上海科技大学
ShanghaiTech University

CS182: Introduction to Machine Learning – Linear Regression

Yujiao Shi
SIST, ShanghaiTech
Spring, 2025

Slides Courtesy of Matt Gormley, Henry Chai, and Maria-Florina Balcan

PRACTICE

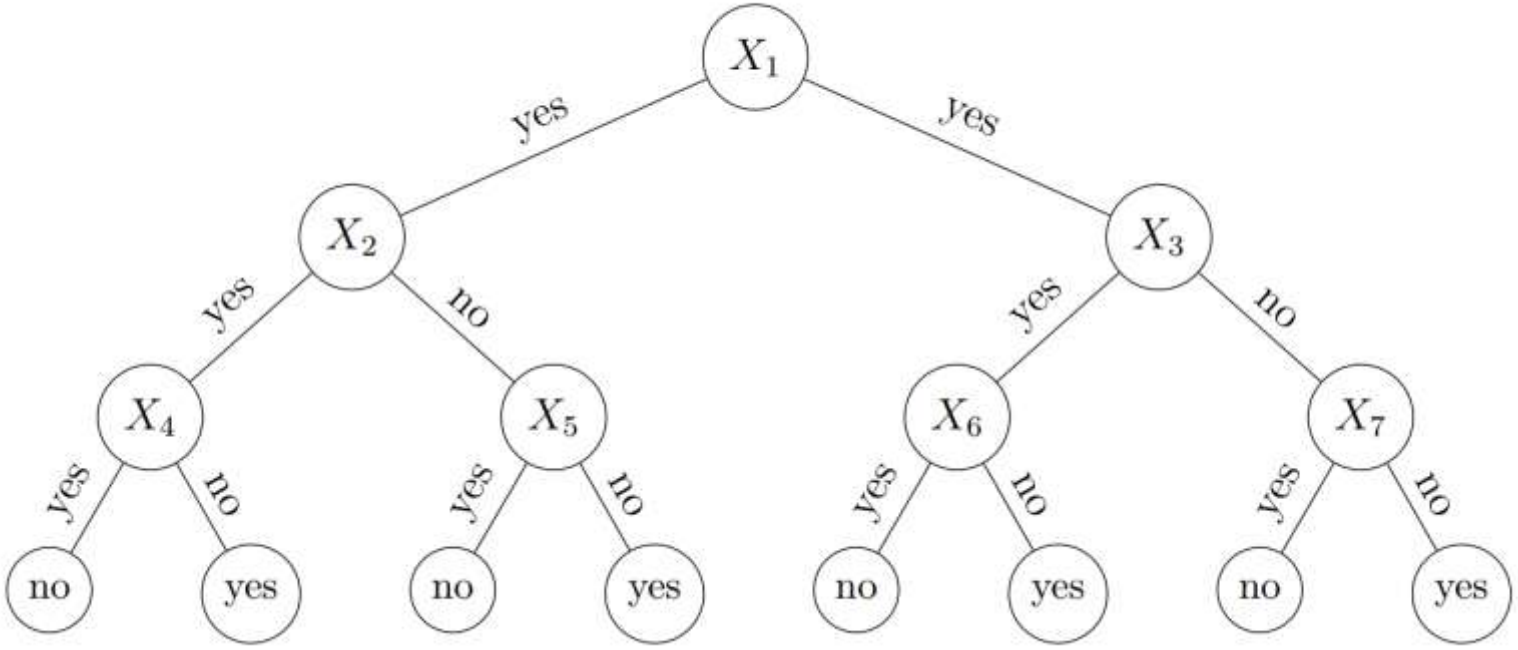
Decision Trees

上海科技大学
ShanghaiTech University

- **Depth of a Decision Tree:**
  - The depth of a tree is the length (number of edges) of the longest path from a root to a leaf.

- **Depth of a node definition:**
  - The depth of a node is the number of edges between the root and the given node

What is the depth of tree A? What is the depth of node X_4 in tree A?



Tree A

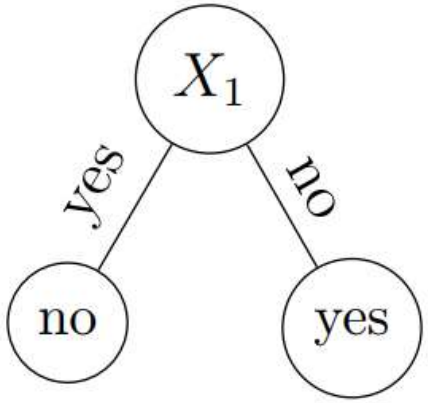
Decision Trees

上海科技大学
ShanghaiTech University

- **Depth of a Decision Tree:**
  - The depth of a tree is the length (number of edges) of the longest path from a root to a leaf.

- **Depth of a node definition:**
  - The depth of a node is the number of edges between the root and the given node

What is the depth of tree B?



Tree B

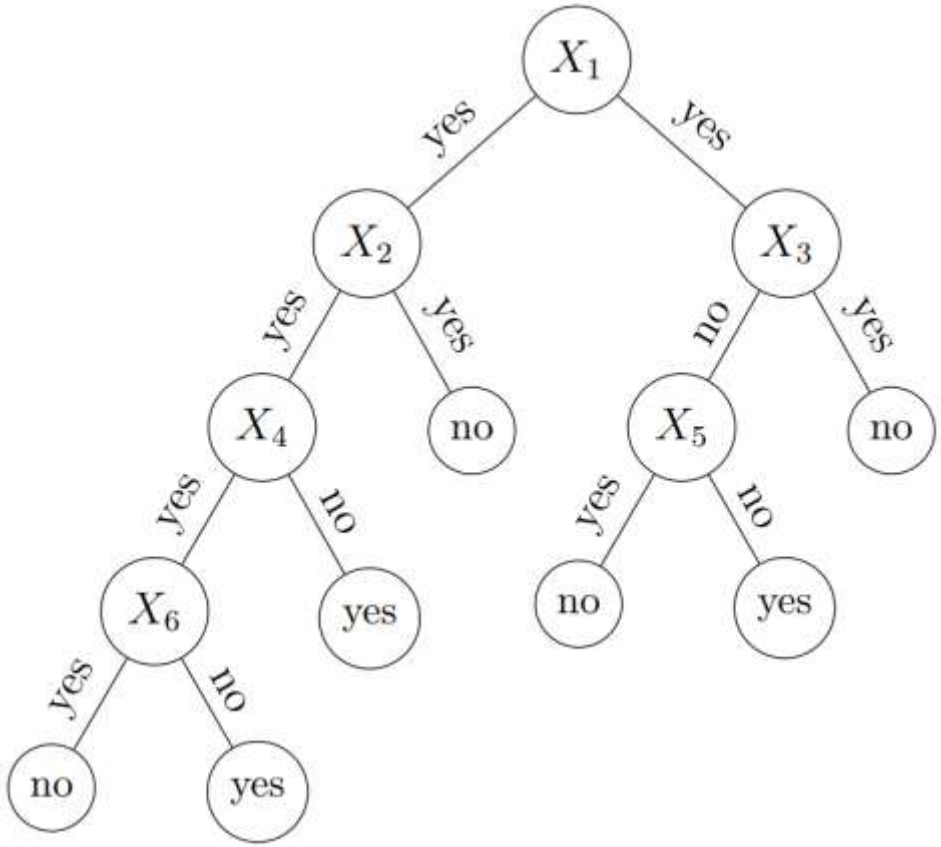
Decision Trees

上海科技大学
ShanghaiTech University

- **Depth of a Decision Tree:**
  - The depth of a tree is the length (number of edges) of the longest path from a root to a leaf.

- **Depth of a node definition:**
  - The depth of a node is the number of edges between the root and the given node

What is the depth of tree C? What are the depths of nodes X_1 and X_5 in tree C?



Tree C

How to construct Decision Trees

上海科技大学
ShanghaiTech University

1. What exactly are the tasks we are tackling?

How to construct Decision Trees

上海科技大学
ShanghaiTech University

2. What are the inputs and outputs at training time? At testing time?

How to construct Decision Trees

上海科技大学
ShanghaiTech University

3. At each node of the tree, what do we need to store?

How to construct Decision Trees

上海科技大学
ShanghaiTech University

4. What do we need to do at training time?

How to construct Decision Trees

上海科技大学
ShanghaiTech University

5. What do we need to do at testing time?

How to construct Decision Trees

6. What happens if max depth is 0?

How to construct Decision Trees

上海科技大学
ShanghaiTech University

7. What happens if max depth is greater than the number of attributes?

Mutual Information

上海科技大学
ShanghaiTech University

Information Theory Definitions:

- $H(Y) = -\sum_{y \in values(Y)} P(Y = y) \log_2 P(Y = y)$
- $H(Y \mid X = x) = -\sum_{y \in values(Y)} P(Y = y|X = x) \log_2 P(Y = y|X = x)$
- $H(Y \mid X) = \sum_{x \in values(X)} P(X = x) H(Y \mid X = x)$
- $I(Y; X) = H(Y) - H(Y \mid X) = H(X) - H(X \mid Y)$

Exercises:

1. Calculate the entropy of tossing a fair coin.

Mutual Information

上海科技大学
ShanghaiTech University

Information Theory Definitions:

- $H(Y) = -\sum_{y \in values(Y)} P(Y = y) \log_2 P(Y = y)$
- $H(Y \mid X = x) = -\sum_{y \in values(Y)} P(Y = y|X = x) \log_2 P(Y = y|X = x)$
- $H(Y \mid X) = \sum_{x \in values(X)} P(X = x) H(Y \mid X = x)$
- $I(Y; X) = H(Y) - H(Y \mid X) = H(X) - H(X \mid Y)$

Exercises:

2. Calculate the entropy of tossing a coin that lands only on tails.

Mutual Information

上海科技大学
ShanghaiTech University

Information Theory Definitions:

- $H(Y) = -\sum_{y \in values(Y)} P(Y = y) \log_2 P(Y = y)$
- $H(Y \mid X = x) = -\sum_{y \in values(Y)} P(Y = y|X = x) \log_2 P(Y = y|X = x)$
- $H(Y \mid X) = \sum_{x \in values(X)} P(X = x) H(Y \mid X = x)$
- $I(Y; X) = H(Y) - H(Y \mid X) = H(X) - H(X \mid Y)$

Exercises:

3. Calculate the entropy of a fair dice roll.

Mutual Information

上海科技大学
ShanghaiTech University

Information Theory Definitions:

- $H(Y) = -\sum_{y \in values(Y)} P(Y = y) \log_2 P(Y = y)$
- $H(Y \mid X = x) = -\sum_{y \in values(Y)} P(Y = y|X = x) \log_2 P(Y = y|X = x)$
- $H(Y \mid X) = \sum_{x \in values(X)} P(X = x) H(Y \mid X = x)$
- $I(Y; X) = H(Y) - H(Y \mid X) = H(X) - H(X \mid Y)$

Exercises:

4.When is the mutual information $I(Y; X) = 0$?

Decision Tree: Classification with Continuous Attributes

Decision Tree Classification with Continuous Attributes

Given the dataset $\mathcal{D}_1 = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^2, y^{(i)} \in \{\text{Yellow}, \text{Purple}, \text{Green}\}$ as shown in Fig. 1, we wish to learn a decision tree for classifying such points. Provided with a possible tree structure in Fig. 1, what values of α, β and leaf node predictions could we use to perfectly classify the points? Now, draw the associated decision boundaries on the scatter plot.

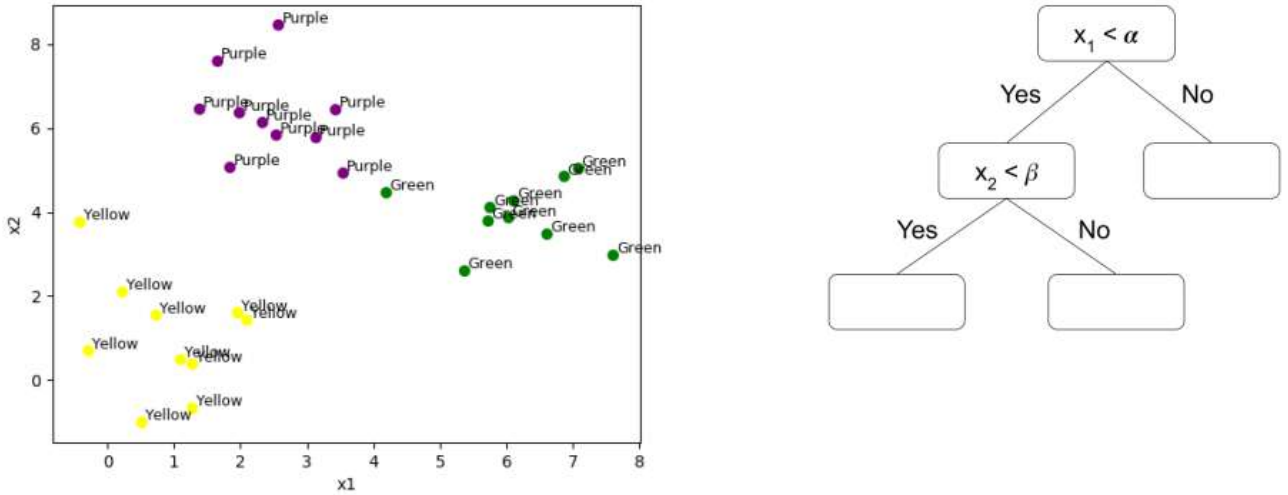
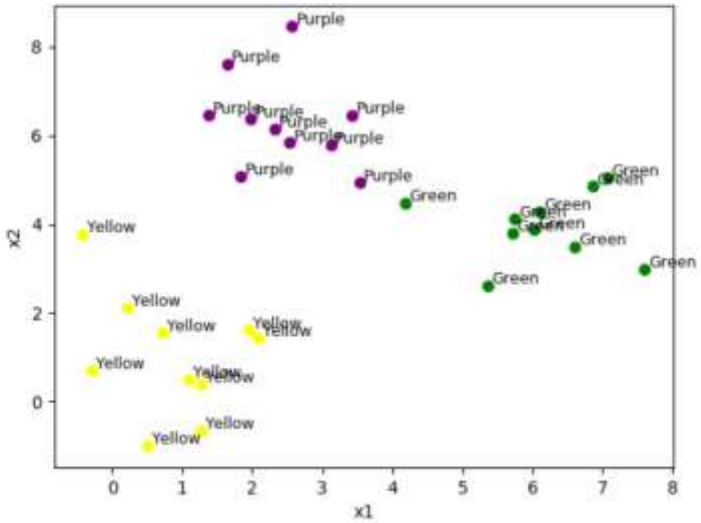


Figure 1: Classification of 2D points, with Decision Tree to fill in

Decision Tree

上海科技大学
ShanghaiTech University

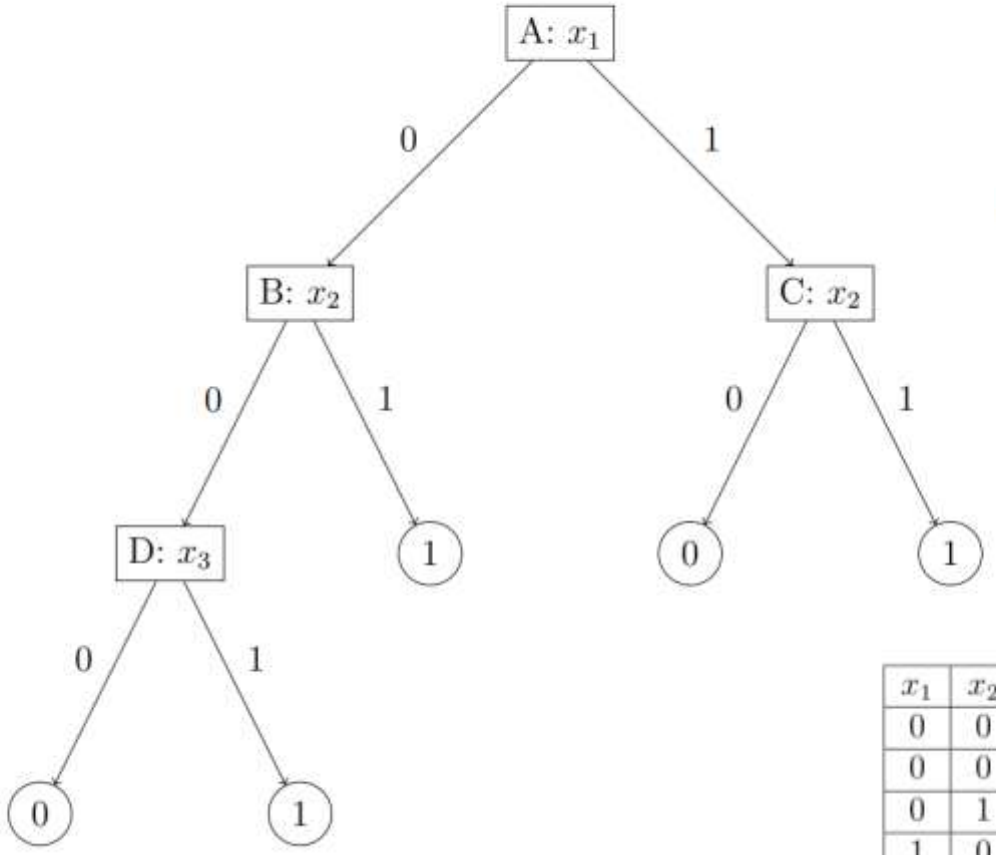
Choosing a Tree: What might happen if we increased the max-depth of the tree? When predicting on unseen data, would we prefer the depth-2 tree above or a very deep tree?



Decision Tree

上海科技大学
ShanghaiTech University

Pruning a Tree: Which node would be the first to be pruned in the following decision tree? In the case of a tie, break the tie in favor of the alphabetically earlier node (eg. prune node B before D)



How do we know when we are done pruning a decision tree?

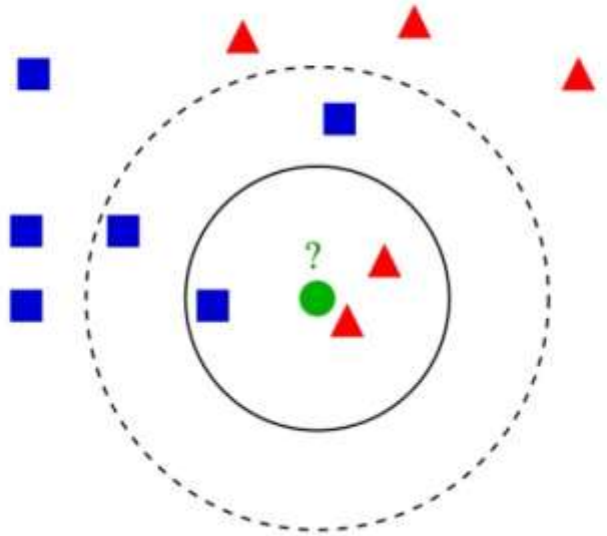
Validation Set

x_1	x_2	x_3	Label	No prune	Prune A	Prune B	Prune C	Prune D
0	0	0	1					
0	0	1	0					
0	1	0	1					
1	0	1	0					
1	1	0	1					

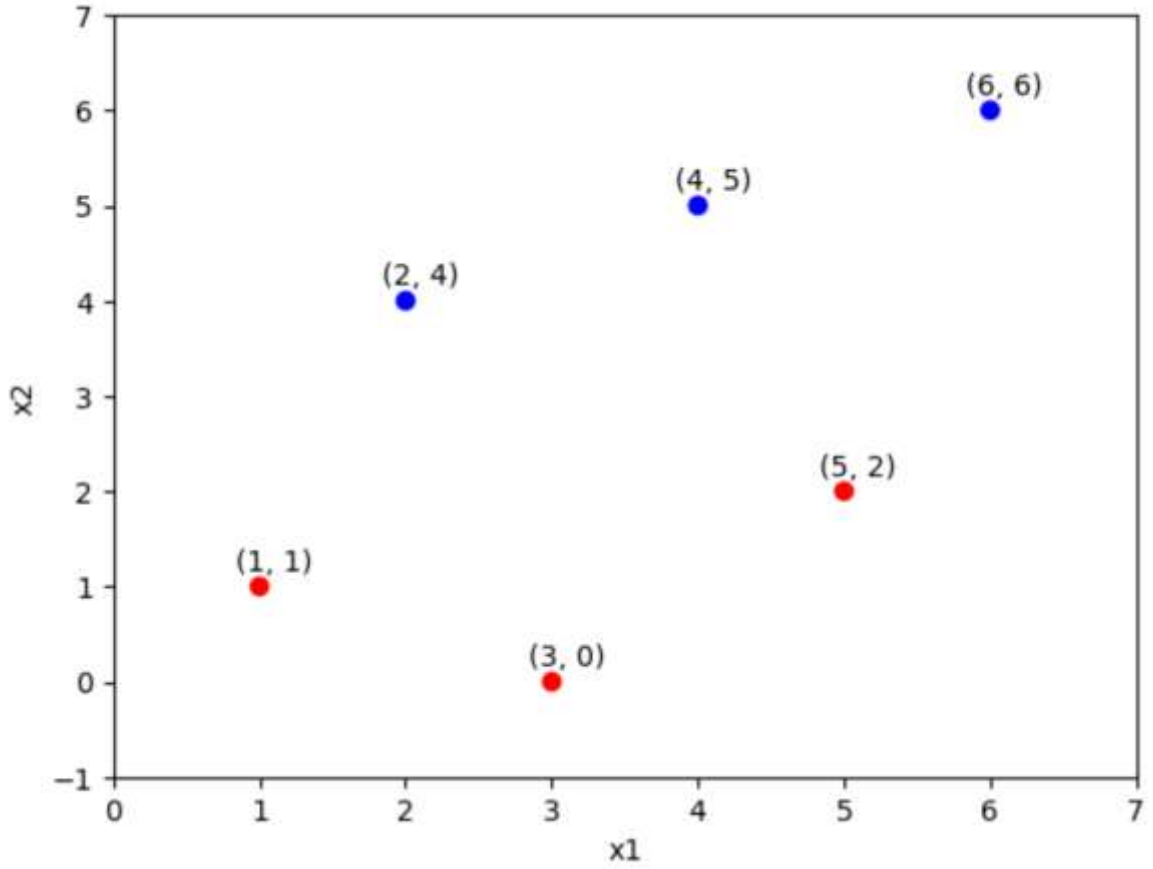
K-NN for Classification

上海科技大学
ShanghaiTech University

Using the figure below, what would you categorize the green circle as with $k = 3$? $k = 5$? $k = 4$?



