

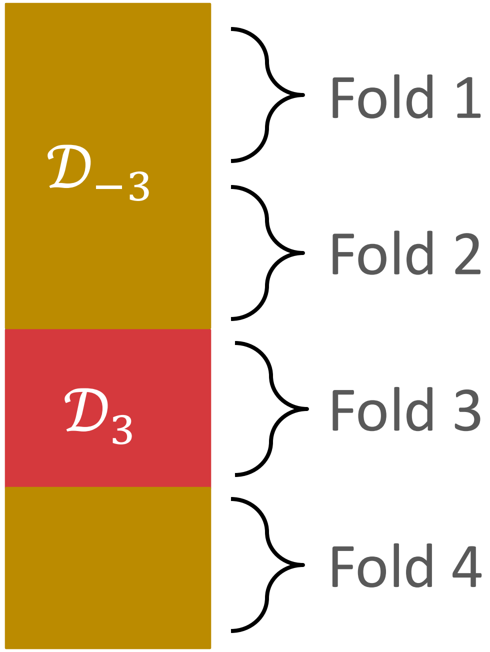


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$
- Use each one as a validation set once:

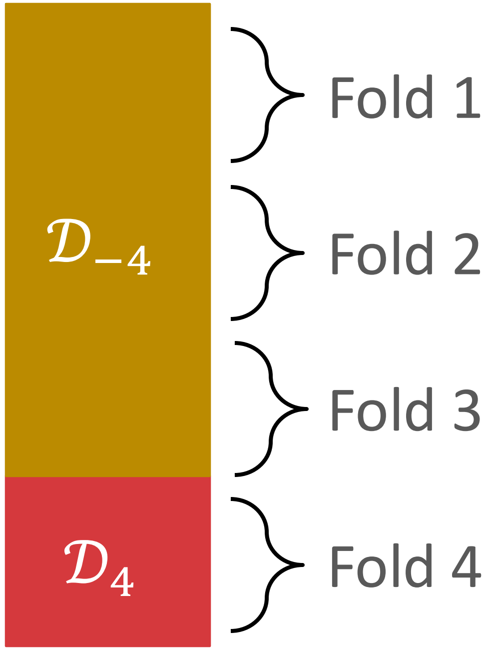


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:



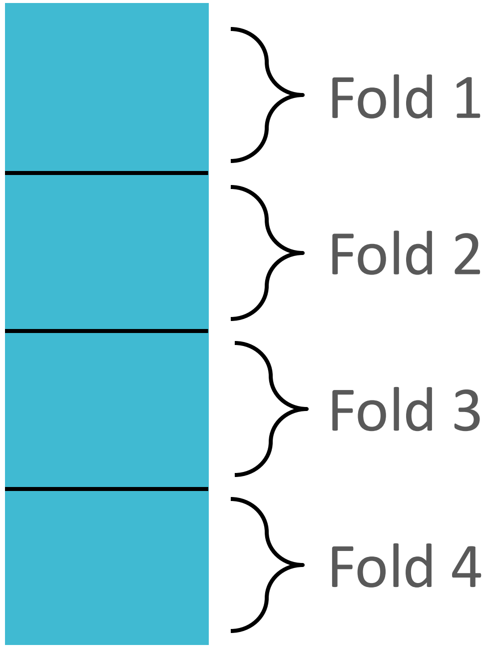
- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$

- Use each one as a validation set once:



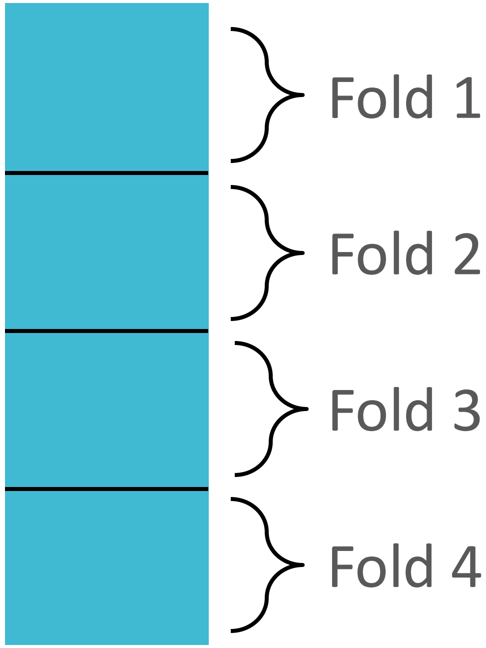
- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

- Special case when $K = N$: Leave-one-out cross-validation.

