



# 10-601 Introduction to Machine Learning

Machine Learning Department  
School of Computer Science  
Carnegie Mellon University

# Linear Regression

Matt Gormley  
Lecture 7  
Feb. 22, 2021

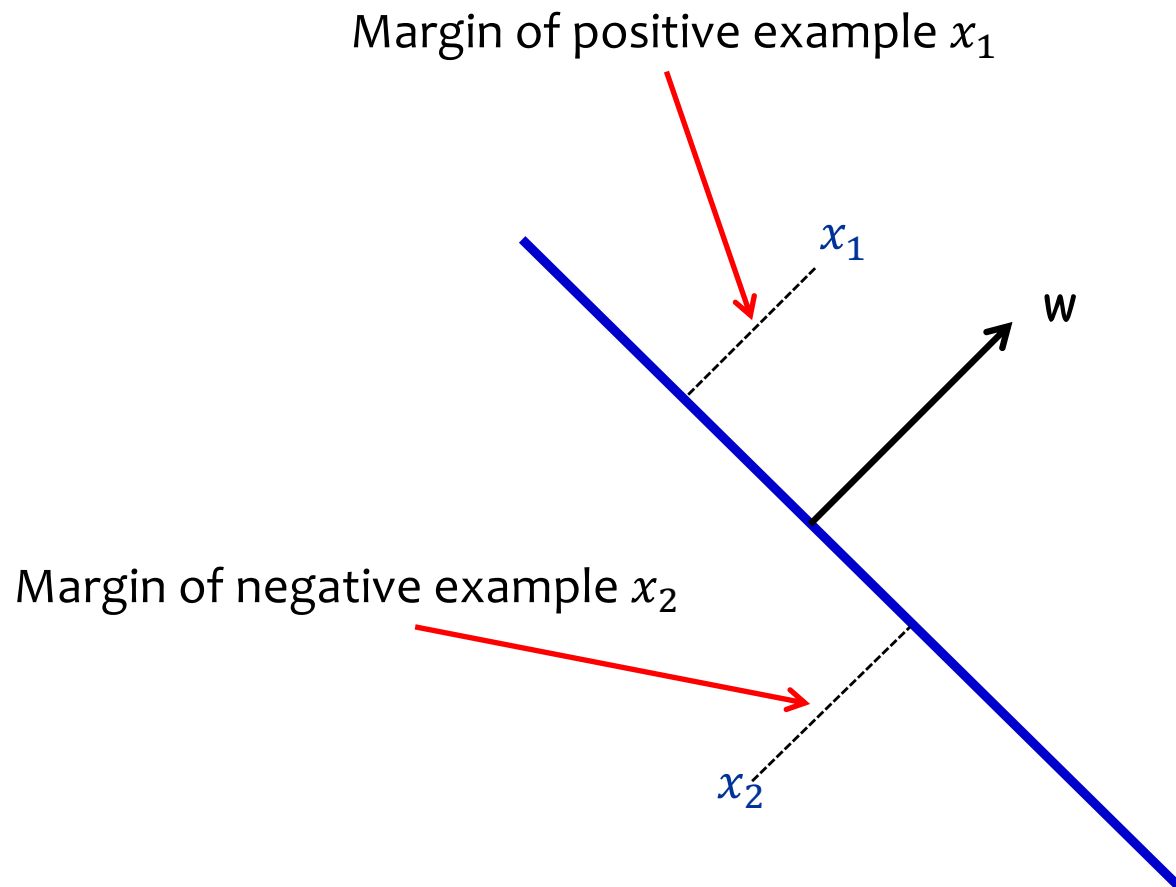
# Reminders

- **Homework 2: Decision Trees**
  - Out: Wed, Feb. 10
  - Due: **Mon**, Feb. 22 at 11:59pm
- **Homework 3: KNN, Perceptron, Lin.Reg.**
  - Out: Mon, Feb. 22
  - Due: Mon, Mar. 01 at 11:59pm