K-NN Decision Boundary



Draw the decision boundary for the above training dataset given using k-NN algorithm considering k=1

Perceptron

The following table shows a dataset of linearly separable datapoints.

x1	x2	y
1	-1	1
0	2	-1
4	0	1

What linear classifier will you learn from this data?

Perceptron

Given dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, suppose:

1. Finite size inputs: $||x^{(i)}|| \leq R$
2. Linearly separable data: $\exists \boldsymbol{\theta}^*$ and $\boldsymbol{\gamma} > 0$ s.t. $||\boldsymbol{\theta}^*|| = 1$ and $y^{(i)}(\boldsymbol{\theta}^* \cdot x^{(i)}) \geq \boldsymbol{\gamma}, \forall i$

Then, the number of mistakes k made by the perceptron algorithm on $\mathcal{D}$ is bounded by $(R/\boldsymbol{\gamma})^2$.

The following table shows a dataset of linearly separable datapoints.

x1	x2	y
1	-1	1
0	2	-1
4	0	1

Assuming that the linear separator with the largest margin is given by:

$$\theta^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0, \text{ where } \theta = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Calculate the theoretical mistake bound for the perceptron.

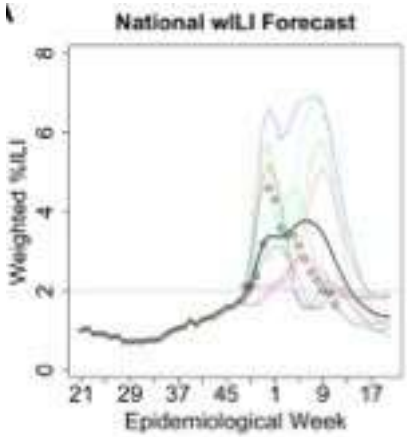
REGRESSION

Regression

Goal:

- Given a training dataset of pairs (**x**,y) where
  - **x** is a vector
  - y is a scalar
- Learn a function (aka. curve or line) y' = h(x) that best fits the training data

This is what differentiates *regression* from *classification*



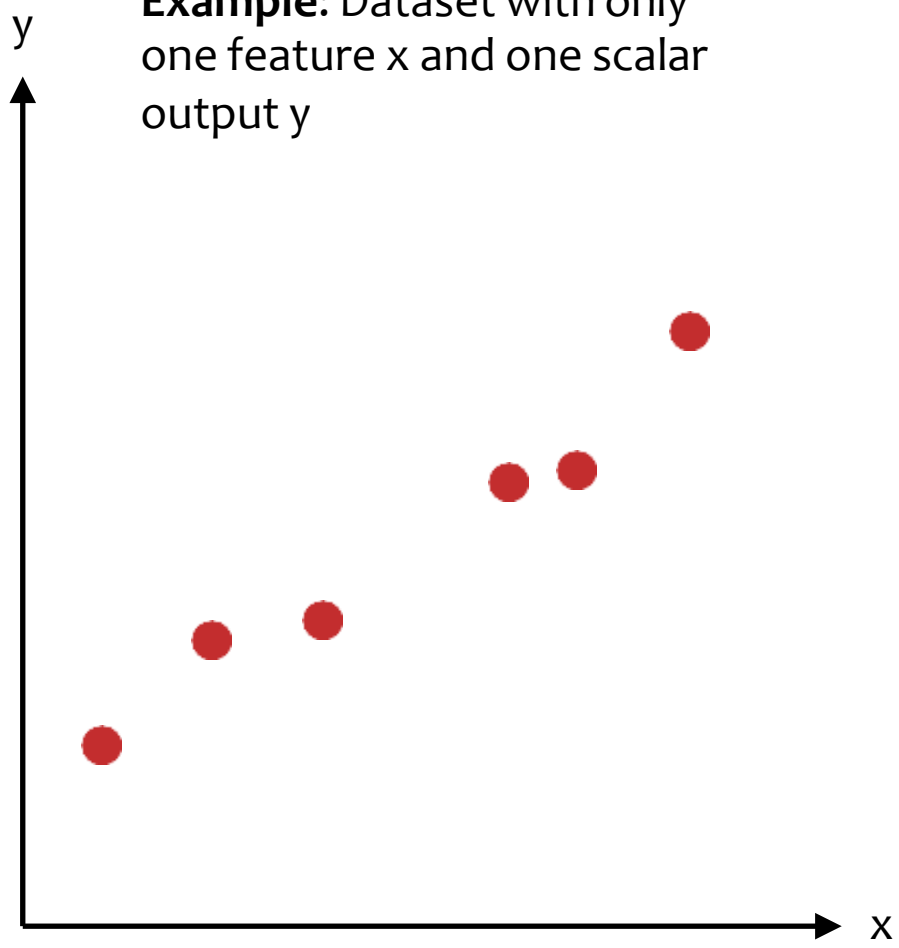
Example Applications:

- Stock price prediction
- Forecasting epidemics
- Speech synthesis
- Generation of images (e.g. *Deep Dream*)

Regression

上海科技大学
ShanghaiTech University

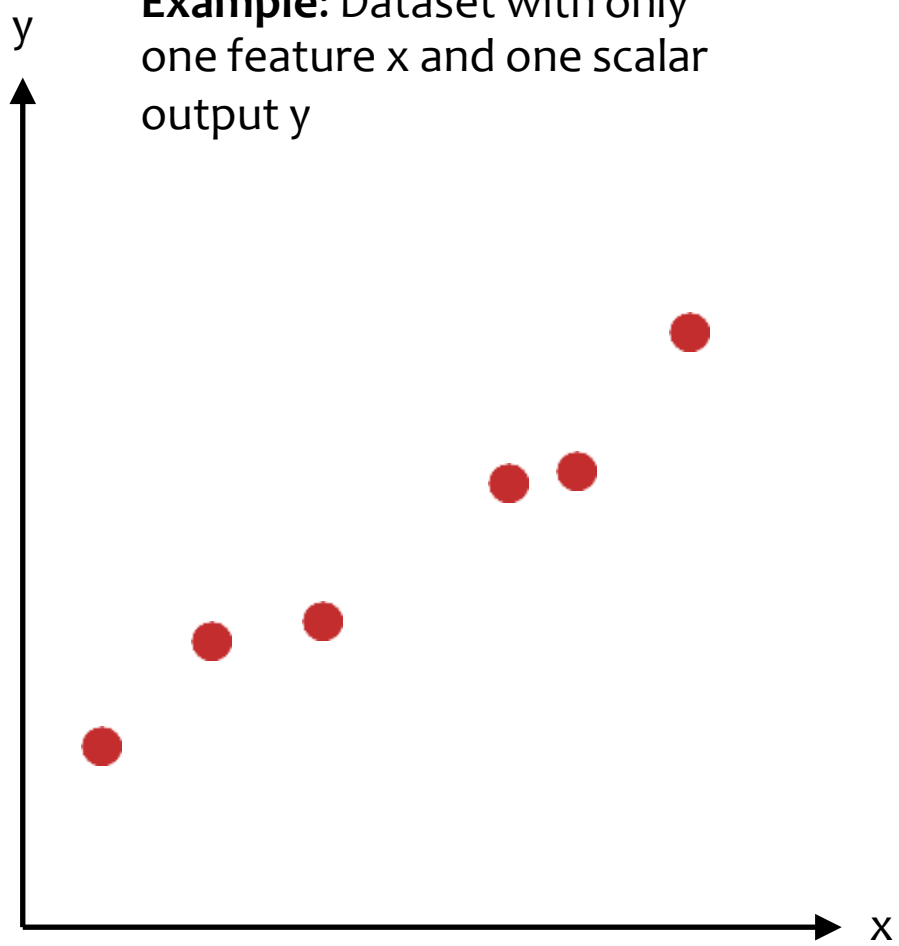
Example: Dataset with only one feature x and one scalar output y



Q: What is the function that best fits these points?

K-NEAREST NEIGHBOR REGRESSION

k-NN Regression



Example: Dataset with only one feature x and one scalar output y

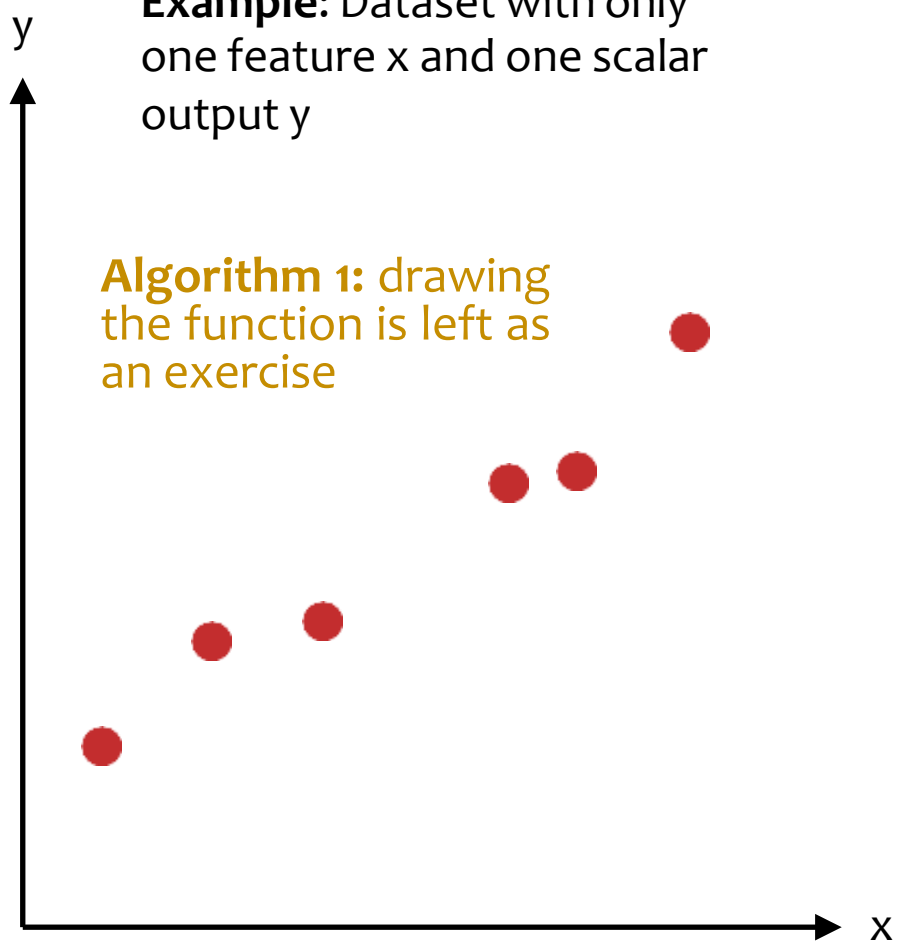
Algorithm 1: k=1 Nearest Neighbor Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

Algorithm 2: k=2 Nearest Neighbors Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

k-NN Regression



Example: Dataset with only one feature x and one scalar output y

Algorithm 1: drawing the function is left as an exercise

Algorithm 1: k=1 Nearest Neighbor Regression

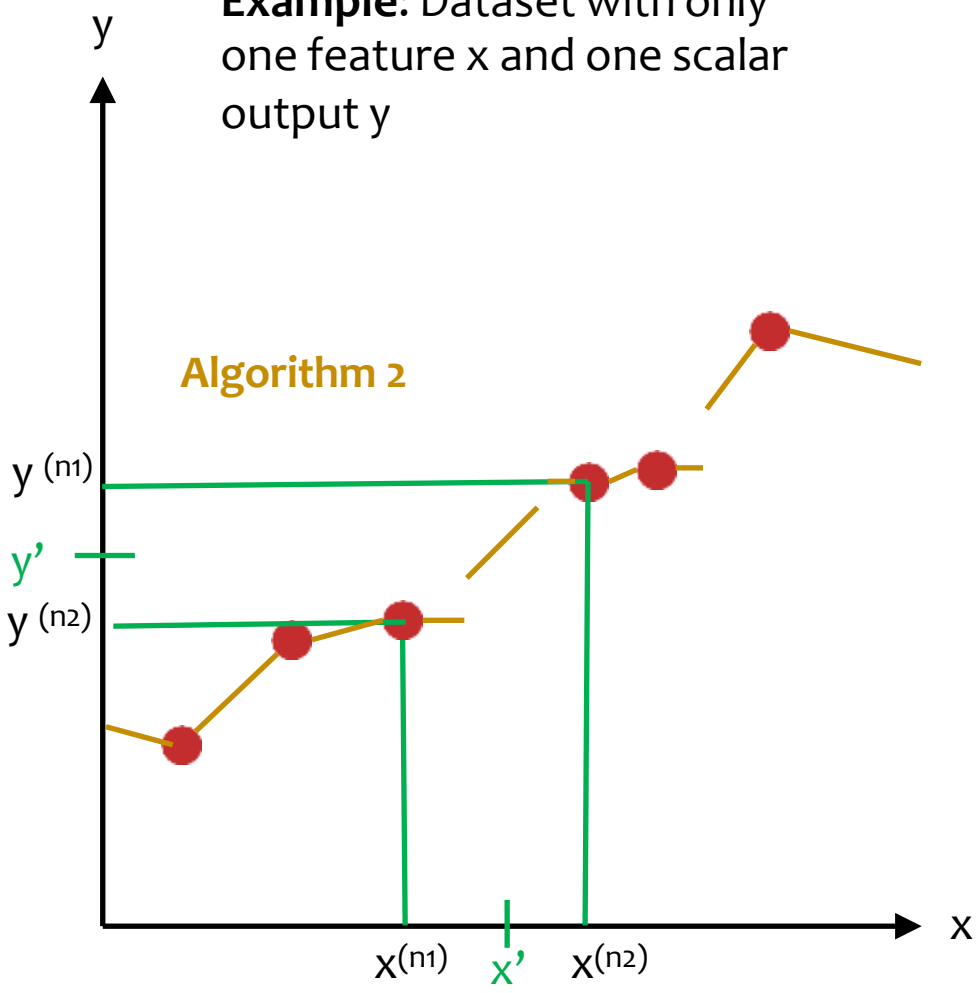
- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

Algorithm 2: k=2 Nearest Neighbors Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

k-NN Regression

Example: Dataset with only one feature x and one scalar output y



Algorithm 1: k=1 Nearest Neighbor Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

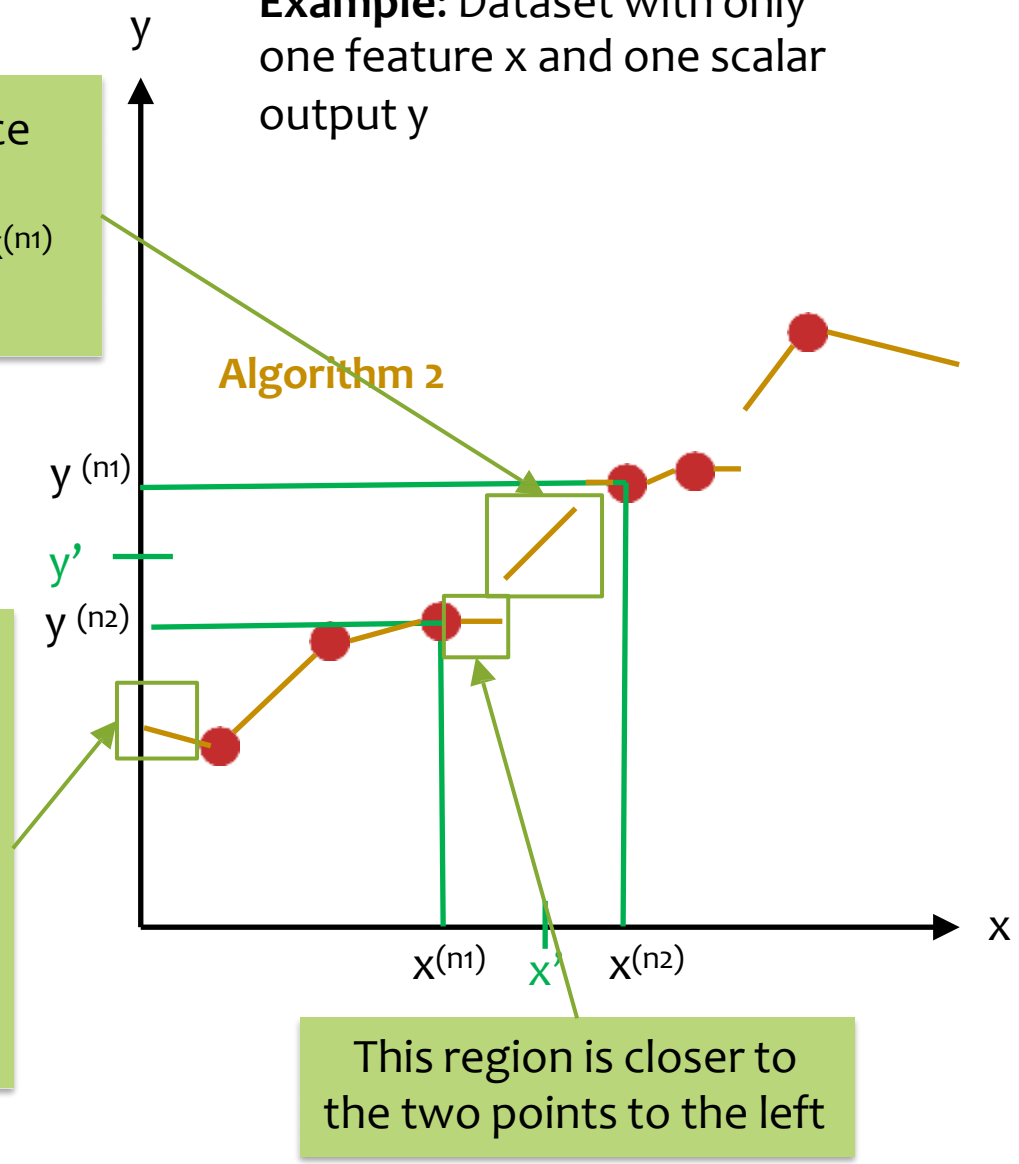
Algorithm 2: k=2 Nearest Neighbors Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

k-NN Regression

上海科技大学 ShanghaiTech University

Example: Dataset with only one feature x and one scalar output y



The distance weighted average of $x^{(n1)}$ and $x^{(n2)}$

This tends toward the average height of the leftmost two points

This region is closer to the two points to the left

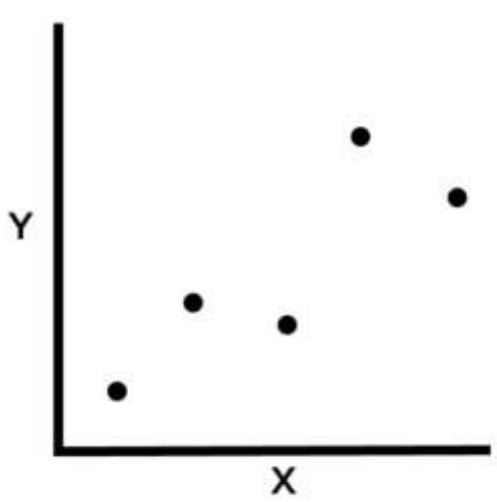
Algorithm 1: k=1 Nearest Neighbor Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