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:



- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$\underline{err_{cv_K}} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

- Choosing between m candidates requires training $\cancel{K}$ times.

mK

Summary

	Input	Output
Training	• training dataset • hyperparameters	• best model parameters
Hyperparameter Optimization	• training dataset • validation dataset	• best hyperparameters
Cross-Validation	• training dataset • validation dataset	• cross-validation error
Testing	• test dataset • classifier	• test error

Key Takeaways

- Real-valued features and decision boundaries
- Nearest neighbor model and generalization guarantees
- kNN "training" and prediction
- Effect of k on model complexity
- kNN inductive bias
- Differences between training, validation and test datasets in the model selection process
- Cross-validation for model selection
- Relationship between training, hyperparameter optimization and model selection

Recall: Regression

- Learning to diagnose heart disease

  as a **(supervised)** **regression task**

features / targets

data points

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	\$0
No	Medium	Normal	\$20
No	Low	Abnormal	\$30
Yes	Medium	Normal	\$100
Yes	High	Abnormal	\$5000

Decision Tree Regression

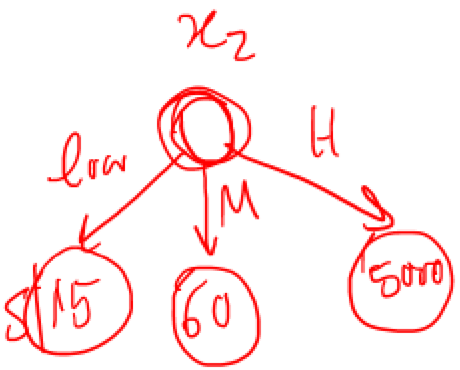
- Learning to diagnose heart disease

  as a **(supervised)** **<u>regression</u> task**

features			targets

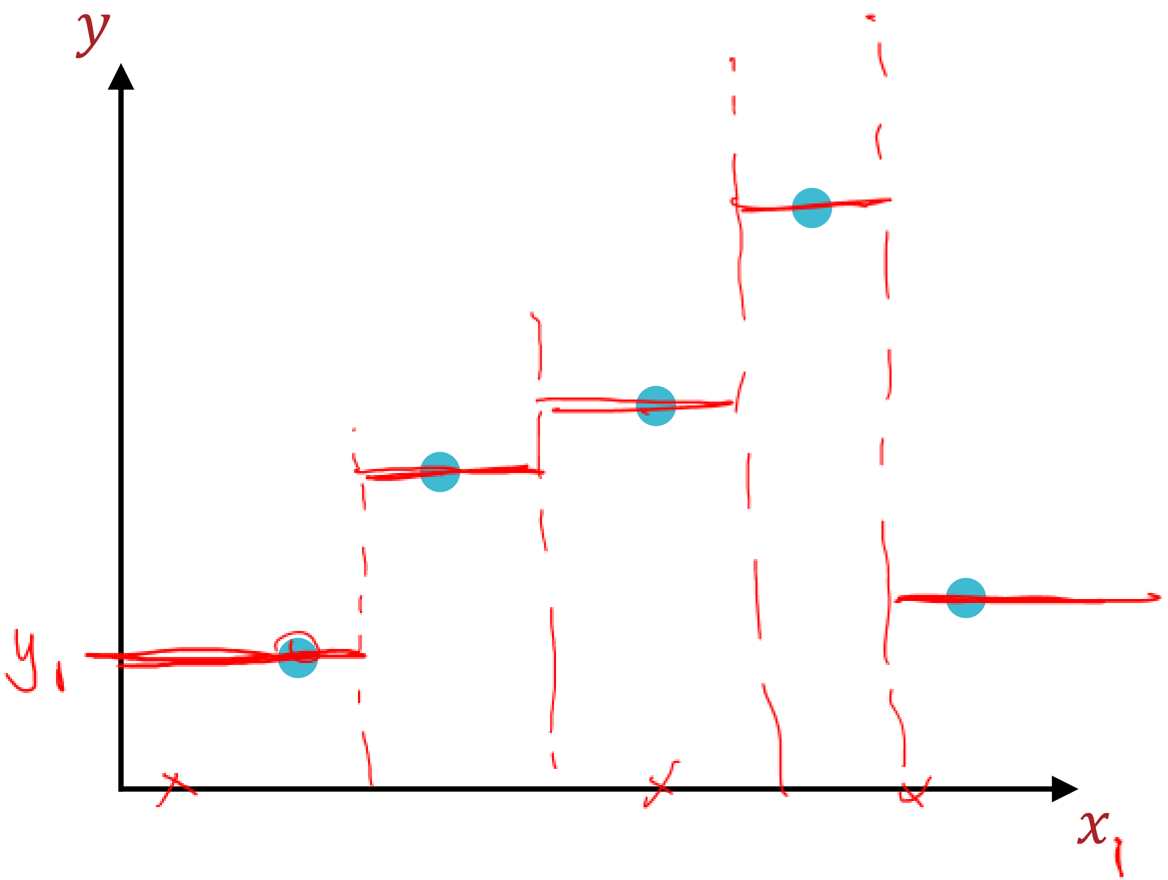
x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	$0
No	Medium	Normal	$20
No	Low	Abnormal	$30
Yes	Medium	Normal	$100
Yes	High	Abnormal	$5000