# **ANALYSIS OF PERCEPTRON**

# Geometric Margin

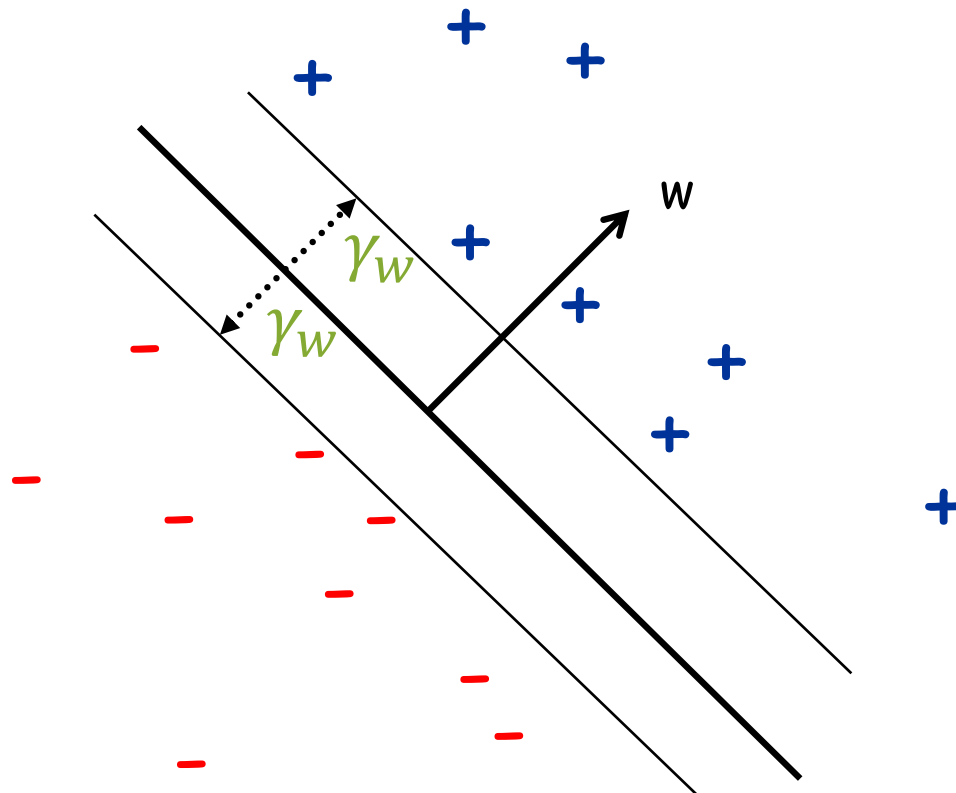
**Definition:** The **margin** of example  $x$  w.r.t. a linear sep.  $w$  is the distance from  $x$  to the plane  $w \cdot x = 0$  (or the negative if on wrong side)



# Geometric Margin

**Definition:** The **margin** of example  $x$  w.r.t. a linear sep.  $w$  is the distance from  $x$  to the plane  $w \cdot x = 0$  (or the negative if on wrong side)

**Definition:** The **margin**  $\gamma_w$  of a set of examples  $S$  wrt a linear separator  $w$  is the smallest margin over points  $x \in S$ .