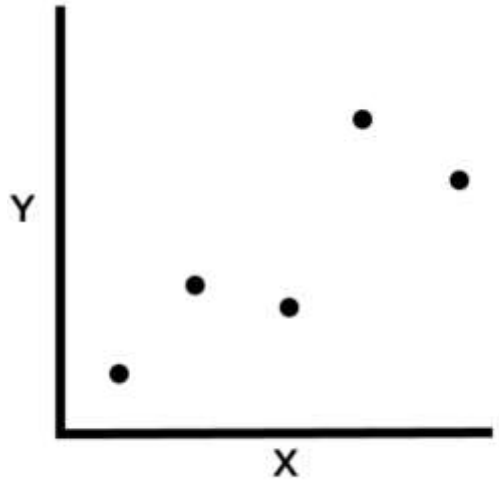
Algorithm 2: k=2 Nearest Neighbors Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

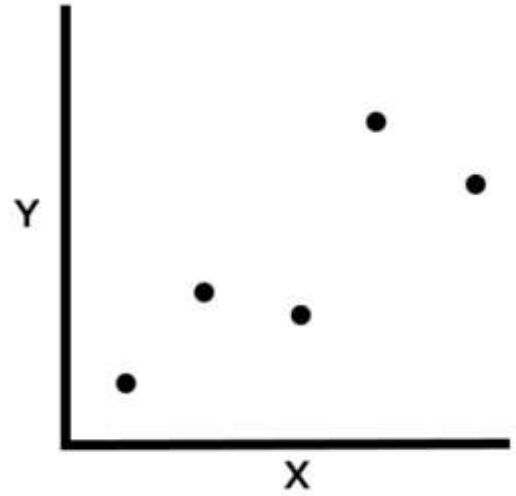
Practice: k-NN Regression



(a) $k = 1$



(b) weighted $k = 2$



(c) unweighted $k = 2$

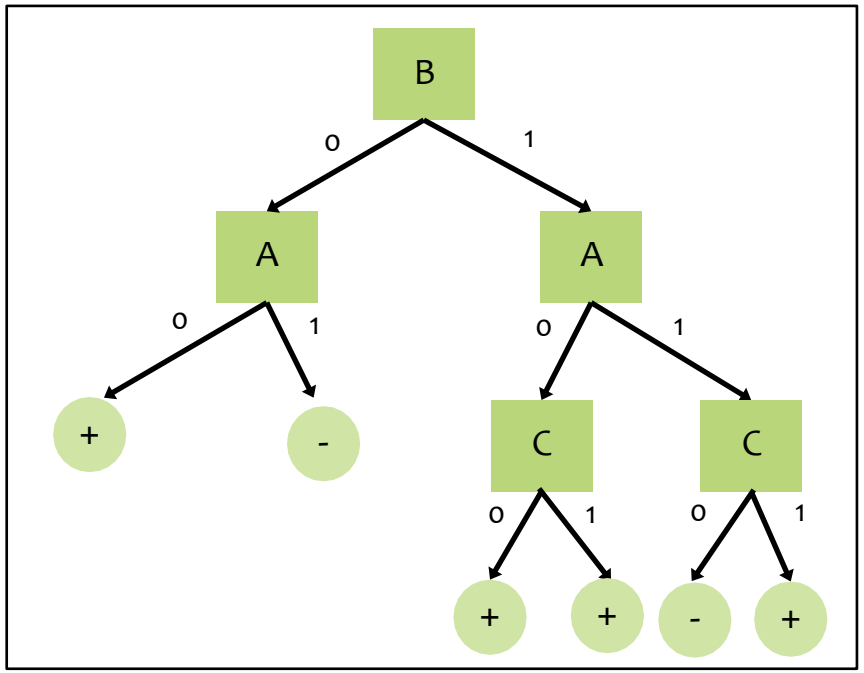
Draw the Regression Lines.

Note: The points are equidistant along the x-axis.

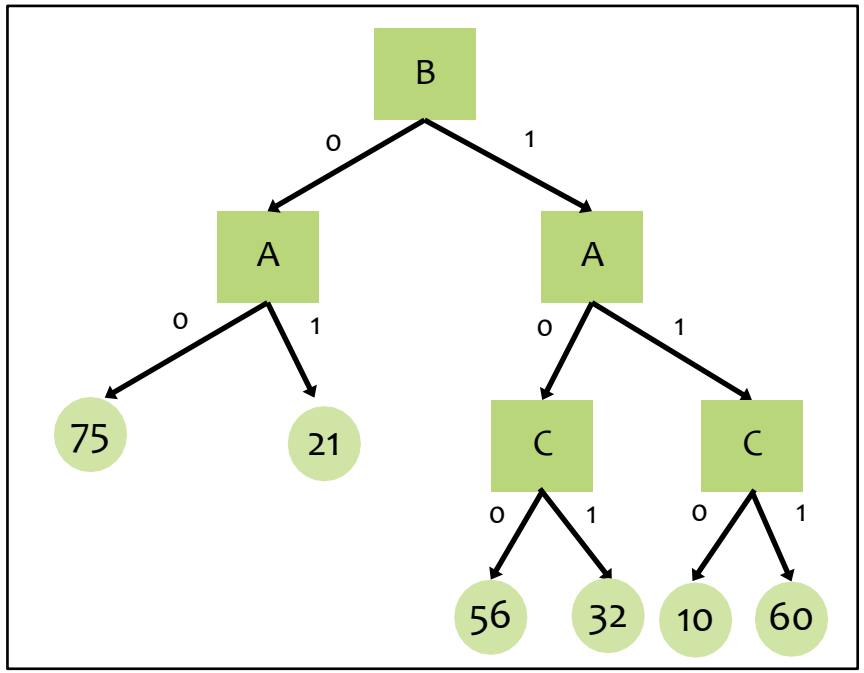
DECISION TREE REGRESSION

Decision Tree Regression

Decision Tree for Classification



Decision Tree for Regression

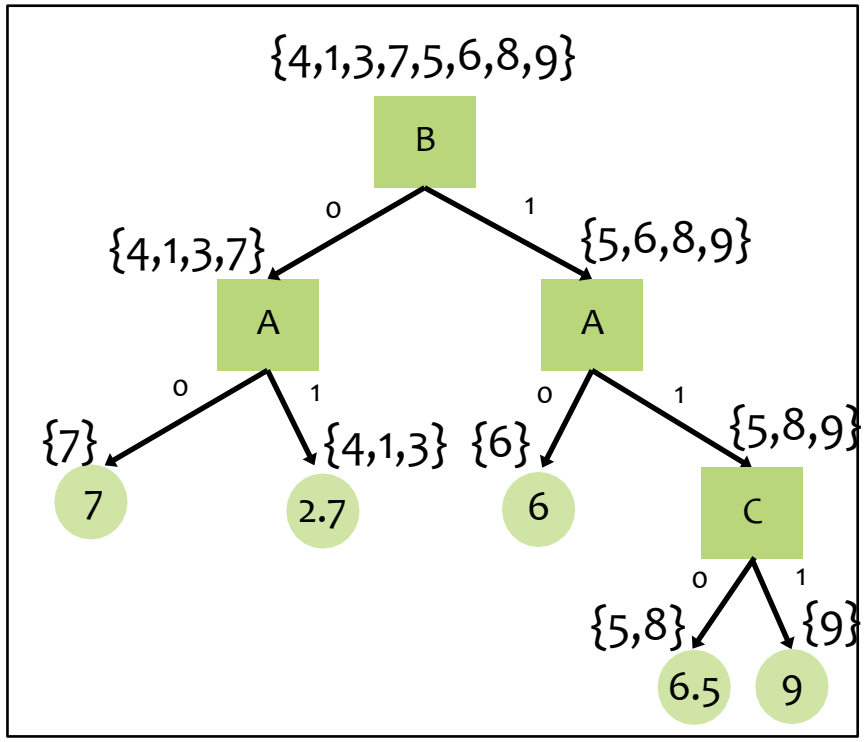


Decision Tree Regression

Dataset for Regression

Y	A	B	C
4	1	0	0
1	1	0	1
3	1	0	0
7	0	0	1
5	1	1	0
6	0	1	1
8	1	1	0
9	1	1	1

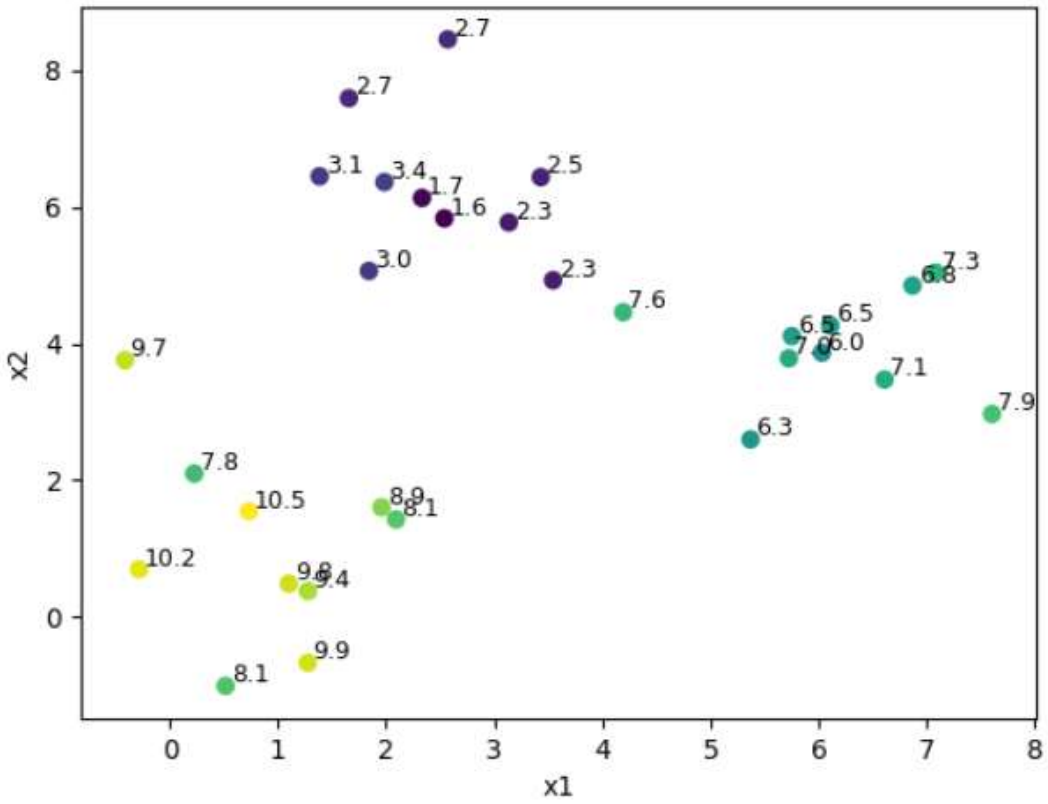
Decision Tree for Regression

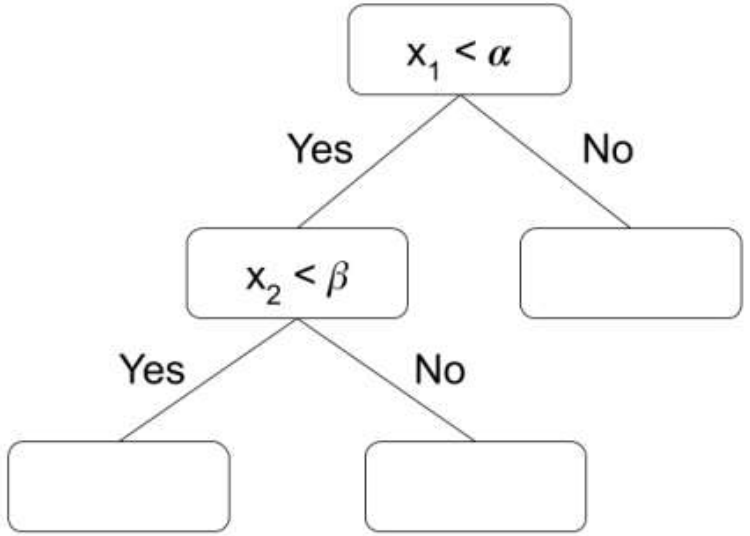


During learning, choose the attribute that minimizes an appropriate splitting criterion (e.g. mean squared error, mean absolute error)

Practice: Decision Tree Regression

上海科技大学
ShanghaiTech University





LINEAR FUNCTIONS, RESIDUALS, AND MEAN SQUARED ERROR

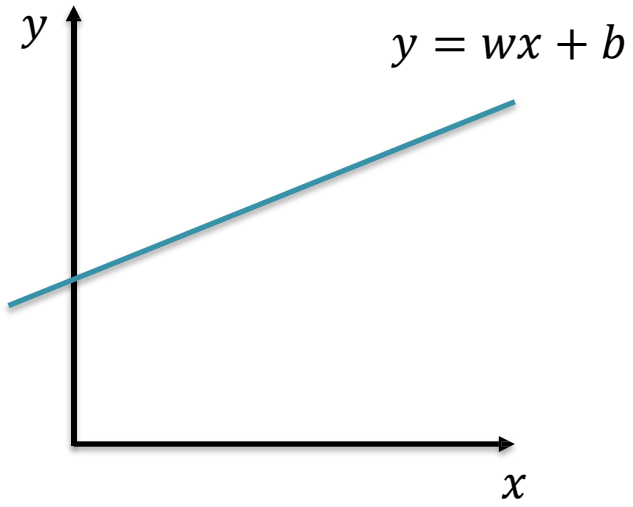
Linear Functions

Def: Regression is predicting real-valued outputs

$$\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n} \text{ with } \mathbf{x}^{(i)} \in \mathbb{R}^M, y^{(i)} \in \mathbb{R}$$

***Common Misunderstanding*:**

Linear functions ≠ Linear decision boundaries



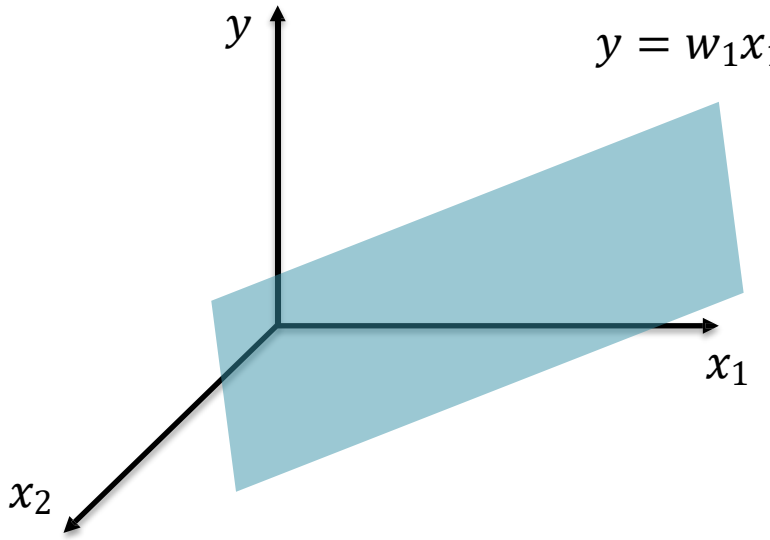
Linear Functions

Def: Regression is predicting real-valued outputs

$$\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n} \text{ with } \mathbf{x}^{(i)} \in \mathbb{R}^M, y^{(i)} \in \mathbb{R}$$

***Common Misunderstanding*:**

Linear functions ≠ Linear decision boundaries



$$y = w_1 x_1 + w_2 x_2 + b$$

- A general linear function is
  $$y = \mathbf{w}^T \mathbf{x} + b$$
- A general linear decision boundary is
  $$y = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

Key Idea of Linear Regression

Residuals

Key Idea of Linear Regression

Mean squared error

The Big Picture

OPTIMIZATION FOR ML

Unconstrained Optimization

- *Def*: In **unconstrained optimization**, we try minimize (or maximize) a function with *no constraints* on the inputs to the function

Given a function

$$J(\boldsymbol{\theta}), J : \mathbb{R}^M \rightarrow \mathbb{R}$$

Our goal is to find

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^M}{\operatorname{argmin}} J(\boldsymbol{\theta})$$

For ML, these are the parameters

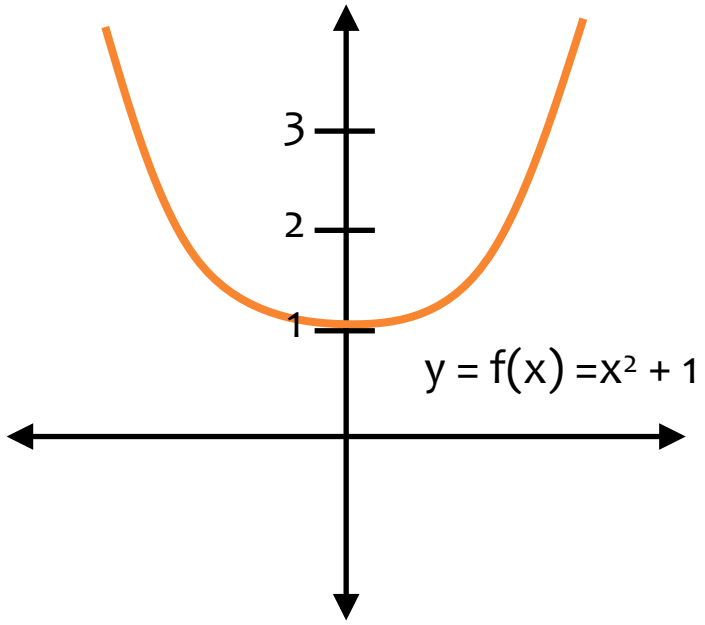
For ML, this is the objective function

Optimization for ML

上海科技大学 ShanghaiTech University

- Not quite the same setting as other fields…
  - Function we are optimizing might not be the true goal (e.g. likelihood vs generalization error)
  - Precision might not matter
    - (e.g. data is noisy, so optimal up to 1e-16 might not help)
  - Stopping early can help generalization error
    - (i.e. “early stopping” is a technique for regularization – discussed more next time)

min vs. argmin



$v^* = \min_x f(x)$

$x^* = \text{argmin}_x f(x)$

1. Question: What is v*?
2. Question: What is x*?

OPTIMIZATION METHOD #0: RANDOM GUESSING

Notation Trick: Folding in the Intercept Term

$$\mathbf{x}' = [1, x_1, x_2, \ldots, x_M]^T$$

$$\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$$

Notation Trick: fold the bias *b* and the weights ***w*** into a single vector **θ** by prepending a constant to ***x*** and increasing dimensionality by one!

$$h_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}') = \boldsymbol{\theta}^T\mathbf{x}'$$

This convenience trick allows us to more compactly talk about linear functions as a simple dot product (without explicitly writing out the intercept term every time).

Linear Regression as Function Approximation

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$
where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

1. Assume $\mathcal{D}$ generated as:

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$
$$y^{(i)} = h^*(\mathbf{x}^{(i)})$$

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} e_i^2$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)^2$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)})\right)^2$$

4. Solve the unconstrained optimization problem via favorite method:
   - *gradient descent*
   - *closed form*
   - *stochastic gradient descent*
   - ...