data points



1-NN Regression

- Suppose we have real-valued targets $y \in \mathbb{R}$ and one-dimensional inputs $x \in \mathbb{R}$



Linear Regression

- Suppose we have real-valued targets $y \in \mathbb{R}$ and D-dimensional inputs $\boldsymbol{x} = [1, x_1, \dots, x_D]^T \in \mathbb{R}^D$
- **Assume**

$$\underbrace{y}_{} = \underbrace{\boldsymbol{w}^T\boldsymbol{x} + w_0}_{}$$

$$y = w^T x \qquad x \in \mathbb{R}^{D+1}$$

General Recipe for Machine Learning

1. Define a hypothesis class (and model parameters)

2. Write down an objective function

3. Optimize the objective w.r.t. the model parameters

Recipe for Linear Regression

1. Define a hypothesis class (and model parameters)
    1. Assume $y = \boldsymbol{w}^T \boldsymbol{x}$
    2. Parameters: $\boldsymbol{w} = [w_0, w_1, \ldots, w_D]$

2. Write down an objective function
    1. Minimize the mean squared error

$$\ell_{\mathcal{D}}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \left(\boldsymbol{w}^T \boldsymbol{x}^{(n)} - y^{(n)} \right)^2$$

3. Optimize the objective w.r.t. the model parameters
    1. Solve in *closed form*: take partial derivatives, set to 0 and solve

Matrix Notation

- Suppose we have real-valued targets $y \in \mathbb{R}$ and D-dimensional inputs $\boldsymbol{x} = [1, x_1, \dots, x_D]^T \in \mathbb{R}^{D+1}$
- **Assume**

$$y = \boldsymbol{w}^T\boldsymbol{x} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

- Notation: given training data $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$

- $$X = \begin{bmatrix} 1 & \boldsymbol{x}^{(1)^T} \\ 1 & \boldsymbol{x}^{(2)^T} \\ \vdots & \vdots \\ 1 & \boldsymbol{x}^{(N)^T} \end{bmatrix} = \begin{bmatrix} \boxed{1 \quad x_1^{(1)} \quad \cdots \quad x_D^{(1)}} \\ 1 \quad x_1^{(2)} \quad \cdots \quad x_D^{(2)} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ 1 \quad x_1^{(N)} \quad \cdots \quad x_D^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times D+1}$$

feature vector for i=1

⋮

i=N

is the *design matrix*

- $\boldsymbol{y} = \left[y^{(1)}, \dots, y^{(N)}\right]^T \in \mathbb{R}^N$ is the *target vector*