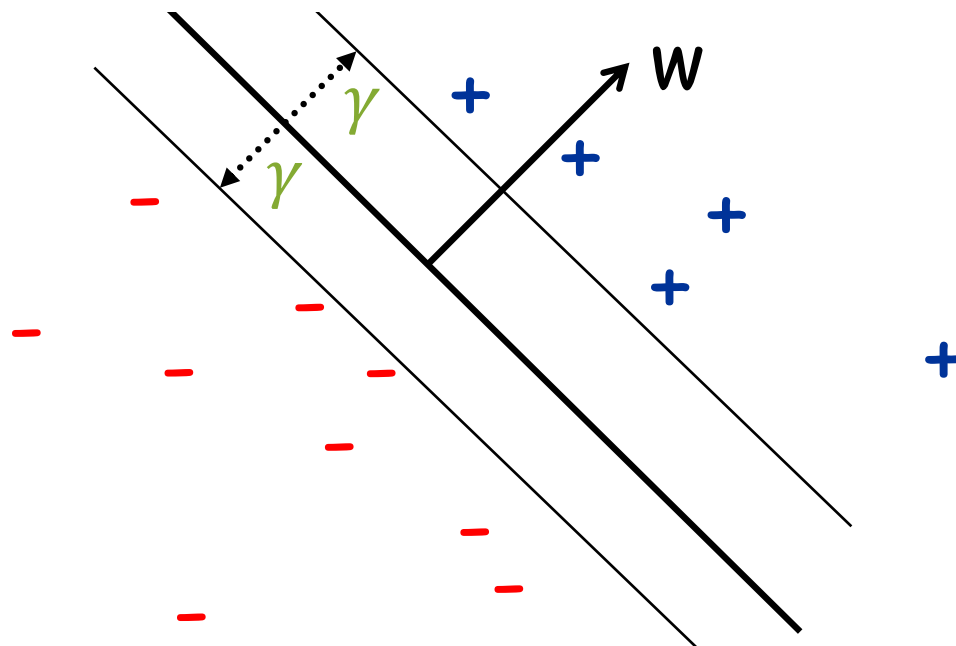


# Geometric Margin

**Definition:** The **margin** of example  $x$  w.r.t. a linear sep.  $w$  is the distance from  $x$  to the plane  $w \cdot x = 0$  (or the negative if on wrong side)

**Definition:** The **margin**  $\gamma_w$  of a set of examples  $S$  wrt a linear separator  $w$  is the smallest margin over points  $x \in S$ .

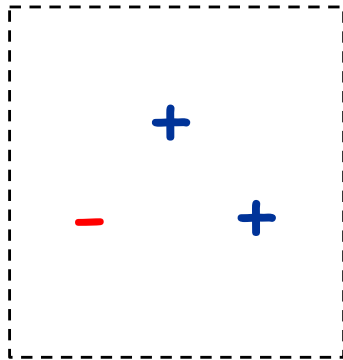
**Definition:** The **margin**  $\gamma$  of a set of examples  $S$  is the **maximum**  $\gamma_w$  over all linear separators  $w$ .



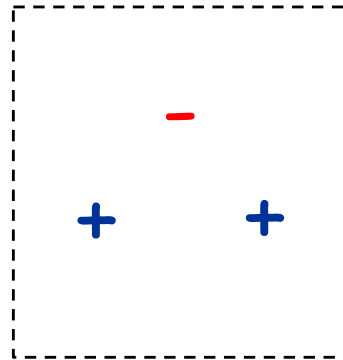
# Linear Separability

**Def:** For a **binary classification** problem, a set of examples  $S$  is **linearly separable** if there exists a linear decision boundary that can separate the points

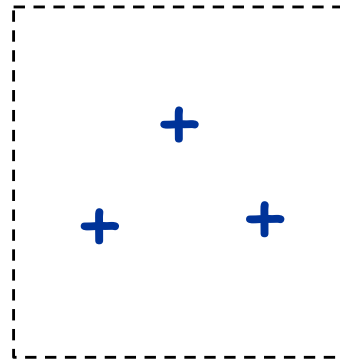
Case 1:



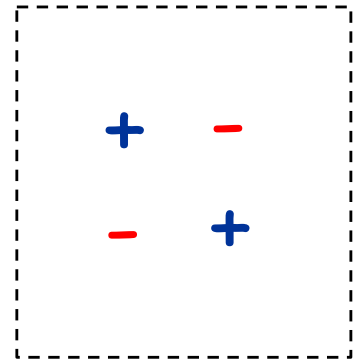
Case 2:



Case 3:



Case 4:

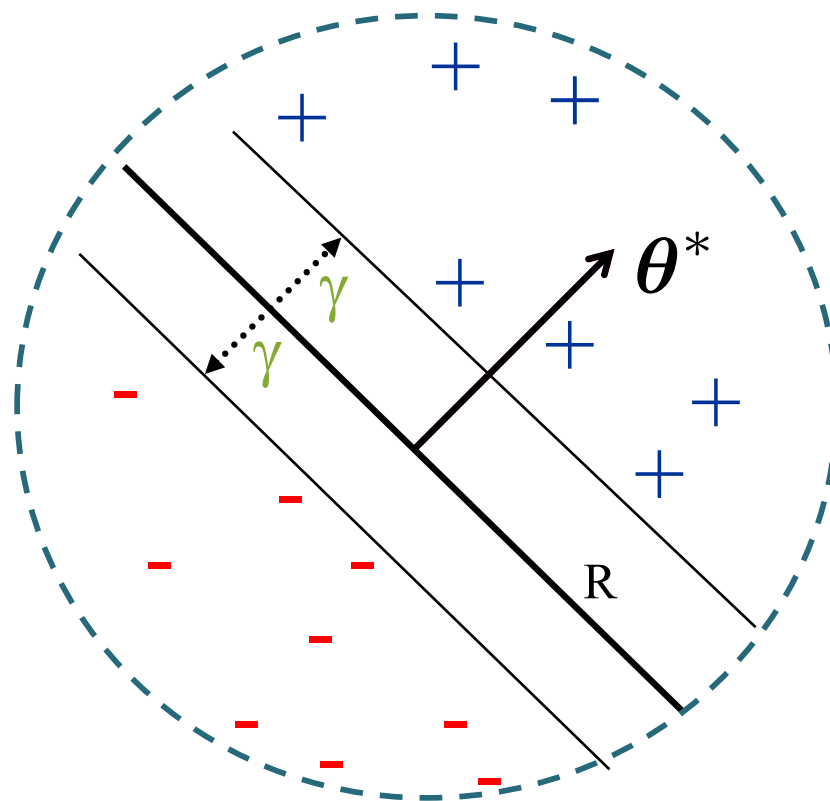


# Analysis: Perceptron

## Perceptron Mistake Bound

**Guarantee:** If data has margin  $\gamma$  and all points inside a ball of radius  $R$ , then Perceptron makes  $\leq (R/\gamma)^2$  mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes; algo is invariant to scaling.)



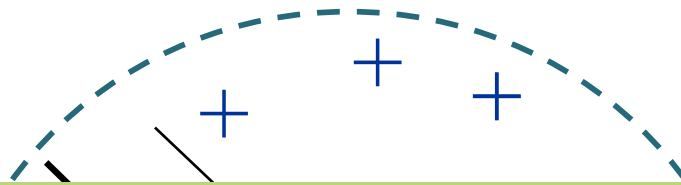


# Analysis: Perceptron

## Perceptron Mistake Bound

**Guarantee:** If data has margin  $\gamma$  and all points inside a ball of radius  $R$ , then Perceptron makes  $\leq (R/\gamma)^2$  mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes; algo is invariant to scaling.)



**Def:** We say that the (batch) perceptron algorithm has **converged** if it stops making mistakes on the training data (perfectly classifies the training data).

**Main Takeaway:** For **linearly separable** data, if the perceptron algorithm cycles repeatedly through the data, it will **converge** in a finite # of steps.

# Analysis: Perceptron

## Perceptron Mistake Bound

**Theorem 0.1** (Block (1962), Novikoff (1962)).

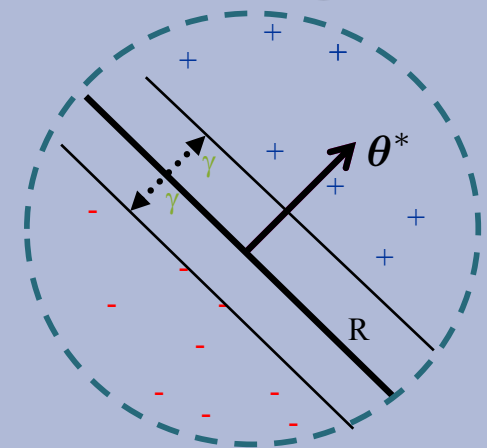
Given dataset:  $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$ .

Suppose:

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

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$



# Analysis: Perceptron

**Common Misunderstanding:**

The radius is centered at the origin, not at the center of the points.

## Perceptron Mistake Bound

**Theorem 0.1** (Block (1962), Novikoff (1962))

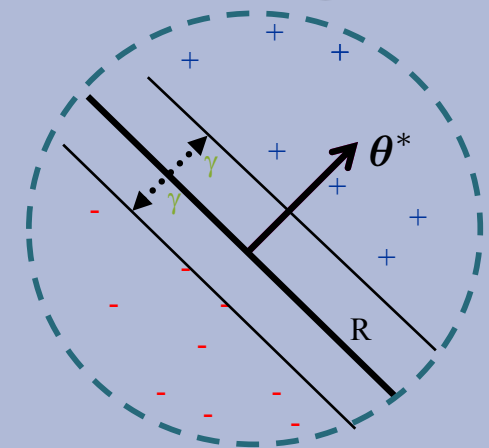
Given dataset:  $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$

Suppose:

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

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$



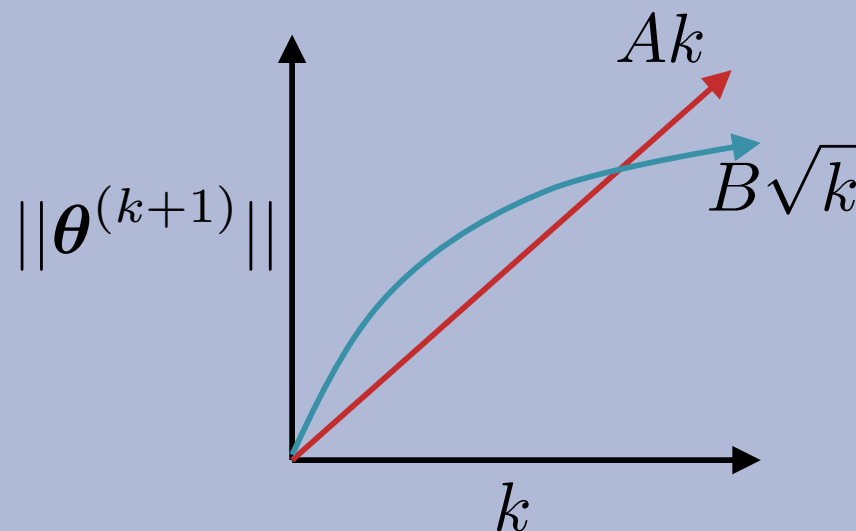
# Analysis: Perceptron

## Proof of Perceptron Mistake Bound:

We will show that there exist constants A and B s.t.

$$Ak \leq \|\boldsymbol{\theta}^{(k+1)}\| \leq B\sqrt{k}$$

parameters  
after k'th  
mistake



# Analysis: Perceptron

**Theorem 0.1** (Block (1962), Novikoff (1962)).

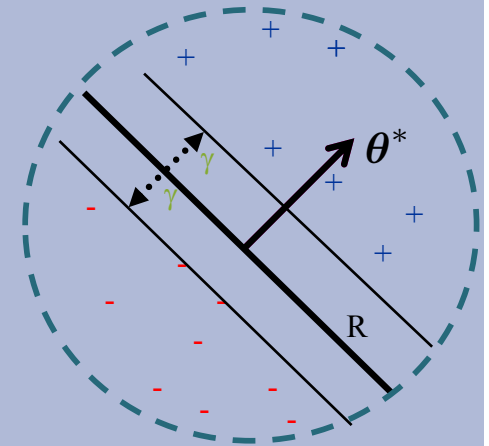
Given dataset:  $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$ .

Suppose:

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

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$



---

## Algorithm 1 Perceptron Learning Algorithm (Online)

---

```
1: procedure PERCEPTRON( $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots\}$ )
2:    $\boldsymbol{\theta} \leftarrow \mathbf{0}, k = 1$  ▷ Initialize parameters
3:   for  $i \in \{1, 2, \dots\}$  do ▷ For each example
4:     if  $y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)}) \leq 0$  then ▷ If mistake
5:        $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + y^{(i)} \mathbf{x}^{(i)}$  ▷ Update parameters
6:        $k \leftarrow k + 1$ 
7:   return  $\boldsymbol{\theta}$ 
```

---

# Analysis: Perceptron

## Proof of Perceptron Mistake Bound:

Part 1: for some  $A$ ,  $Ak \leq \|\theta^{(k+1)}\|$

$$\theta^{(k+1)} \cdot \theta^* = (\theta^{(k)} + y^{(i)} \mathbf{x}^{(i)}) \theta^*$$

by Perceptron algorithm update

$$= \theta^{(k)} \cdot \theta^* + y^{(i)} (\theta^* \cdot \mathbf{x}^{(i)})$$

$$\geq \theta^{(k)} \cdot \theta^* + \gamma$$

by assumption

$$\Rightarrow \theta^{(k+1)} \cdot \theta^* \geq k\gamma$$

by induction on  $k$  since  $\theta^{(1)} = \mathbf{0}$

$$\Rightarrow \|\theta^{(k+1)}\| \geq k\gamma$$

since  $\|\mathbf{w}\| \times \|\mathbf{u}\| \geq \mathbf{w} \cdot \mathbf{u}$  and  $\|\theta^*\| = 1$