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T\mathbf{x}$$

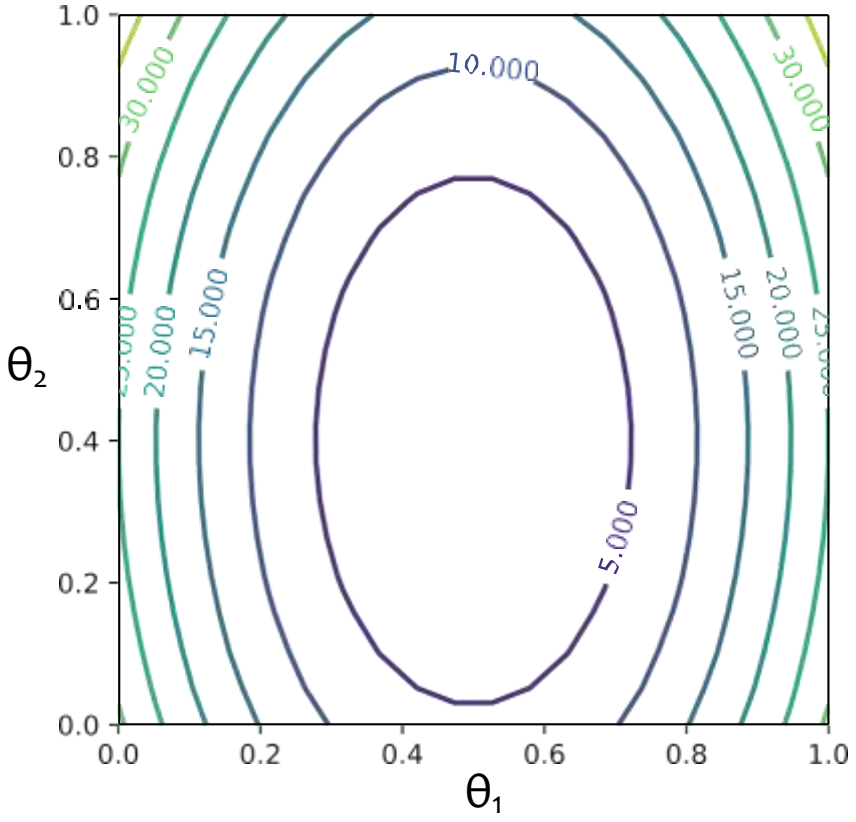
Contour Plots

上海科技大学
ShanghaiTech University

Contour Plots

1. Each level curve labeled with value
2. Value label indicates the value of the function for all points lying on that level curve
3. Just like a topographical map, but for a function

$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = (10(\theta_1 - 0.5))^2 + (6(\theta_1 - 0.4))^2$

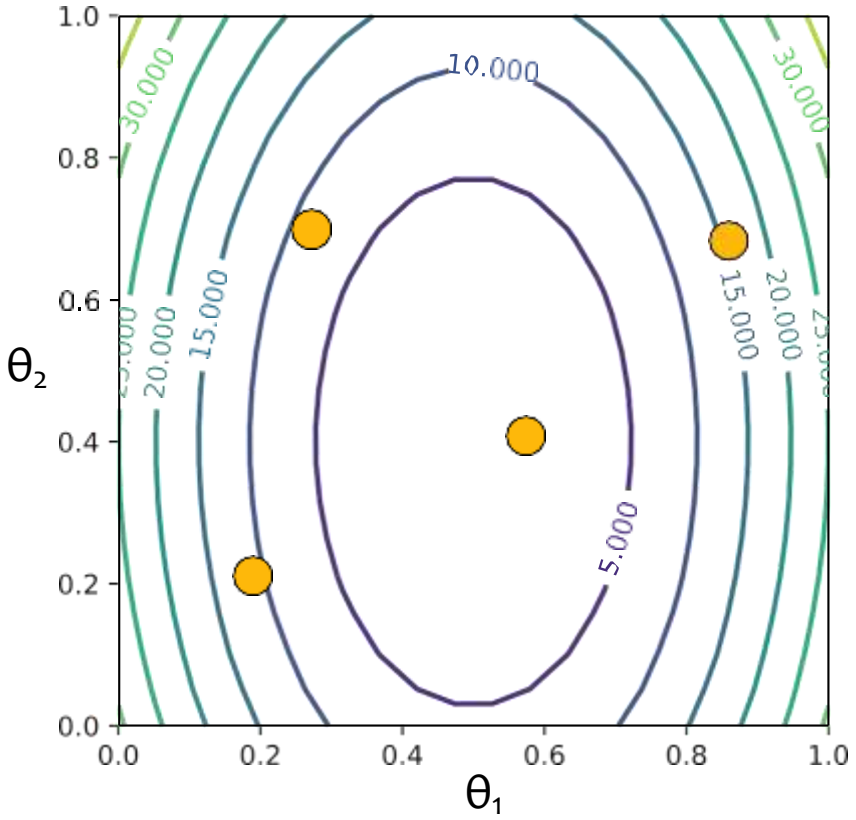


Optimization by Random Guessing

Optimization Method #0: Random Guessing

1. Pick a random $\boldsymbol{\theta}$
2. Evaluate $J(\boldsymbol{\theta})$
3. Repeat steps 1 and 2 many times
4. Return $\boldsymbol{\theta}$ that gives smallest $J(\boldsymbol{\theta})$

$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = (10(\theta_1 - 0.5))^2 + (6(\theta_1 - 0.4))^2$



t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.2	0.2	10.4
2	0.3	0.7	7.2
3	0.6	0.4	1.0
4	0.9	0.7	16.2

Optimization by Random Guessing

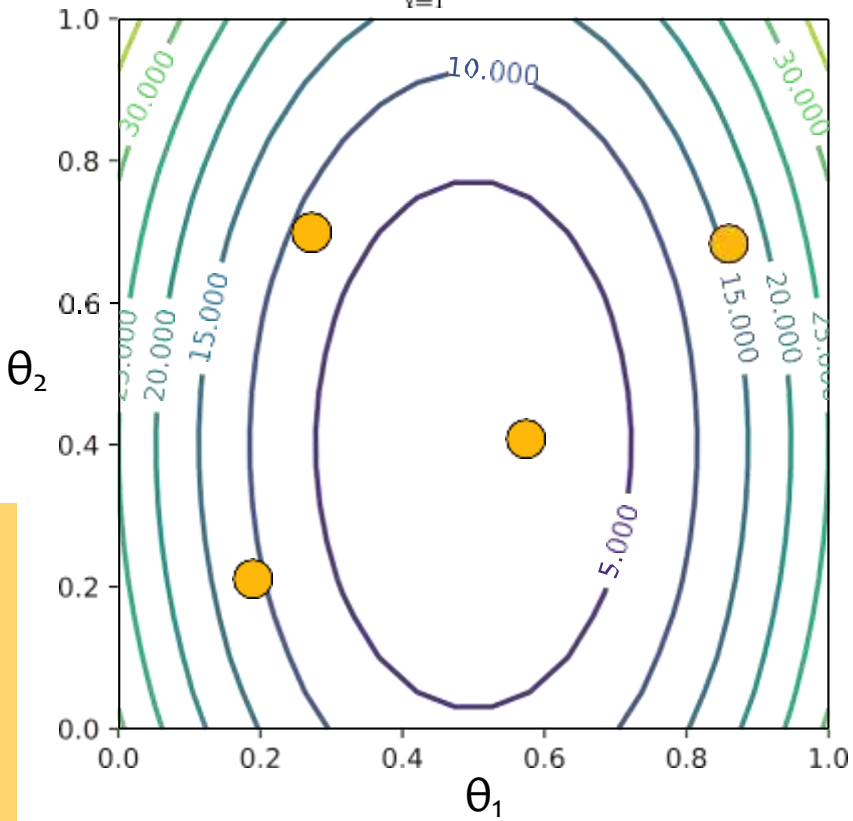
Optimization Method #0: Random Guessing

1. Pick a random **θ**
2. Evaluate J(**θ**)
3. Repeat steps 1 and 2 many times
4. Return **θ** that gives smallest J(**θ**)

For Linear Regression:

- **objective function** is Mean Squared Error (MSE)
- MSE = J(w, b)
  $= J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2$
- contour plot: each line labeled with MSE – **lower means a better fit**
- **minimum** corresponds to parameters (w,b) = (θ_1, θ_2) that **best fit** some training dataset

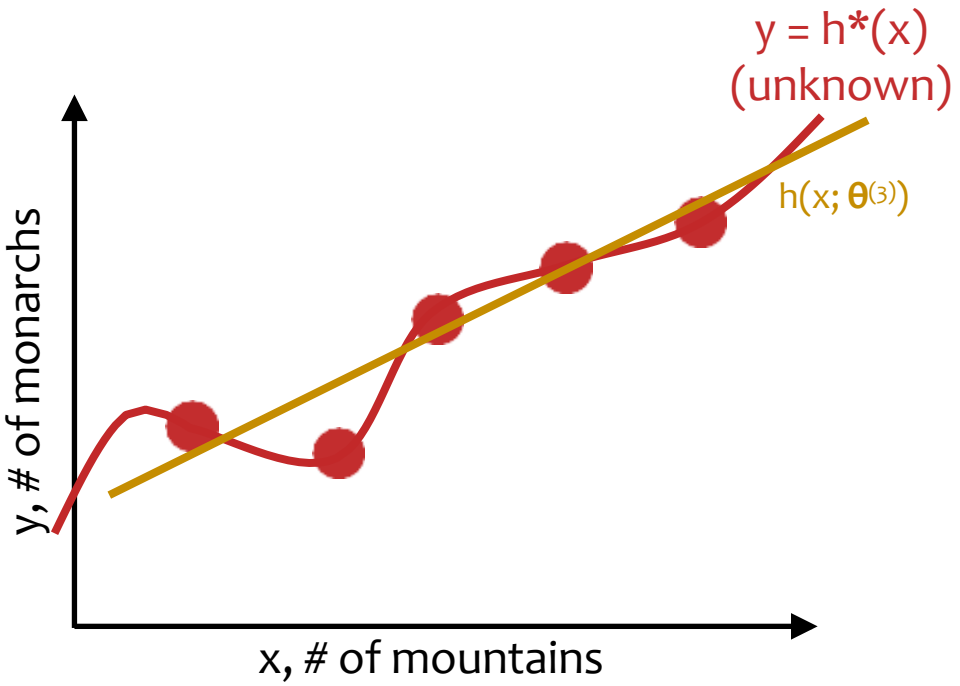
$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2$



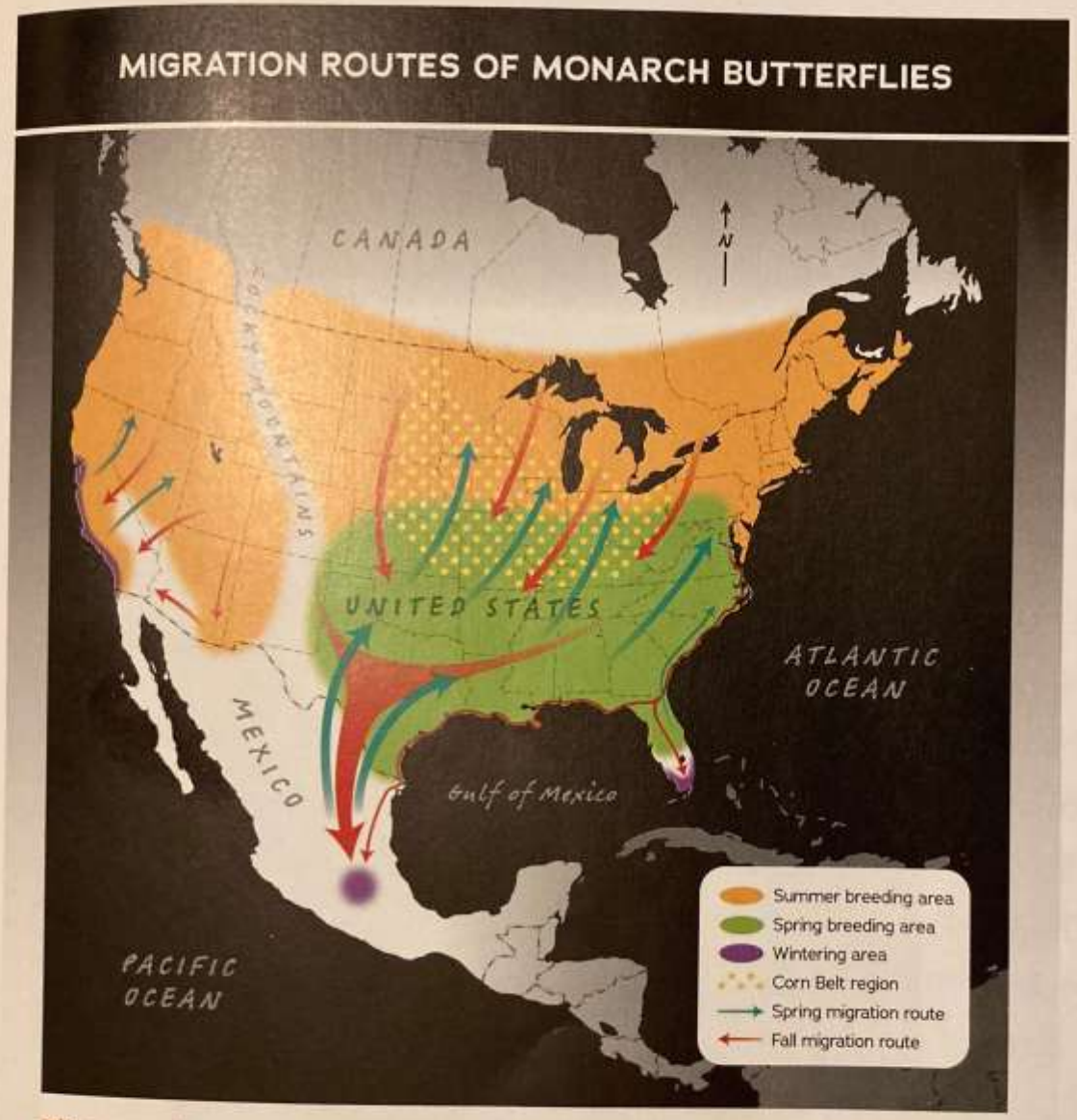
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.2	0.2	10.4
2	0.3	0.7	7.2
3	0.6	0.4	1.0
4	0.9	0.7	16.2

Linear Regression: Running Example

Counting Butterflies



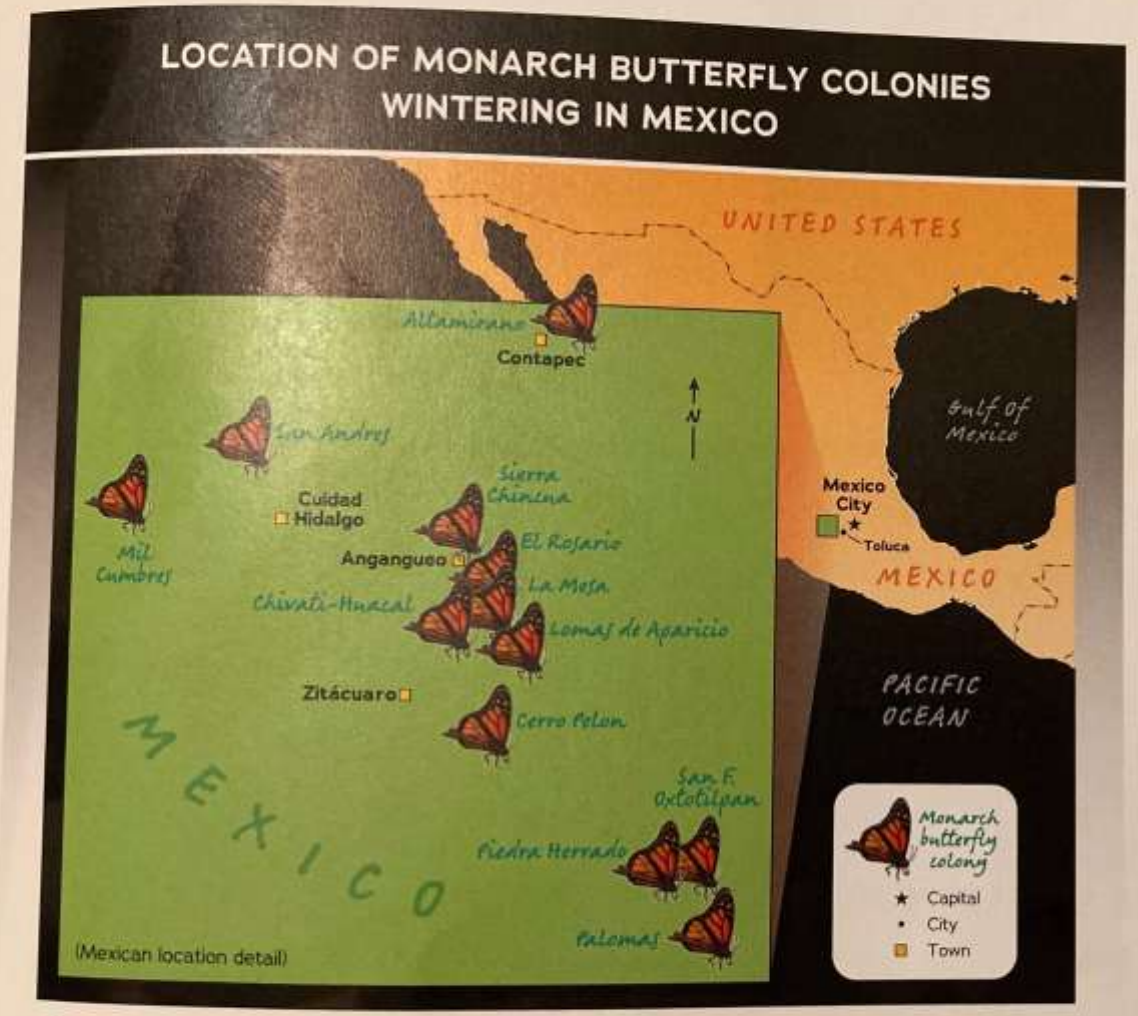
breed only in places where milkweed grows. When the last ice age ended, and the cold and glaciers retreated, milkweed may have gradually spread northward, and monarchs may have followed. But the monarch butterfly remained a tropical creature, unable to survive the severe northern winters. So every year as winter approached, monarchs left their summer fields of milkweed and flew south again. To this day, every spring and summer, monarchs travel north to their breeding grounds across the eastern United States and Canada. Every winter, they return to Mexico.



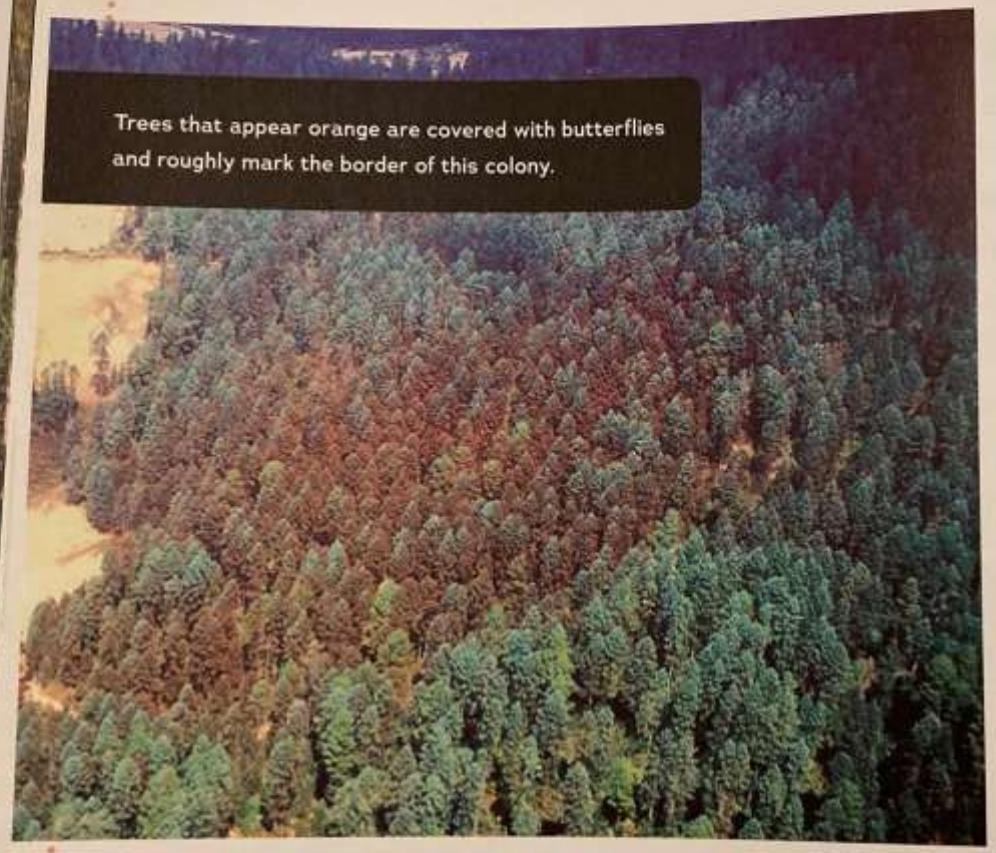
This map shows migration routes of fall and spring migrations, both east and west of the Rocky Mountains.

Researchers began taking measurements in 1993. The highest year on record came in 1997, when the colonies covered about 45 acres (18 ha), an area equal to about thirty-four football fields. Scientists aren't sure exactly how many butterflies that represented, but one estimate is that there were one billion monarchs in the colonies that winter.