Minimizing the Squared Error

$$\ell_D(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\underbrace{\boldsymbol{w}^T\boldsymbol{x}^{(n)}} - y^{(n)}\right)^2 = \underbrace{\frac{1}{N}\sum_{n=1}^{N}\left(x^{(n)T}\omega - y^{(n)}\right)^2}$$

verify

$$= \frac{1}{N}(X\omega - Y)^T(X\omega - Y) \quad \left[= \frac{1}{N}\|X\omega - Y\|_2^2\right]$$

$$= \frac{1}{N}\Big(\omega^T X^T X \omega - \underbrace{\underbrace{Y^T X \omega} - \underbrace{\omega^T X^T Y}}_{2\omega^T X^T Y} + \underbrace{Y^T Y}\Big)$$

$$\nabla_\omega \ell_D(\omega) = 2X^TX\omega - 2X^TY \cancel{+0}$$

$$= 0 \Rightarrow \cancel{2}X^TX\hat{\omega} = \cancel{2}X^TY$$

$$\Rightarrow \boxed{\hat{\omega} = (X^TX)^{-1}X^TY}$$

$$H_\omega \ell_D(\omega) = 2X^TX \succeq 0$$

$\hookrightarrow$ p.s.d

Minimizing the Squared Error

$$\ell_{\mathcal{D}}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} - y^{(n)}\right)^2 = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}^{(n)^T}\boldsymbol{w} - y^{(n)}\right)^2$$

$$= \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 \text{ where } \|\boldsymbol{z}\|_2 = \sqrt{\sum_{d=1}^{D} z_d^2} = \sqrt{\boldsymbol{z}^T\boldsymbol{z}}$$

$$= \frac{1}{N}(X\boldsymbol{w} - \boldsymbol{y})^T(X\boldsymbol{w} - \boldsymbol{y})$$

$$= \frac{1}{N}(\boldsymbol{w}^T X^T X\boldsymbol{w} - 2\boldsymbol{w}^T X^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y})$$

$$\nabla_{\boldsymbol{w}}\ell_{\mathcal{D}}(\widehat{\boldsymbol{w}}) = \frac{1}{N}(2X^T X\widehat{\boldsymbol{w}} - 2X^T\boldsymbol{y}) = 0$$

$$\rightarrow X^T X\widehat{\boldsymbol{w}} = X^T\boldsymbol{y}$$

$$\rightarrow \widehat{\boldsymbol{w}} = (X^T X)^{-1}X^T\boldsymbol{y}$$

Minimizing the Squared Error

$$\ell_{\mathcal{D}}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} - y^{(n)}\right)^2 = \frac{1}{N}\sum_{n=1}^{N}\left({\boldsymbol{x}^{(n)}}^T\boldsymbol{w} - y^{(n)}\right)^2$$

$$= \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 \text{ where } \|\boldsymbol{z}\|_2 = \sqrt{\sum_{d=1}^{D} z_d^2} = \sqrt{\boldsymbol{z}^T\boldsymbol{z}}$$

$$= \frac{1}{N}(X\boldsymbol{w} - \boldsymbol{y})^T(X\boldsymbol{w} - \boldsymbol{y})$$

$$= \frac{1}{N}\left(\boldsymbol{w}^T X^T X\boldsymbol{w} - 2\boldsymbol{w}^T X^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y}\right)$$

$$\nabla_{\boldsymbol{w}}\ell_{\mathcal{D}}(\hat{\boldsymbol{w}}) = \frac{1}{N}\left(2X^T X\hat{\boldsymbol{w}} - 2X^T\boldsymbol{y}\right) = 0$$

$$H_{\boldsymbol{w}}\ell_{\mathcal{D}}(\boldsymbol{w}) = \frac{2}{N}X^T X \rightarrow H_{\boldsymbol{w}}\ell_{\mathcal{D}}(\boldsymbol{w}) \text{ is positive semi-definite}$$