But as researchers measured the colonies year after year, they noticed that the colonies were shrinking. By 2014 the colonies measured just 1.7 acres (0.7 ha), or less than one and a half football fields. That year there may have been only about thirty-five million monarchs in the colonies.



The eastern monarchs migrate to just twelve mountaintops, all located in central Mexico.



Trees that appear orange are covered with butterflies and roughly mark the border of this colony.

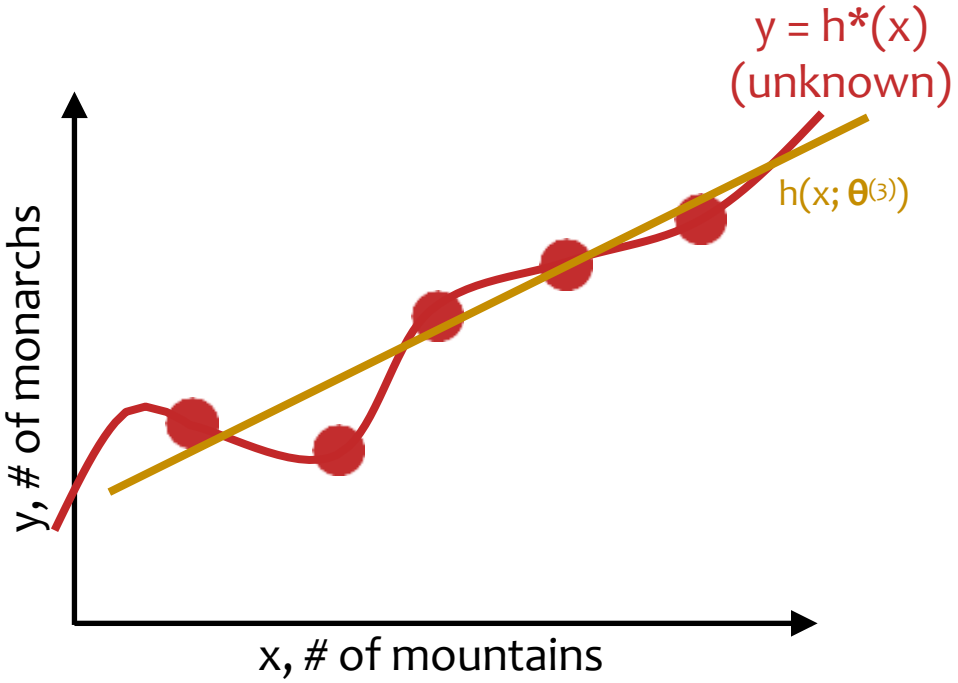
Many scientists were worried. The population of eastern monarchs had dropped more than 90 percent in just seventeen years.

At the same time, scientists in California reported that the number of western monarchs was dropping as well. From 1997 to 2014, the number of monarchs overwintering along the California coast had fallen by 74 percent.

Populations of overwintering monarchs were falling fast. By 2014 their numbers had fallen so far that people wondered whether the monarch butterfly should be listed as an endangered species—a species in danger of becoming extinct, or disappearing forever.

Losing monarchs could be bad for our world because monarchs play an important part in the food web. Despite the milkweed toxins in their bodies, they are food for songbirds, spiders, and insects. Monarchs visit many flowers and act as pollinators.

Counting Butterflies



Linear Regression in High Dimensions

- In our discussions of linear regression, we will always assume there is just one output, y
- But our inputs will usually have many features:

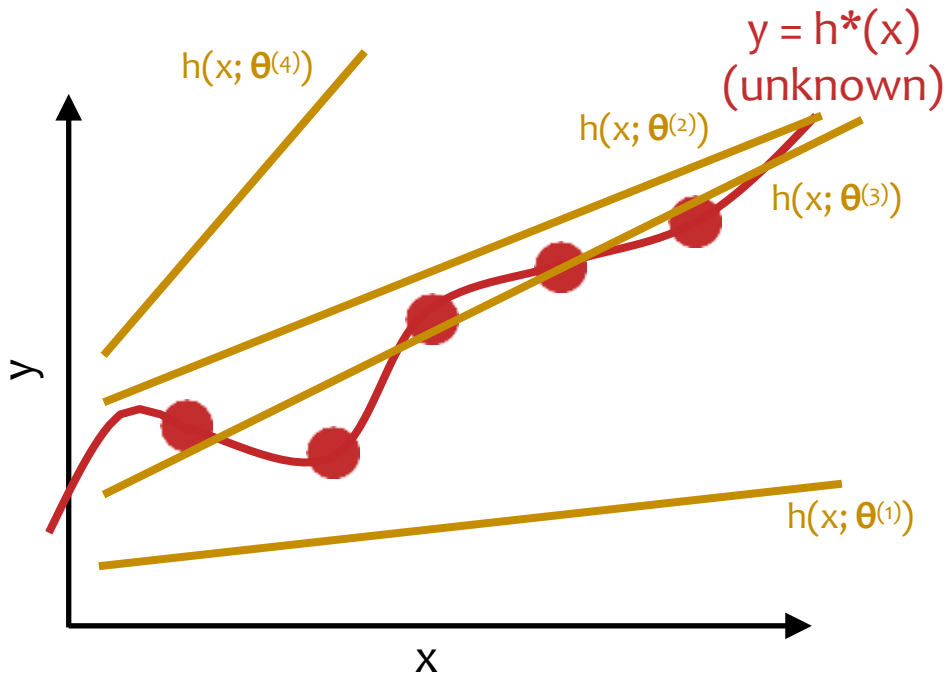
  $\mathbf{x} = [x_1, x_2, \ldots, x_M]^T$
- For example:
  - suppose we had a drone take pictures of each section of forest
  - each feature could correspond to a pixel in this image such that $x_m = 1$ if the pixel is orange and $x_m = 0$ otherwise
  - the output y would be the number of butterflies in each picture

Q: How would you obtain ground truth data?

Linear Regression by Rand. Guessing

Optimization Method #0: Random Guessing

1. Pick a random θ
2. Evaluate $J(\theta)$
3. Repeat steps 1 and 2 many times
4. Return θ that gives smallest $J(\theta)$



For Linear Regression:

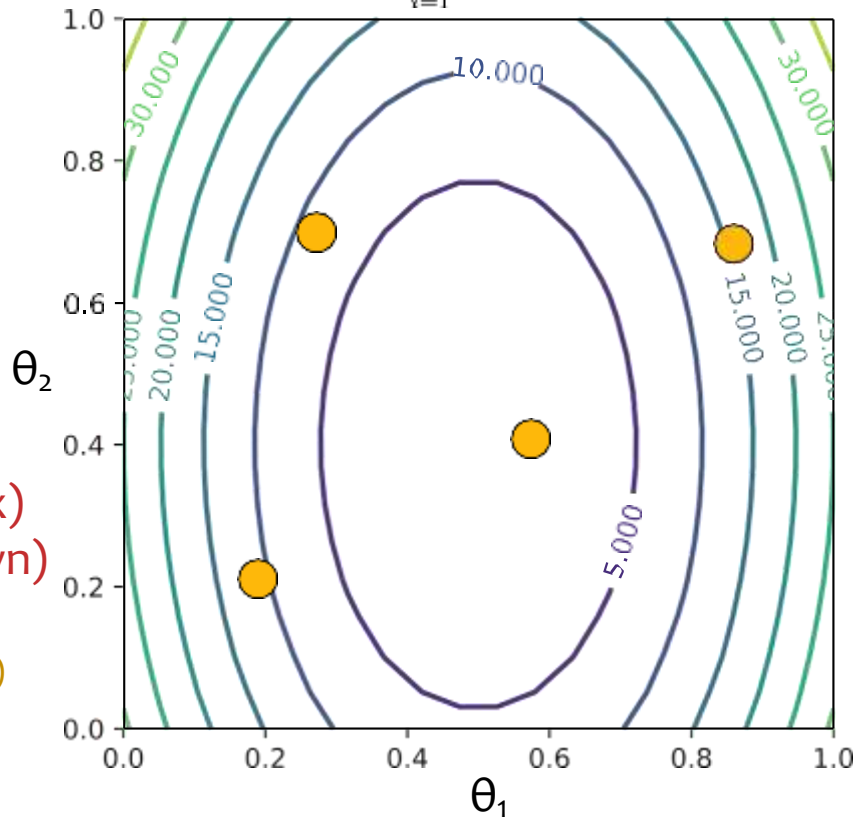
- target function h*(x) is **unknown**
- only have access to h*(x) through **training examples** $(x^{(i)},y^{(i)})$
- want $h(x;\theta^{(t)})$ that **best approximates** h*(x)
- **enable generalization** w/inductive bias that restricts hypothesis class to **linear functions**

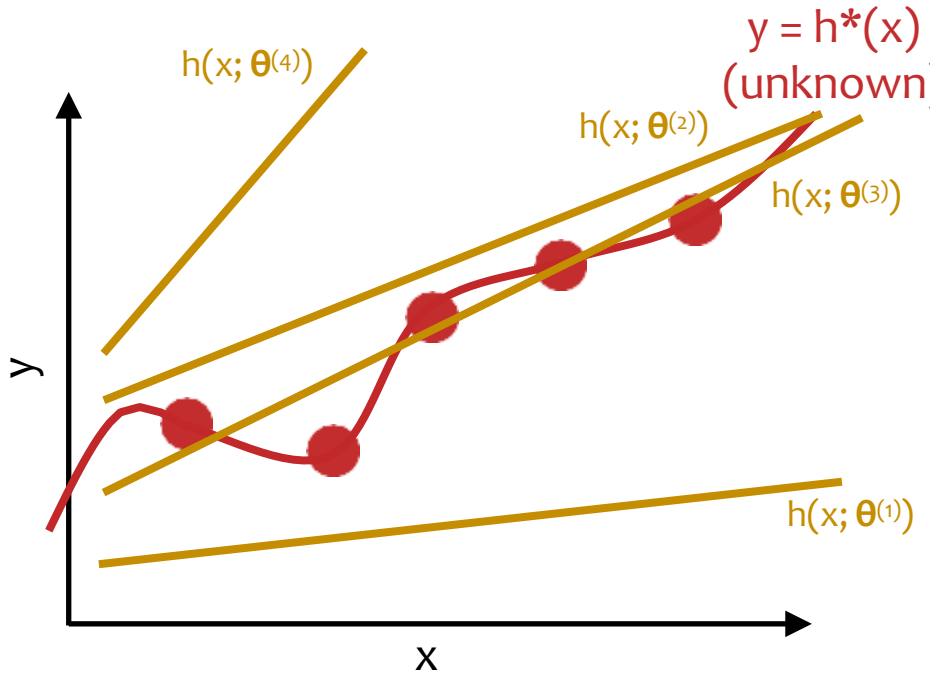
Linear Regression by Rand. Guessing

Optimization Method #0: Random Guessing

1. Pick a random **θ**
2. Evaluate J(**θ**)
3. Repeat steps 1 and 2 many times
4. Return **θ** that gives smallest J(**θ**)

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2$$





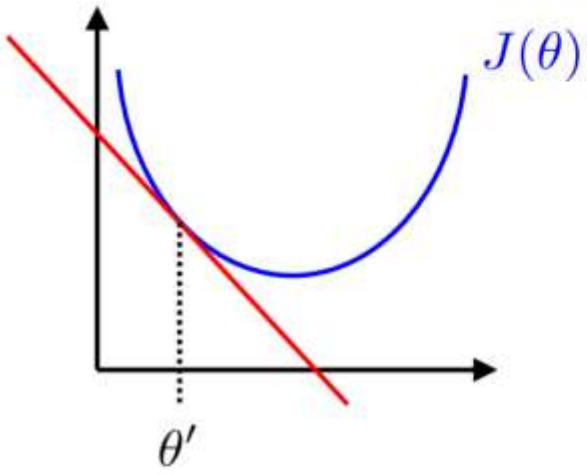
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.2	0.2	10.4
2	0.3	0.7	7.2
3	0.6	0.4	1.0
4	0.9	0.7	16.2

OPTIMIZATION METHOD #1: GRADIENT DESCENT

Derivatives

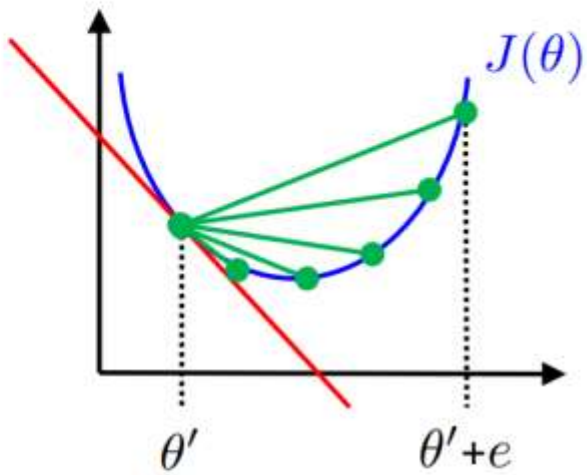
上海科技大学
ShanghaiTech University

1. Derivative as a Slope



slope = $\frac{\partial J(\theta)}{\partial \theta}$

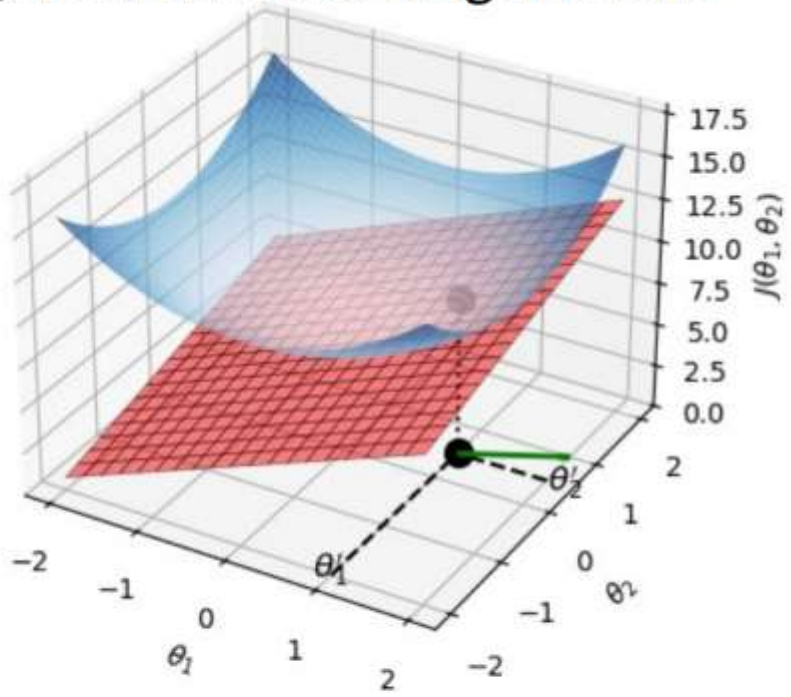
2. Derivative as a Limit



$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \lim_{e \to 0} \frac{J(\boldsymbol{\theta} + e) - J(\boldsymbol{\theta})}{e}$$

The limit of the secants is the tangent

3. Derivative as a Tangent Plane



$$\begin{bmatrix} \frac{\partial J(\theta_1,\theta_2)}{\partial \theta_1} \\ \frac{\partial J(\theta_1,\theta_2)}{\partial \theta_2} \end{bmatrix}$$

Gradient

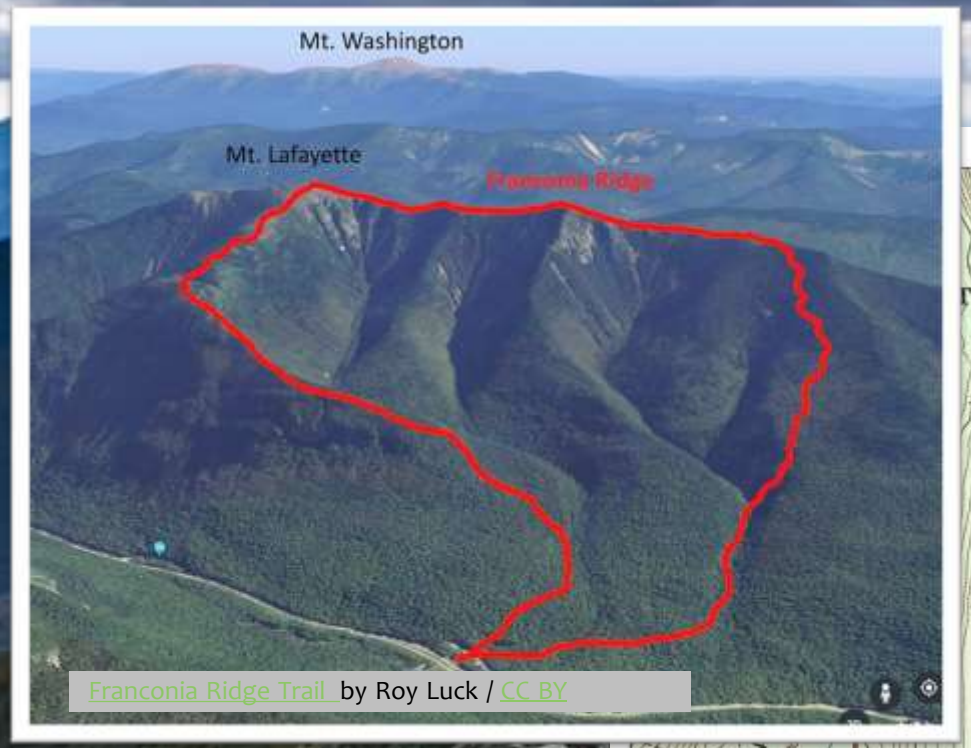
Def: The gradient of $J : \mathbb{R}^M \to \mathbb{R}$ is

$$\nabla J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_M} \end{bmatrix}$$

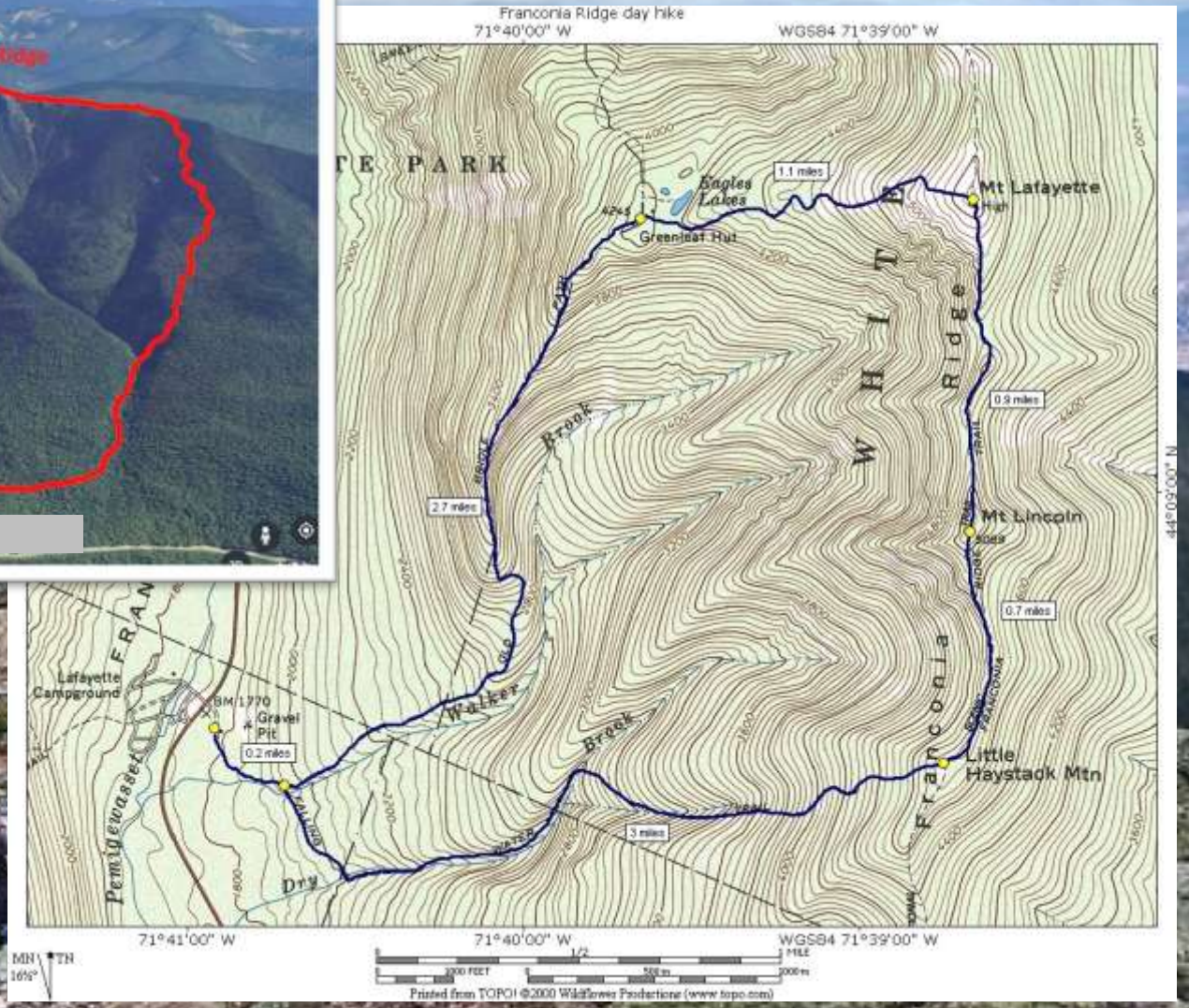
Each entry is a first-order partial derivative