$$\hat{\boldsymbol{w}} = (X^T X)^{-1} X^T \boldsymbol{y}$$

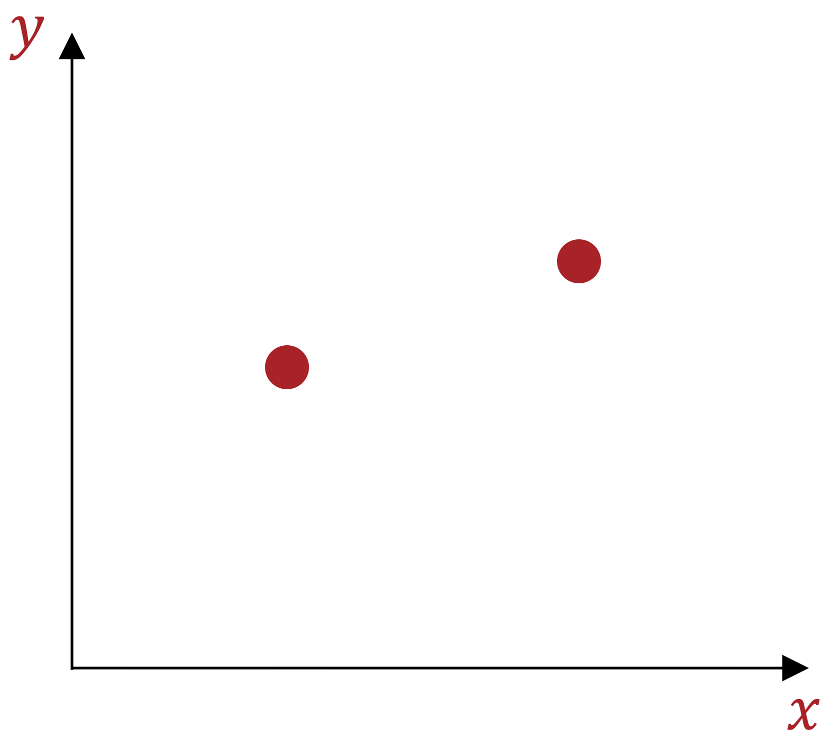
Closed Form Solution

1. Is $X^T X$ invertible?

2. If so, how computationally expensive is inverting $X^T X$?

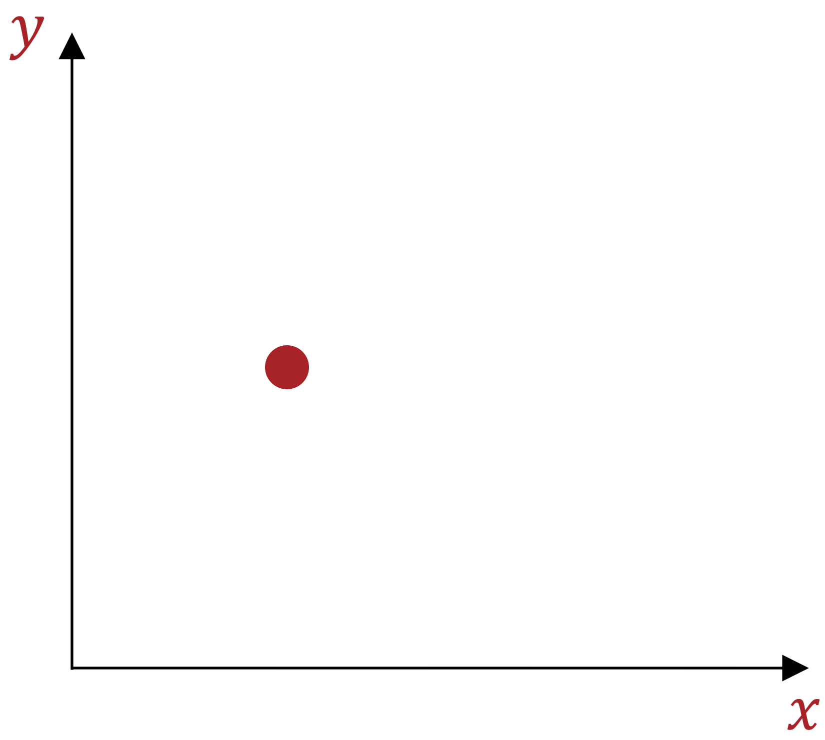
Linear Regression: Uniqueness

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



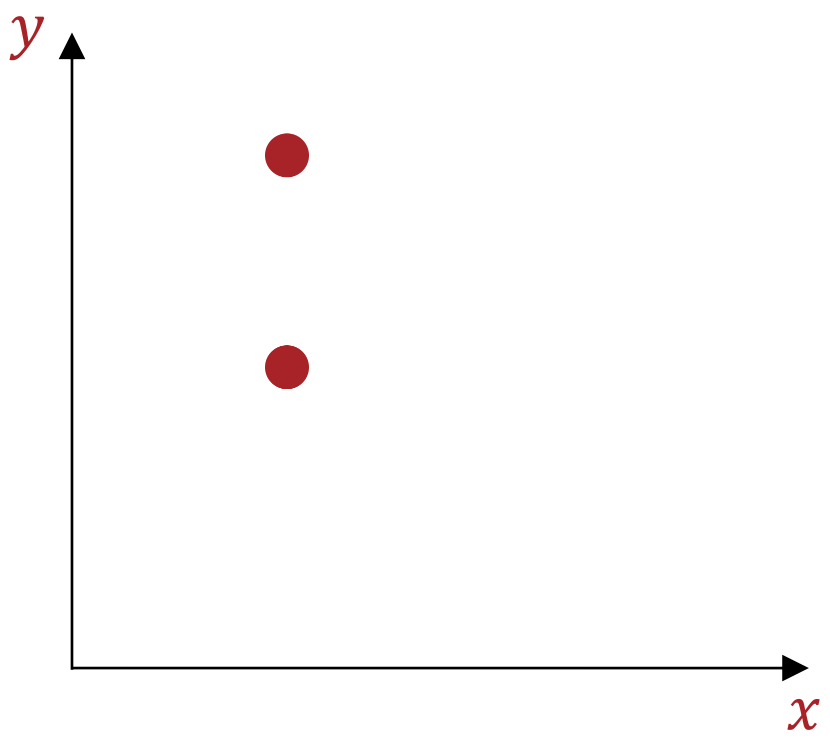
Linear Regression: Uniqueness

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



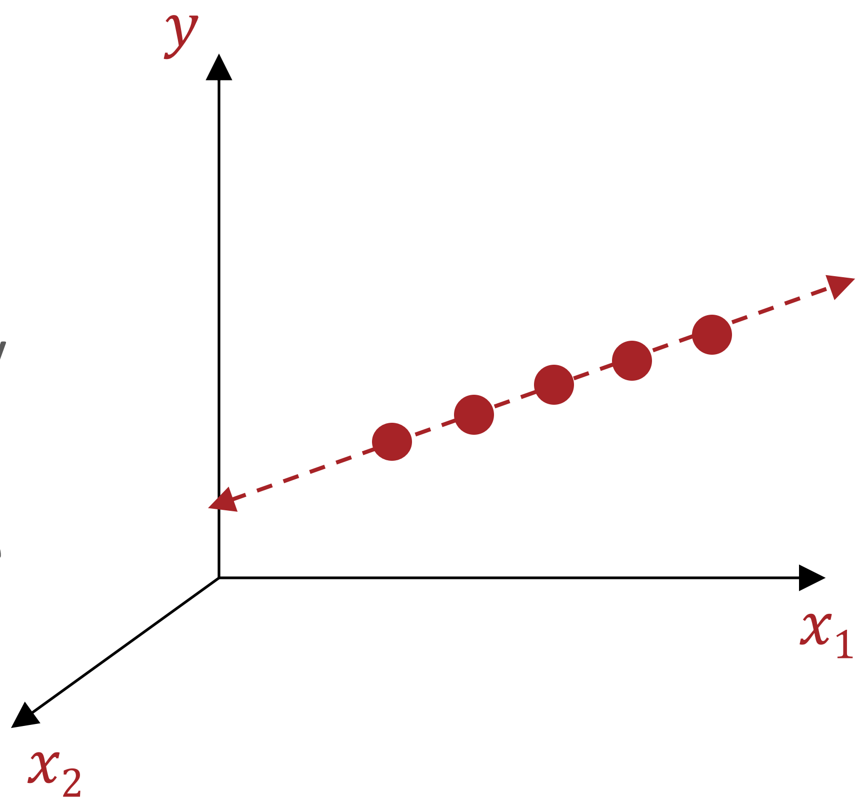
Linear Regression: Uniqueness