Topographical Maps

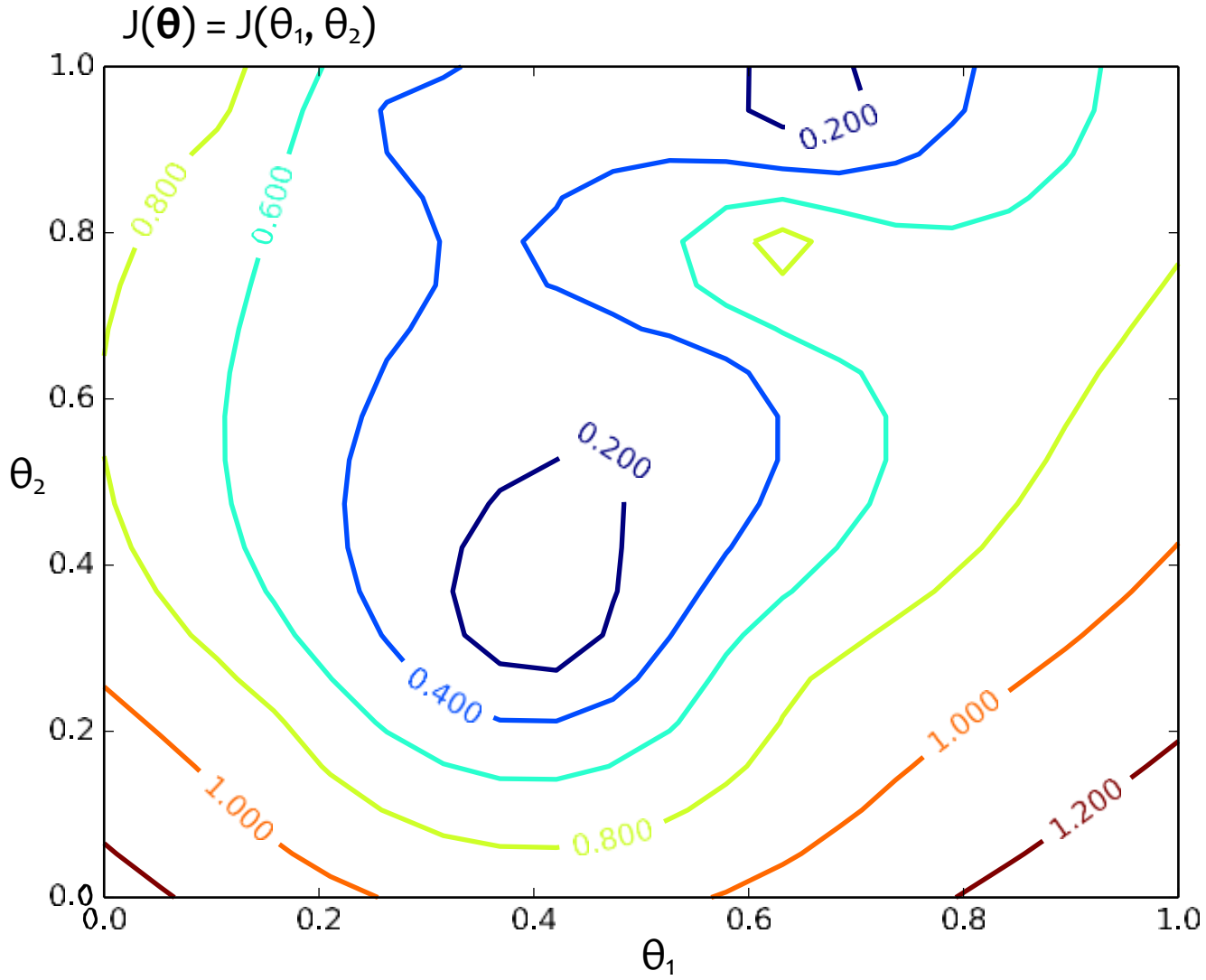
Topographical Maps



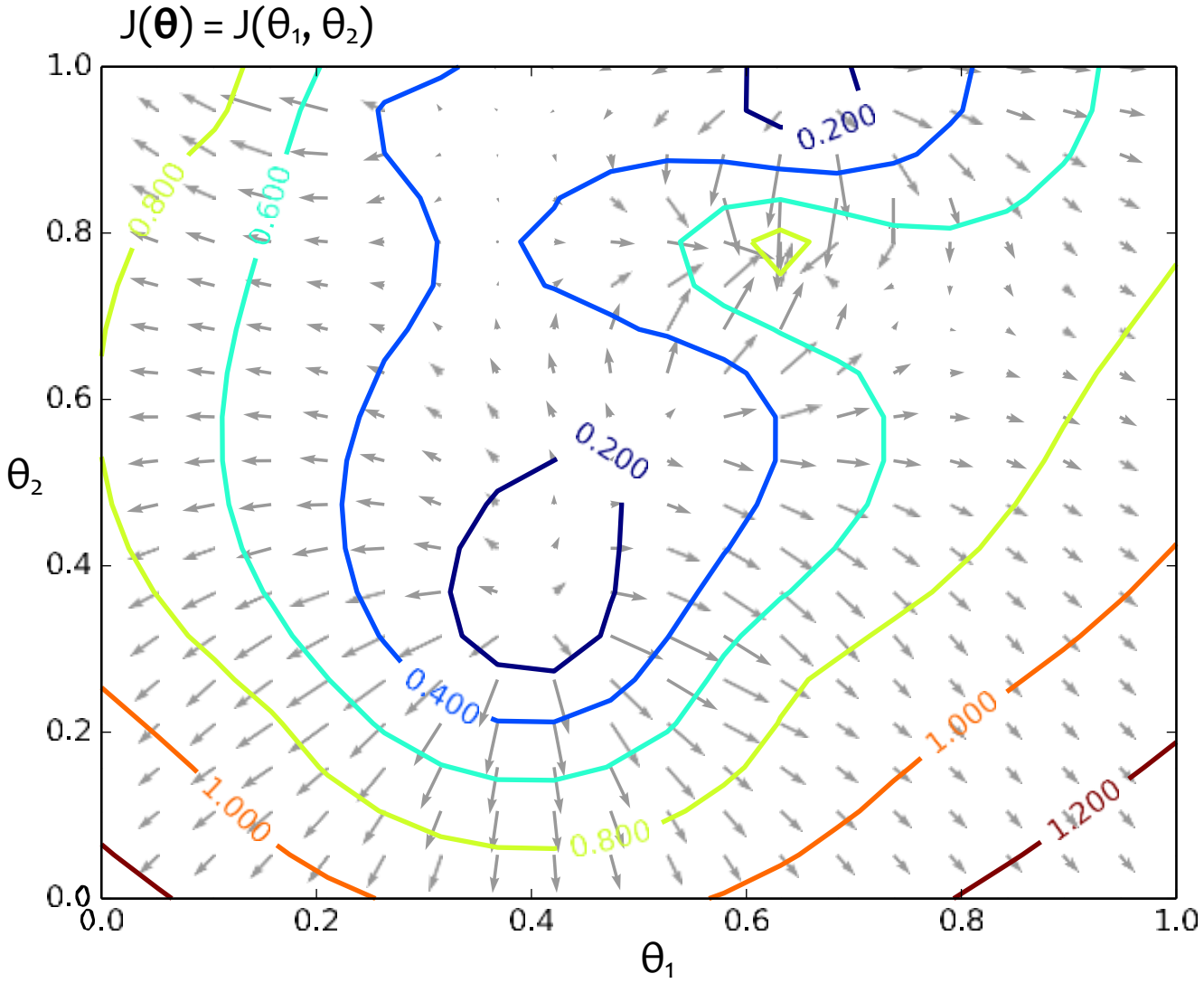
Franconia Ridge Trail by Roy Luck / CC BY



Gradients

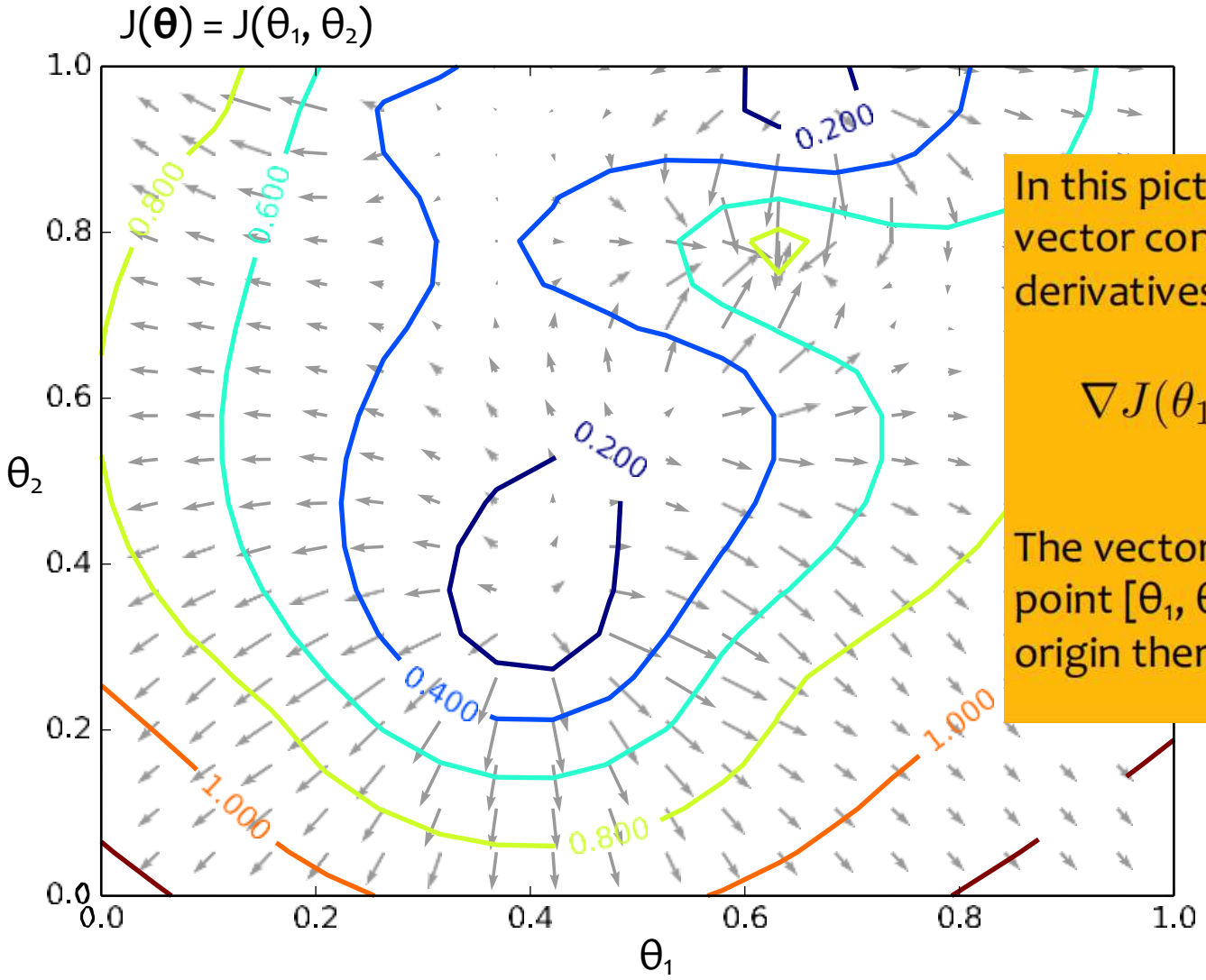

J(θ) = J(θ1, θ2)
1.0
0.8
0.6
0.4
0.2
0.0
θ2
0.0
0.2
0.4
0.6
0.8
1.0
θ1
0.800
0.600
0.200
0.200
0.400
1.000
0.800
1.000
1.200

Gradients



These are the **gradients** that Gradient **Ascent** would follow.

Gradients



In this picture, each arrow is a 2D vector consisting of two partial derivatives.

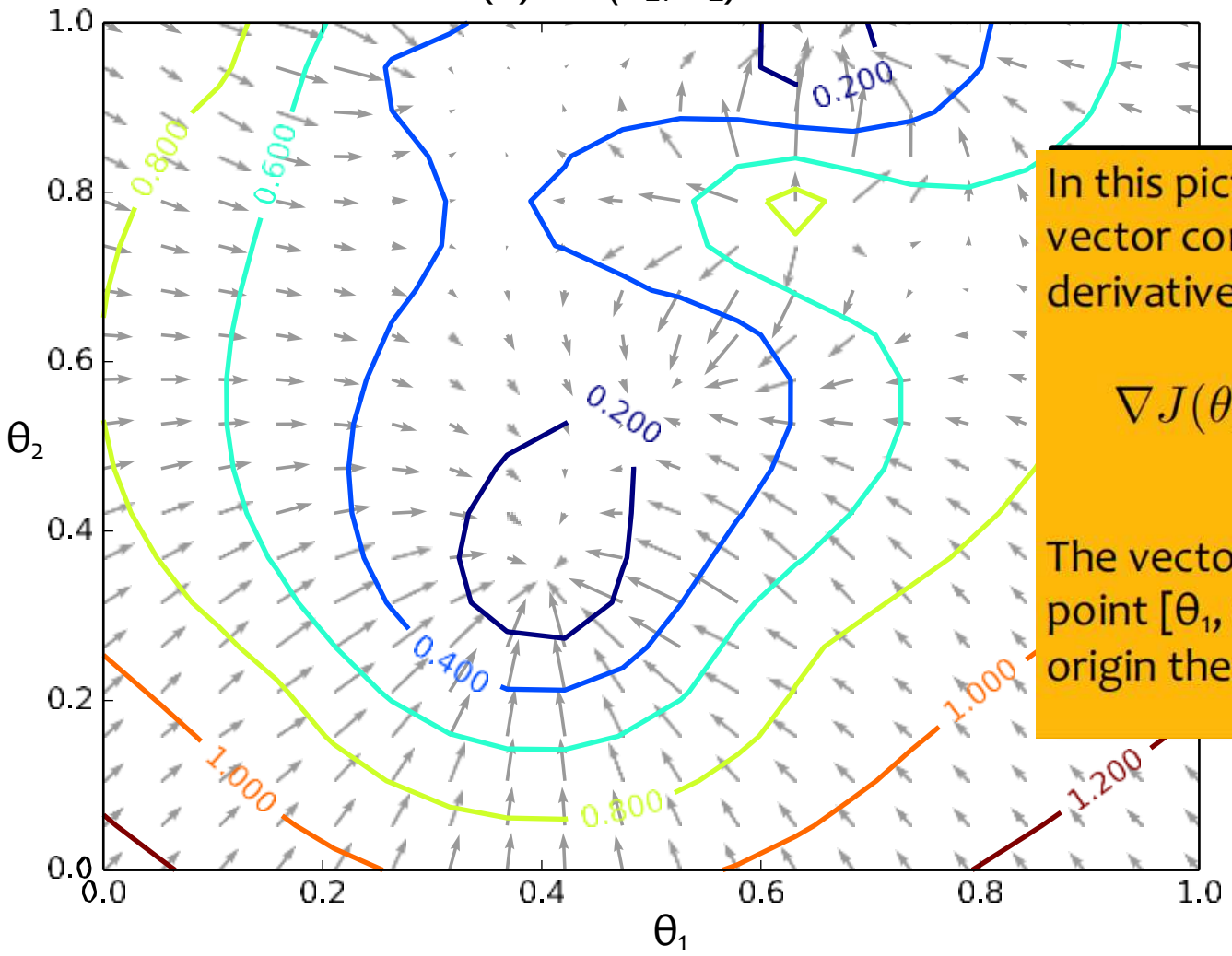
$$\nabla J(\theta_1, \theta_2) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \\ \frac{\partial J}{\partial \theta_2} \end{bmatrix}$$

The vector is evaluated at the point $[\theta_1, \theta_2]^T$ and plotted with its origin there as well.

These are the **gradients** that Gradient **Ascent** would follow.

(Negative) Gradients

$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2)$



In this picture, each arrow is a 2D vector consisting of two partial derivatives.

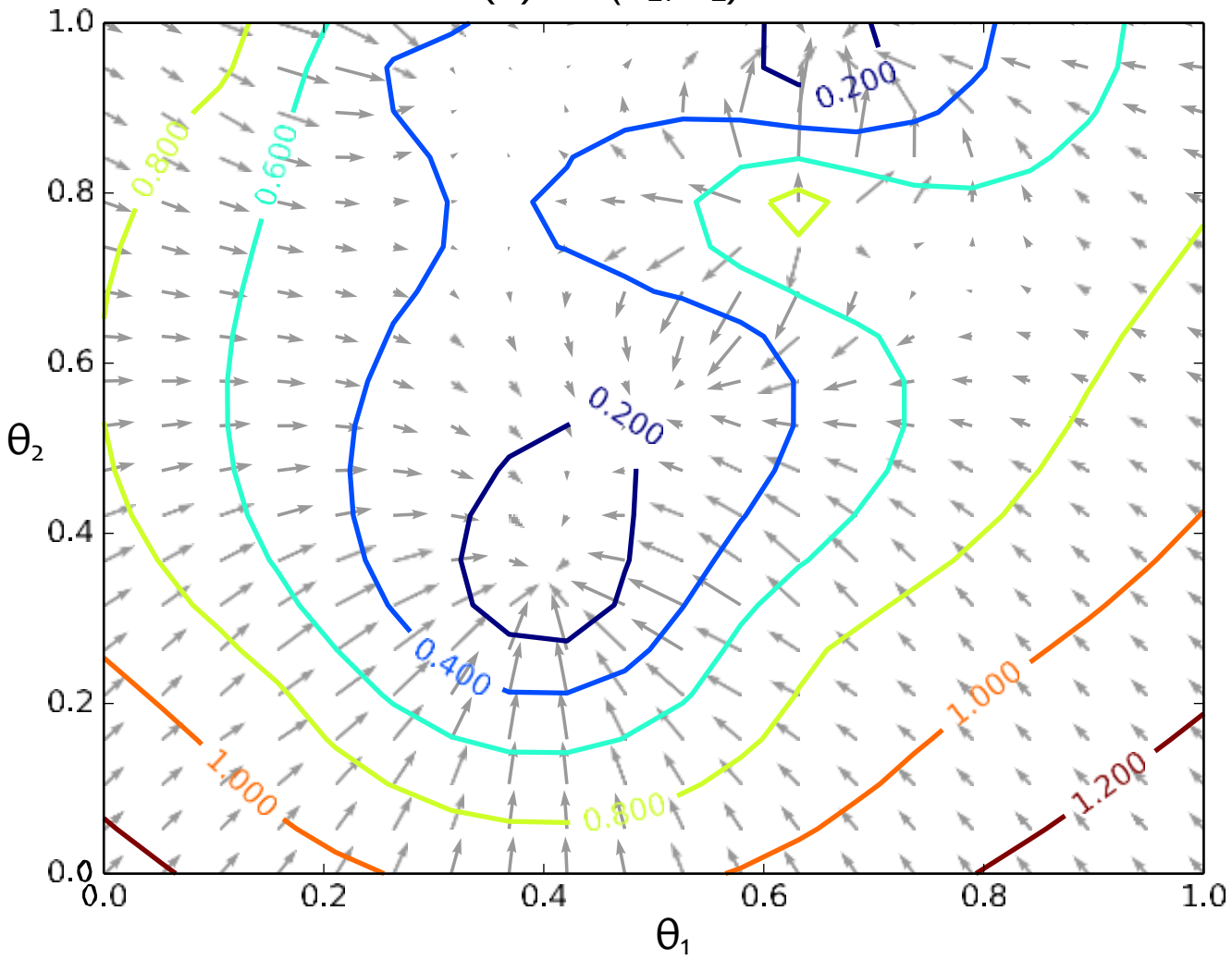
$$\nabla J(\theta_1, \theta_2) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \\ \frac{\partial J}{\partial \theta_2} \end{bmatrix}$$

The vector is evaluated at the point $[\theta_1, \theta_2]^T$ and plotted with its origin there as well.