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



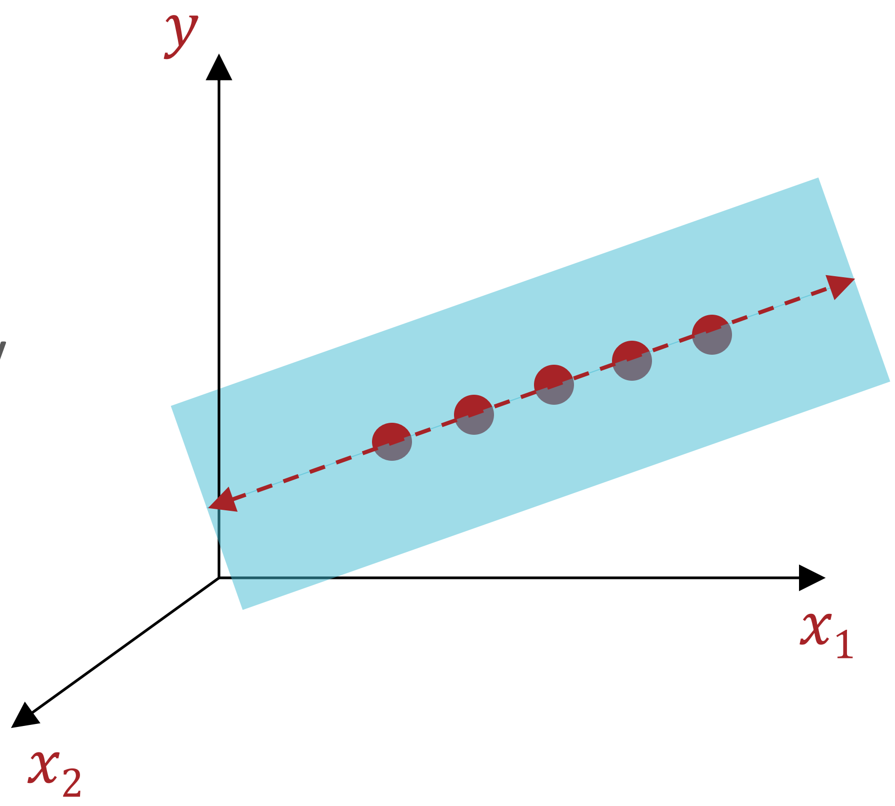
Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



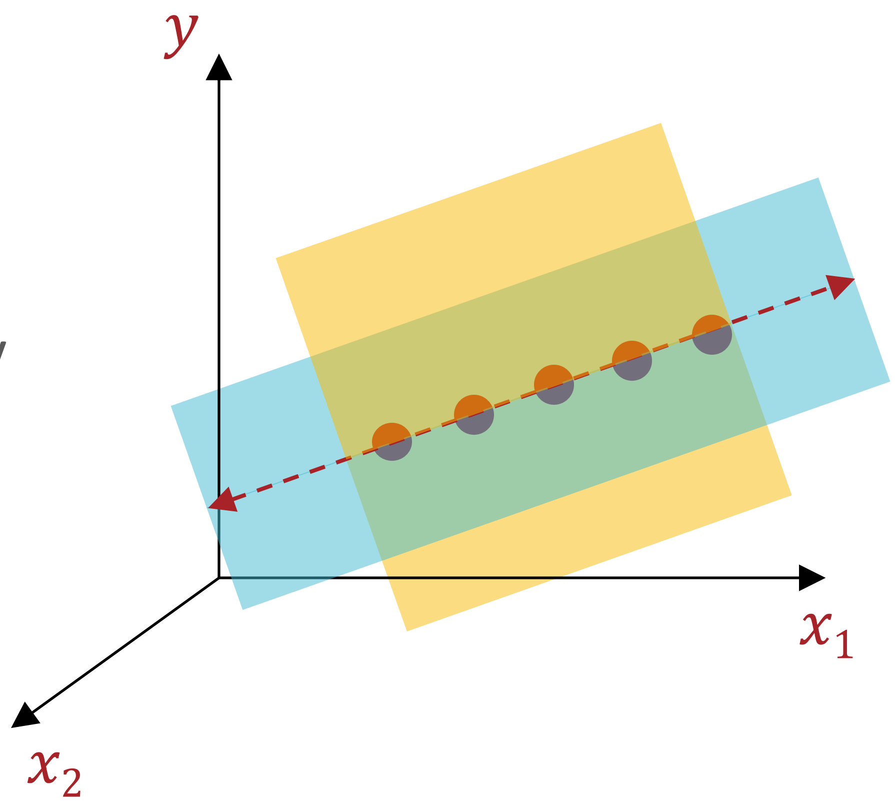
Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