These are the **negative** gradients that Gradient **Descent** would follow.

(Negative) Gradients

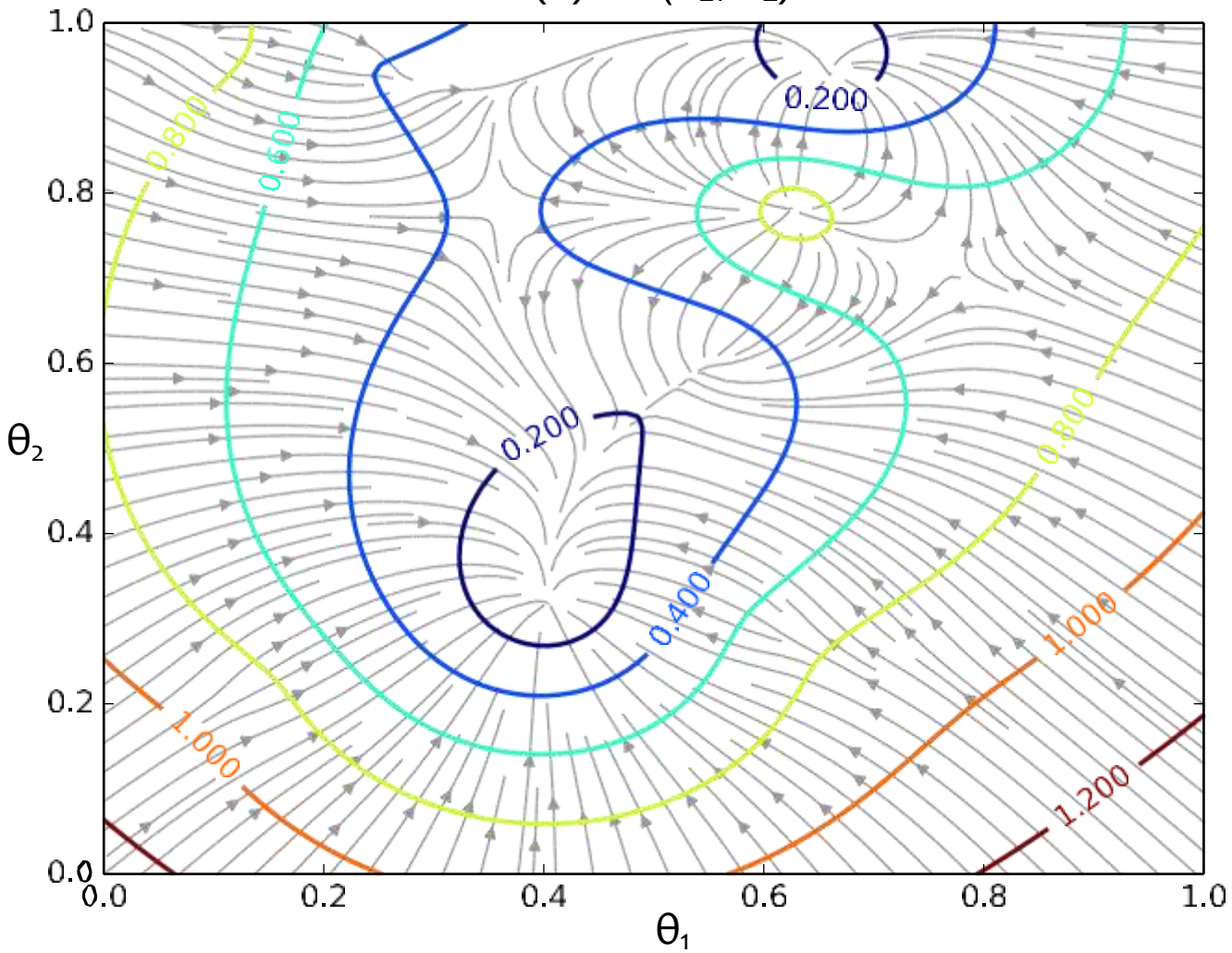
$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2)$



These are the **negative** gradients that Gradient **Descent** would follow.

(Negative) Gradient *Paths*

$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2)$



Shown are the **paths** that Gradient Descent would follow if it were making **infinitesimally small steps.**

Gradient Descent

上海科技大学
ShanghaiTech University

Gradient Descent Algorithm

Remarks

Gradient Descent: Step Size

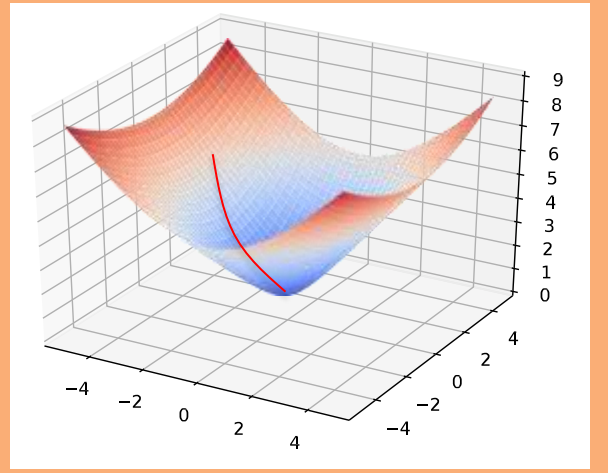
Question:
In gradient descent, what could go wrong if we *always* use the same step size (or step size schedule) for every problem we encounter?

Answer:

Gradient Descent

Algorithm 1 Gradient Descent

1: **procedure** GD(D, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
5: $\quad$**return** $\boldsymbol{\theta}$



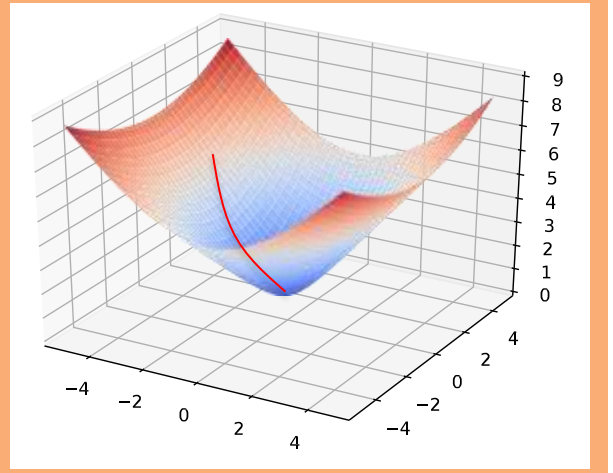
In order to apply GD to Linear Regression all we need is the **gradient** of the objective function (i.e. vector of partial derivatives).

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix}$$

Gradient Descent

Algorithm 1 Gradient Descent

1: **procedure** GD(D, $\theta^{(0)}$)
2: $\theta \leftarrow \theta^{(0)}$
3: **while** not converged **do**
4: $\theta \leftarrow \theta - \gamma \nabla_{\theta} J(\theta)$
5: **return** θ



There are many possible ways to detect **convergence**. For example, we could check whether the L2 norm of the gradient is below some small tolerance.

$$||\nabla_{\theta} J(\theta)||_2 \leq \epsilon$$

Alternatively we could check that the reduction in the objective function from one iteration to the next is small.

GRADIENT DESCENT FOR LINEAR REGRESSION

Linear Regression as Function Approximation

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$
where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

1. Assume $\mathcal{D}$ generated as:

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$
$$y^{(i)} = h^*(\mathbf{x}^{(i)})$$

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} e_i^2$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)^2$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)})\right)^2$$

4. Solve the unconstrained optimization problem via favorite method:

- *gradient descent*
- *closed form*
- *stochastic gradient descent*
- ...

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

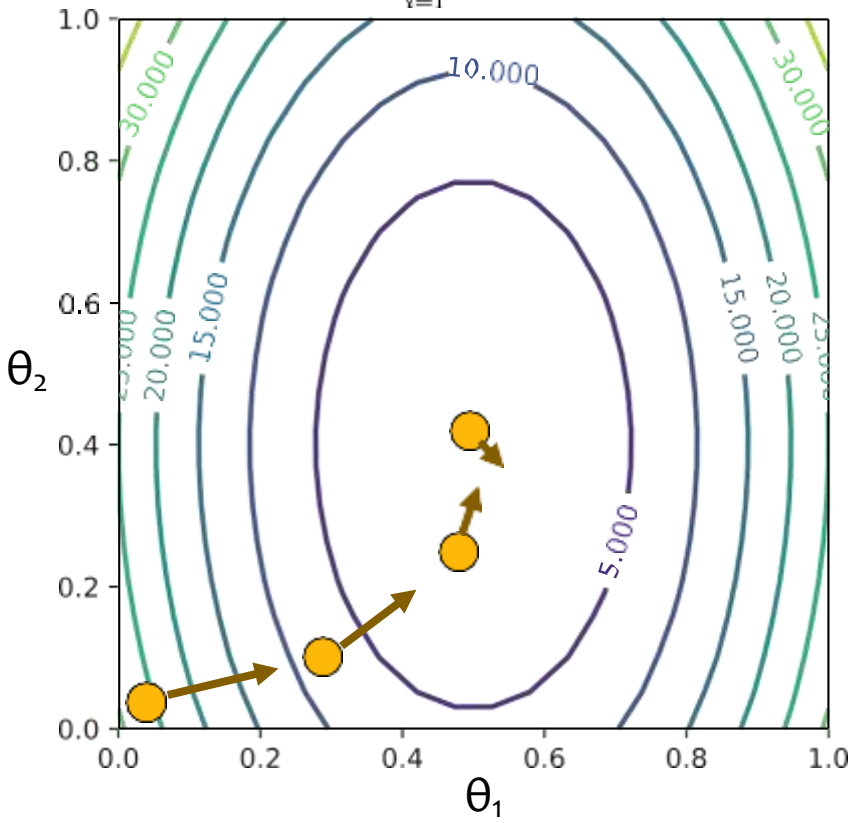
$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T\mathbf{x}$$

Linear Regression by Gradient Descent

Optimization Method #1: Gradient Descent

1. Pick a random **θ**
2. Repeat:
   a. Evaluate gradient ∇J(**θ**)
   b. Step opposite gradient
3. Return **θ** that gives smallest J(**θ**)

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2$$

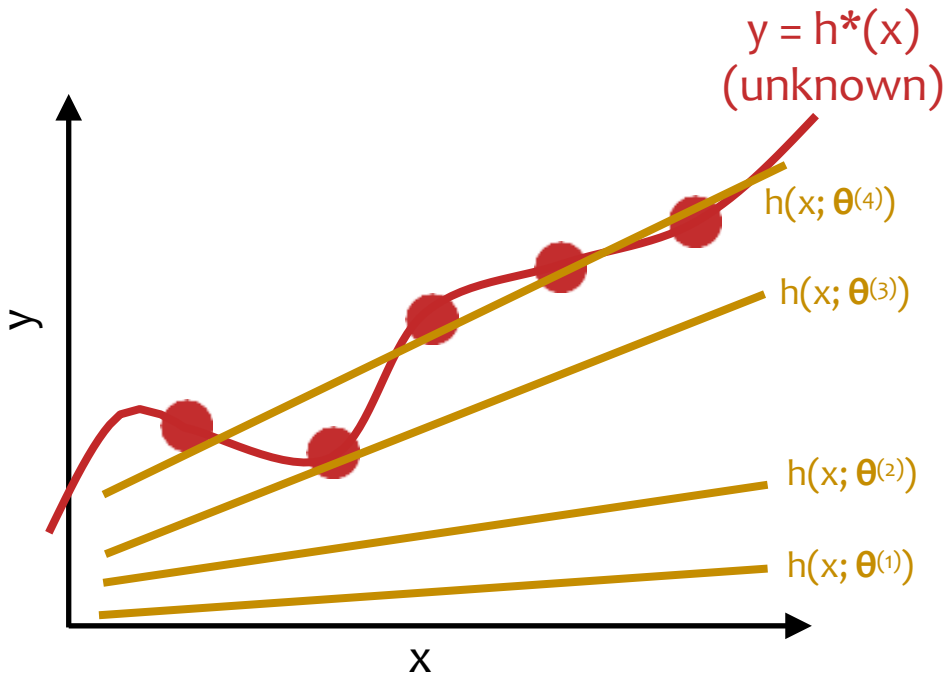


t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Linear Regression by Gradient Descent

Optimization Method #1: Gradient Descent

1. Pick a random θ
2. Repeat:
   a. Evaluate gradient $\nabla J(\theta)$
   b. Step opposite gradient
3. Return θ that gives smallest $J(\theta)$



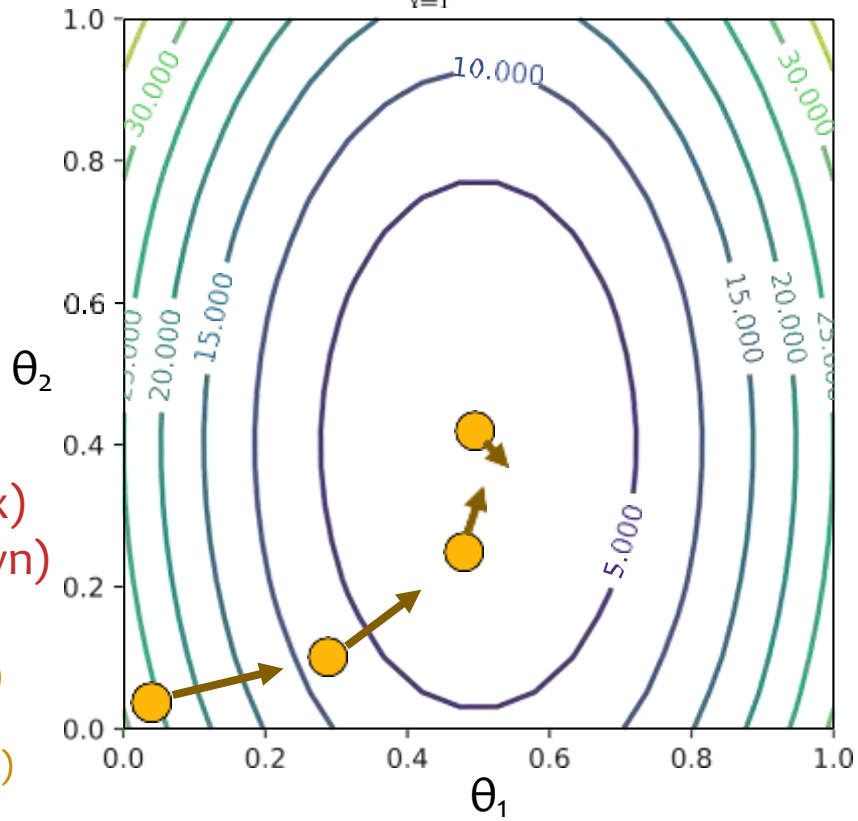
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

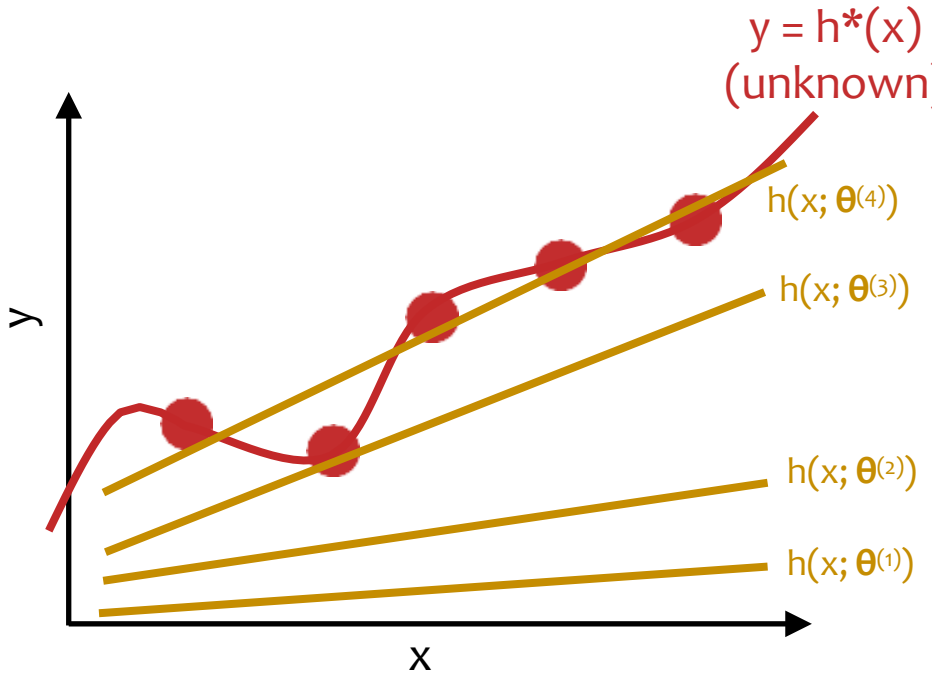
Linear Regression by Gradient Descent

Optimization Method #1: Gradient Descent

1. Pick a random **θ**
2. Repeat:
   a. Evaluate gradient ∇J(**θ**)
   b. Step opposite gradient
3. Return **θ** that gives smallest J(**θ**)

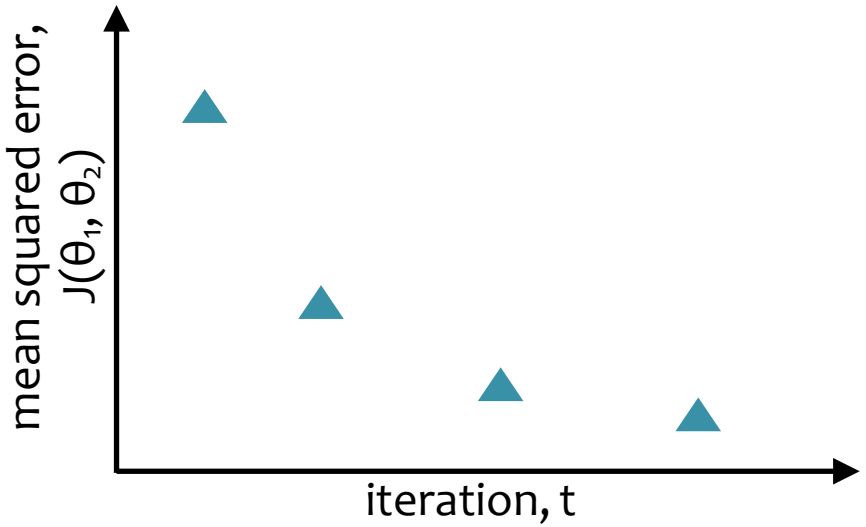
$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2$$