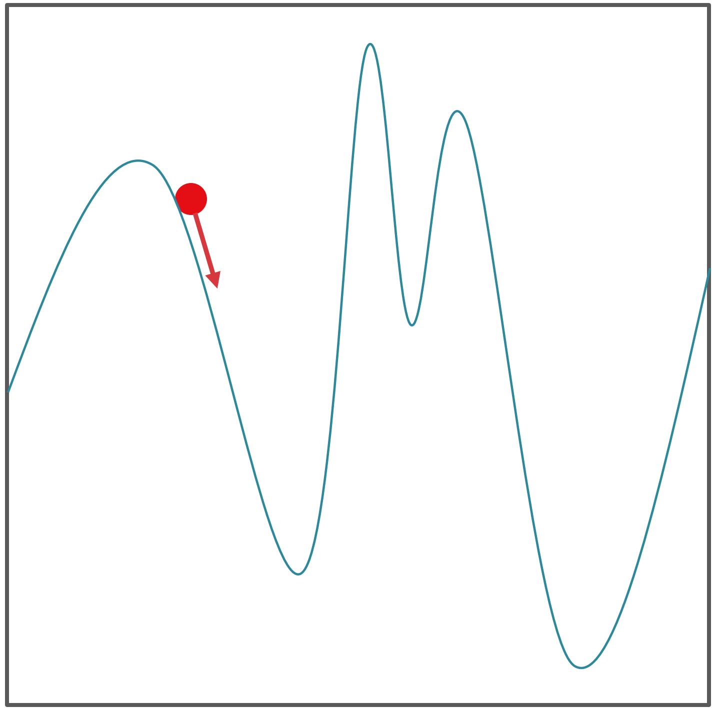
$$\hat{\boldsymbol{w}} = (X^T X)^{-1} X^T \boldsymbol{y}$$

Closed Form Solution

1. Is $X^T X$ invertible?
   - When $N \gg D + 1$, $X^T X$ is (almost always) full rank and therefore, invertible
   - If $X^T X$ is not invertible (occurs when one of the features is a linear combination of the others) then there are infinitely many solutions.
2. If so, how computationally expensive is inverting $X^T X$?
   - $X^T X \in \mathbb{R}^{D+1 \times D+1}$ so inverting $X^T X$ takes $O(D^3)$ time…
     - Computing $X^T X$ takes $O(ND^2)$ time
   - What alternative optimization method can we use to minimize the mean squared error?

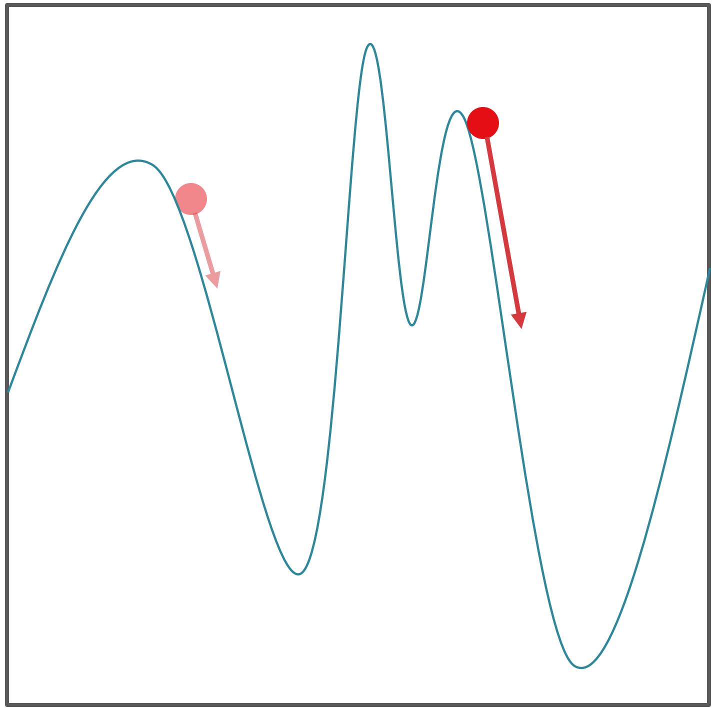
Gradient Descent: Intuition

- An iterative method for minimizing functions
- Requires the gradient to exist everywhere



Gradient Descent: Intuition

- An iterative method for minimizing functions
- Requires the gradient to exist everywhere



Gradient Descent: Intuition

- An iterative method for minimizing functions
- Requires the gradient to exist everywhere