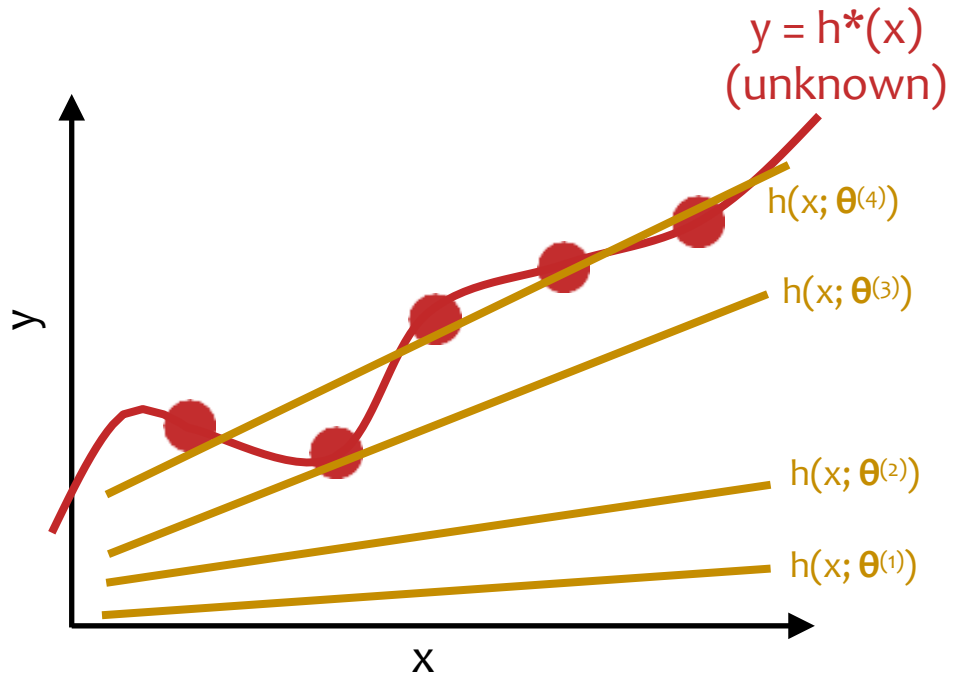




t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

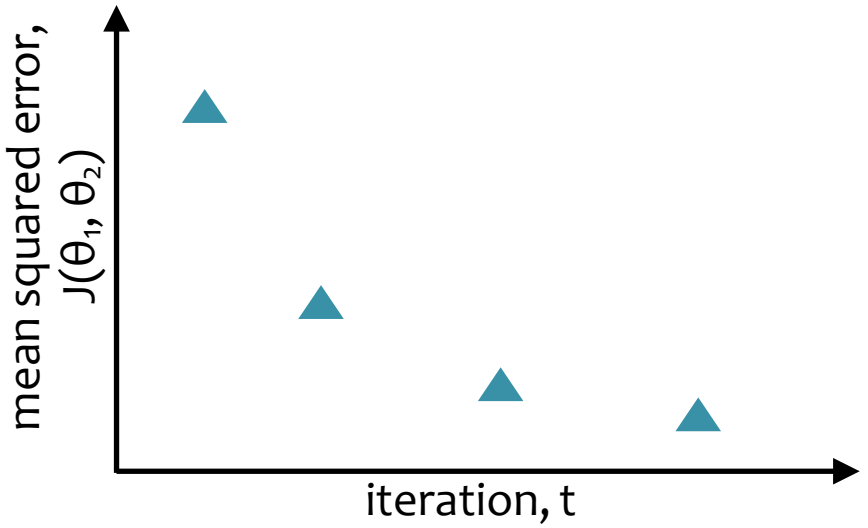
Linear Regression by Gradient Descent



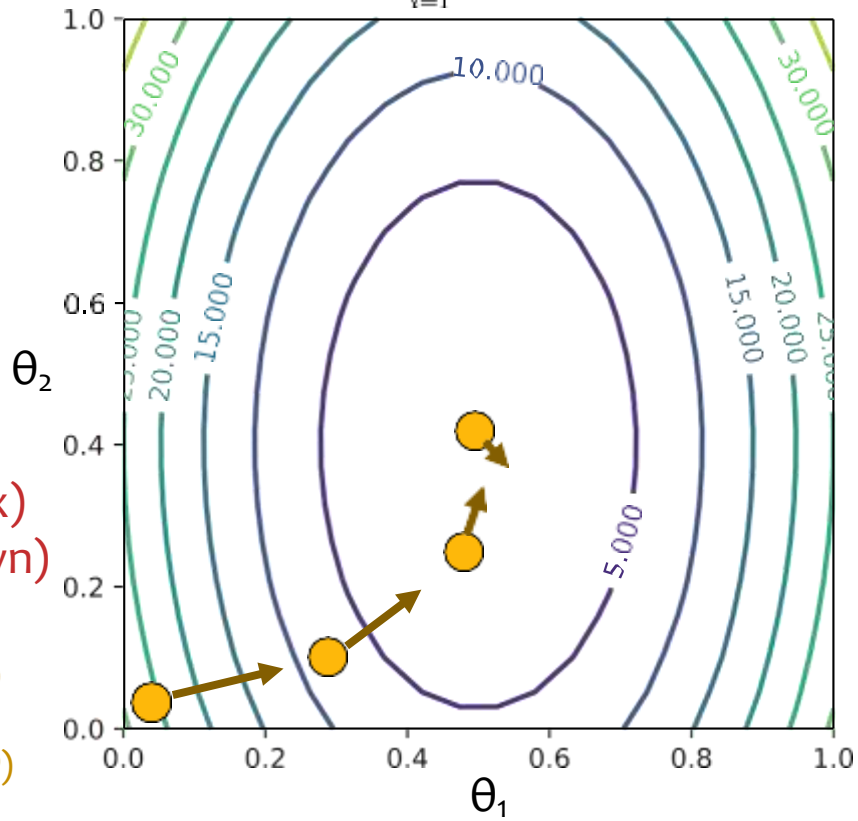


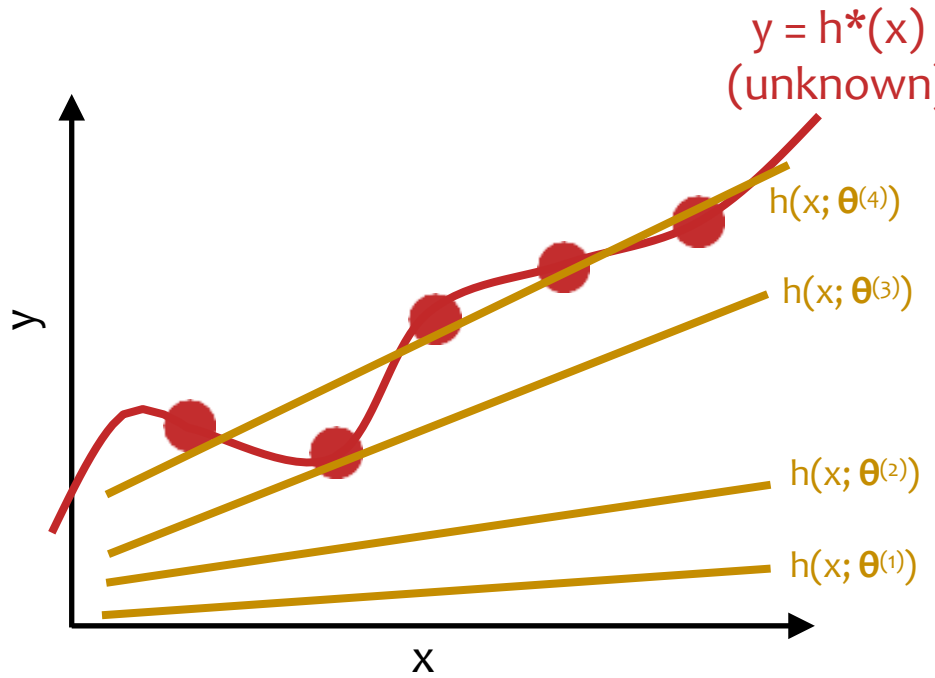
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Linear Regression by Gradient Desc.



$$J(\theta) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \theta^T \mathbf{x}^{(i)}\right)^2$$





t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Gradient Calculation for Linear Regression

Gradient Calculation for Linear Regression

上海科技大学 ShanghaiTech University

Derivative of $J^{(i)}(\boldsymbol{\theta})$:

$$\begin{aligned}
\frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) &= \frac{d}{d\theta_k} \frac{1}{2} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\
&= \frac{1}{2} \frac{d}{d\theta_k} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k} \left(\sum_{j=1}^{K} \theta_j x_j^{(i)} - y^{(i)} \right) \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}
\end{aligned}$$

Derivative of $J(\boldsymbol{\theta})$:

$$\begin{aligned}
\frac{d}{d\theta_k} J(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}
\end{aligned}$$

Gradient of $J(\boldsymbol{\theta})$ [used by Gradient Descent]

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \vdots \\ \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_M^{(i)} \end{bmatrix} \\
&= \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}
\end{aligned}$$

GD for Linear Regression

Gradient Descent for Linear Regression repeatedly takes steps opposite the gradient of the objective function

Algorithm 1 GD for Linear Regression

1: **procedure** GDLR($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
3: $\quad$**while** not converged **do**
4: $\quad\quad\mathbf{g} \leftarrow \sum_{i=1}^{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})\mathbf{x}^{(i)}$ ▷ Compute gradient
5: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma\mathbf{g}$ ▷ Update parameters
6: $\quad$**return** $\boldsymbol{\theta}$

Support Vector Regression

Support Vector Machines (SVMs)

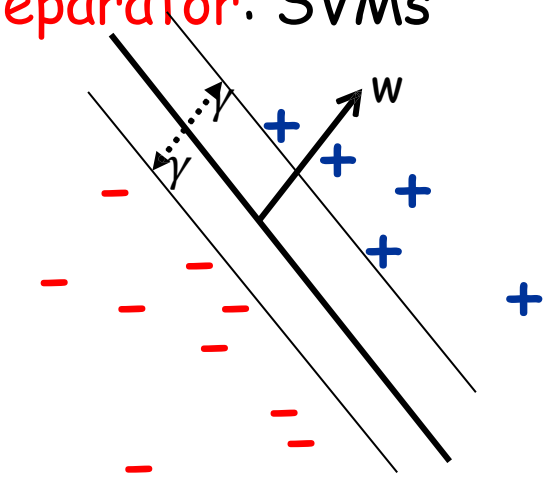
Recall...

Directly optimize for the maximum margin separator: SVMs

Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Maximize γ under the constraint:

- $||w||^2 = 1$
- For all i, $y_i w \cdot x_i \geq \gamma$

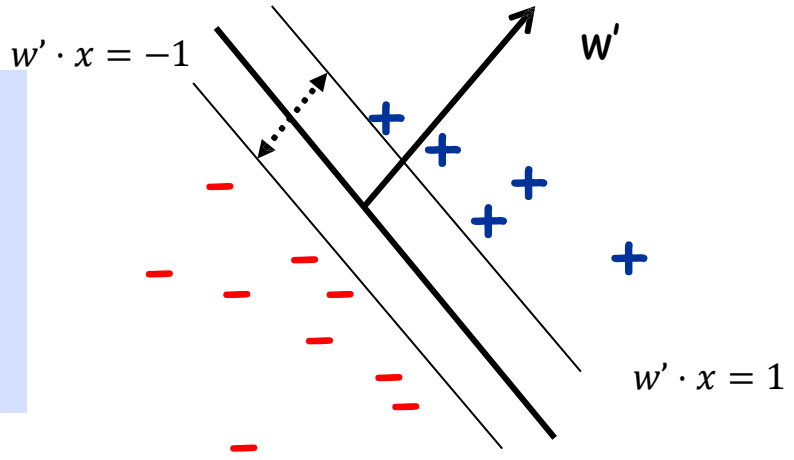


$w' = w/\gamma$, then max γ is equiv. to minimizing $||w'||^2$ (since $||w'||^2 = 1/\gamma^2$).
So, dividing both sides by γ and writing in terms of w' we get:

Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Minimize $||w'||^2$ under the constraint:

- For all i, $y_i w' \cdot x_i \geq 1$



Support Vector Machines (SVMs)

Recall...

Directly optimize for the maximum margin separator: SVMs

Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

$\text{argmin}_w \ ||w||^2$ s.t.:

- For all i, $y_i w \cdot x_i \geq 1$

This is a **constrained optimization** problem.

- The objective is convex (quadratic)
- All constraints are linear
- Can solve efficiently (in poly time) using standard **quadratic programing** (QP) software

Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Recall...

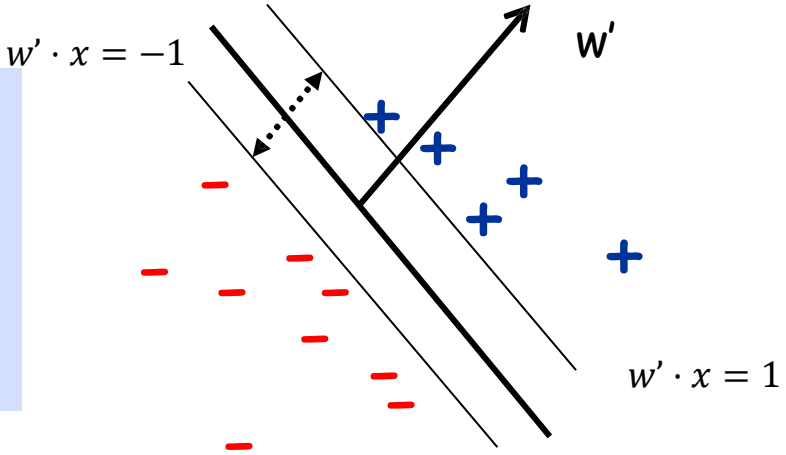
Question: what if data isn't perfectly linearly separable?

Replace "# mistakes" with upper bound called "hinge loss"

Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Minimize $||w'||^2$ under the constraint:

- For all i, $y_i w' \cdot x_i \geq 1$



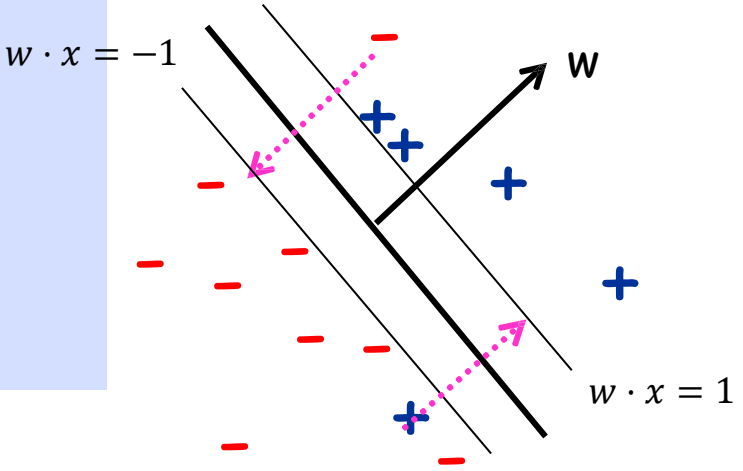
Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Find $\text{argmin}_{w,\xi_1,\ldots,\xi_m} ||w||^2 + C \sum_i \xi_i$ s.t.:

- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$

$\xi_i \geq 0$

ξ_i are "slack variables"



Support Vector Machines (SVMs)

Recall...

Question: what if data isn't perfectly linearly separable?

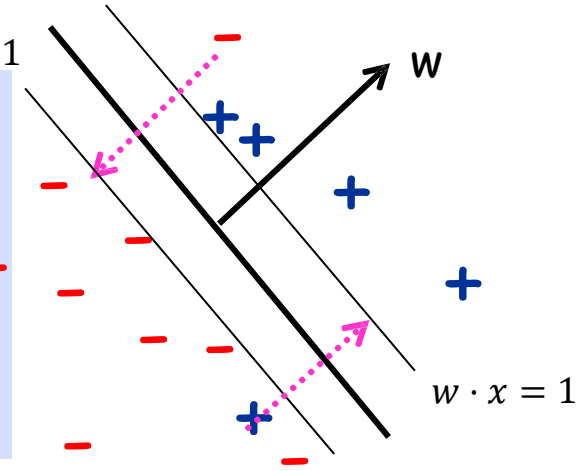
Replace "# mistakes" with upper bound called "hinge loss"

Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Find $\text{argmin}_{w,\xi_1,\ldots,\xi_m} ||w||^2 + C\sum_i \xi_i$ s.t.:

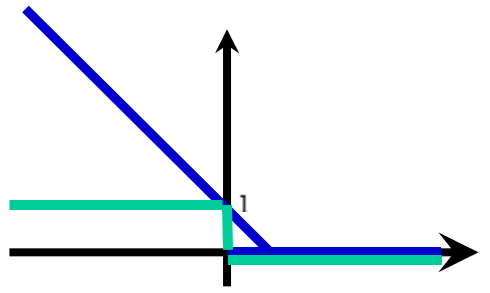
- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$

$\xi_i \geq 0$



ξ_i are "slack variables"

C controls the relative weighting between the twin goals of making the $||w||^2$ small (margin is large) and ensuring that most examples have functional margin ≥ 1.



$l(w, x, y) = \max(0, 1 - y\, w \cdot x)$

t_2 Loss Function

- We start with a linear model for regression as

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

and we have used the squared loss in ordinary linear regression

$$E_2^t(r^t, f(\mathbf{x}^t)) = |r^t - f(\mathbf{x}^t)|^2$$

- Total loss:

$$E_2 = \sum_t E_2^t(r^t, f(\mathbf{x}^t)) = \sum_t |r^t - f(\mathbf{x}^t)|^2$$

- Squared regression (or least squares regression):

$$\underset{\mathbf{w},\, w_0}{\text{minimize}} \quad \frac{1}{N}\sum_{t=1}^{N} |r^t - f(\mathbf{x}^t)|^2$$

Support Vector Regression

ϵ-Insensitive Loss Function – I

- In order for the sparseness property of support vectors in SVM for classification to carry over to regression, we do not use the squared loss but the ϵ-insensitive loss function:

$$E_\epsilon^t(r^t, f(\mathbf{x}^t)) = (|r^t - f(\mathbf{x}^t)| - \epsilon)_+ = \begin{cases} 0 & \text{if } |r^t - f(\mathbf{x}^t)| \leq \epsilon \\ |r^t - f(\mathbf{x}^t)| - \epsilon & \text{otherwise} \end{cases}$$

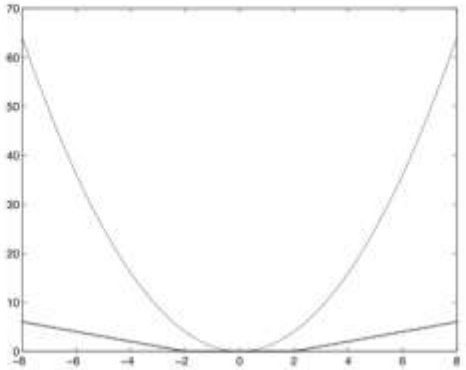
- Two characteristics:
  - Errors are tolerated up to a threshold of ϵ, i.e., no loss for point lying inside an ϵ-tube around the prediction.
  - Errors beyond ϵ have a linear (rather than quadratic) effect so that the model is more more tolerant to noise and robust against noise.
- Total loss:

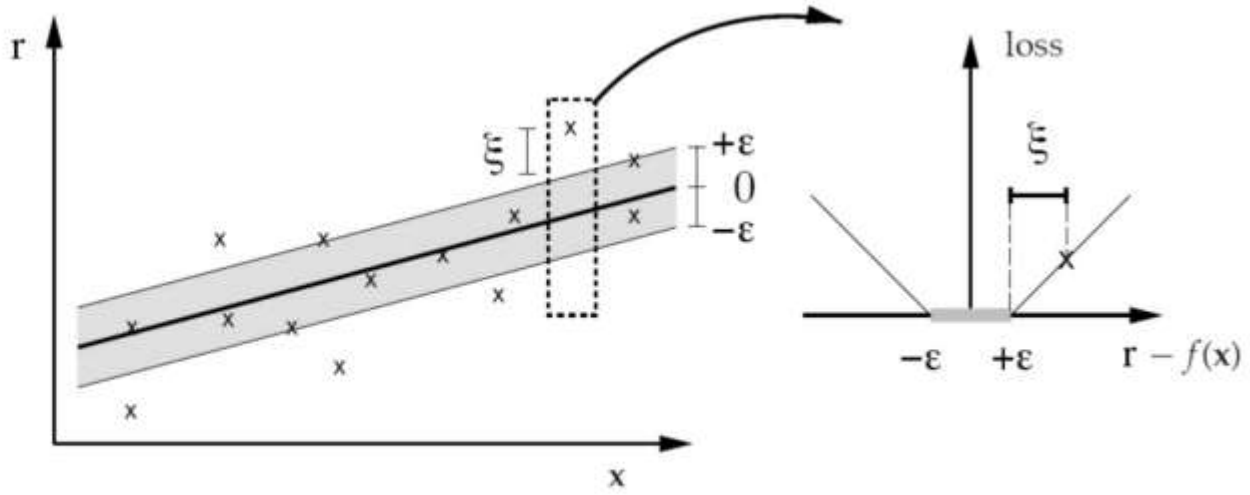
$$E_\epsilon = \sum_t E_\epsilon^t(r^t, f(\mathbf{x}^t)) = \sum_t (|r^t - f(\mathbf{x}^t)| - \epsilon)_+$$

- Tube regression:

$$\underset{\mathbf{w},\, w_0}{\text{minimize}} \quad \frac{1}{N} \sum_{t=1}^{N} (|r^t - f(\mathbf{x}^t)| - \epsilon)_+$$

ϵ-Insensitive Loss Function – II





Support Vector Regression

Support Vector Regression

- Support vector (machine) regression (SVR) is given as

$$\underset{\mathbf{w},\, w_0}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_t (|r^t - f(\mathbf{x}^t)| - \epsilon)_+$$

  where C trades off the model complexity (i.e., the flatness of the model) and data misfit.
- The value of ϵ determines the width of the tube (a smaller value indicates a lower tolerance for error) and also affects the number of support vectors and, consequently, the solution sparsity.
  - If ϵ is decreased, the boundary of the tube is shifted inward. Therefore, more datapoints are around the boundary indicating more support vectors.
  - Similarly, increasing ϵ will result in fewer points around the boundary.
- A convex problem, but not a standard QP.
- We will rewrite it to a form similar to SVM which can be QP-solvable.