Cauchy-Schwartz inequality

# Analysis: Perceptron

## Proof of Perceptron Mistake Bound:

Part 2: for some  $B$ ,  $\|\boldsymbol{\theta}^{(k+1)}\| \leq B\sqrt{k}$

$$\|\boldsymbol{\theta}^{(k+1)}\|^2 = \|\boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)}\|^2$$

by Perceptron algorithm update

$$= \|\boldsymbol{\theta}^{(k)}\|^2 + (y^{(i)})^2\|\mathbf{x}^{(i)}\|^2 + 2y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)})$$

$$\leq \|\boldsymbol{\theta}^{(k)}\|^2 + (y^{(i)})^2\|\mathbf{x}^{(i)}\|^2$$

since  $k$ th mistake  $\Rightarrow y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)}) \leq 0$

$$= \|\boldsymbol{\theta}^{(k)}\|^2 + R^2$$

since  $(y^{(i)})^2\|\mathbf{x}^{(i)}\|^2 = \|\mathbf{x}^{(i)}\|^2 = R^2$  by assumption and  $(y^{(i)})^2 = 1$

$$\Rightarrow \|\boldsymbol{\theta}^{(k+1)}\|^2 \leq kR^2$$

by induction on  $k$  since  $(\boldsymbol{\theta}^{(1)})^2 = 0$

$$\Rightarrow \|\boldsymbol{\theta}^{(k+1)}\| \leq \sqrt{k}R$$

# Analysis: Perceptron

## Proof of Perceptron Mistake Bound:

Part 3: Combining the bounds finishes the proof.

$$k\gamma \leq \|\boldsymbol{\theta}^{(k+1)}\| \leq \sqrt{k}R$$

$$\Rightarrow k \leq (R/\gamma)^2$$



The total number of mistakes  
must be less than this



# Analysis: Perceptron

What if the data is *not* linearly separable?

1. Perceptron will **not converge** in this case (it can't!)
2. However, Freund & Schapire (1999) show that by projecting the points (hypothetically) into a higher dimensional space, we can achieve a similar bound on the number of mistakes made on **one pass** through the sequence of examples

**Theorem 2.** Let  $\langle (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m) \rangle$  be a sequence of labeled examples with  $\|\mathbf{x}_i\| \leq R$ . Let  $\mathbf{u}$  be any vector with  $\|\mathbf{u}\| = 1$  and let  $\gamma > 0$ . Define the deviation of each example as

$$d_i = \max\{0, \gamma - y_i(\mathbf{u} \cdot \mathbf{x}_i)\},$$

and define  $D = \sqrt{\sum_{i=1}^m d_i^2}$ . Then the number of mistakes of the online perceptron algorithm on this sequence is bounded by

$$\left(\frac{R + D}{\gamma}\right)^2.$$

# Perceptron Exercises

## Question:

Unlike Decision Trees and K-Nearest Neighbors, the Perceptron algorithm **does not suffer from overfitting** because it does not have any hyperparameters that could be over-tuned on the training data.

- A. True
- B. False
- C. True and False

# Summary: Perceptron

- Perceptron is a **linear classifier**
- **Simple learning algorithm:** when a mistake is made, add / subtract the features
- Perceptron will converge if the data are **linearly separable**, it will **not** converge if the data are **linearly inseparable**
- For linearly separable and inseparable data, we can **bound the number of mistakes** (geometric argument)
- **Extensions** support nonlinear separators and structured prediction

# Perceptron Learning Objectives

*You should be able to...*

- Explain the difference between online learning and batch learning
- Implement the perceptron algorithm for binary classification [CIML]
- Determine whether the perceptron algorithm will converge based on properties of the dataset, and the limitations of the convergence guarantees
- Describe the inductive bias of perceptron and the limitations of linear models
- Draw the decision boundary of a linear model
- Identify whether a dataset is linearly separable or not
- Defend the use of a bias term in perceptron

# REGRESSION

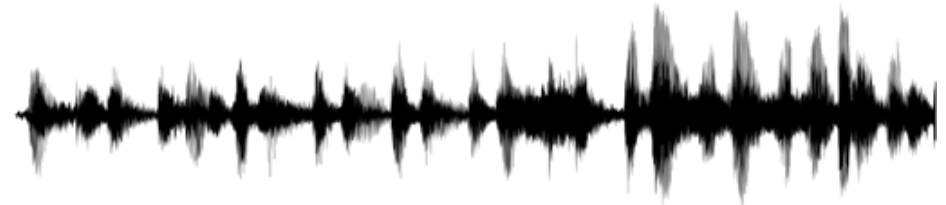
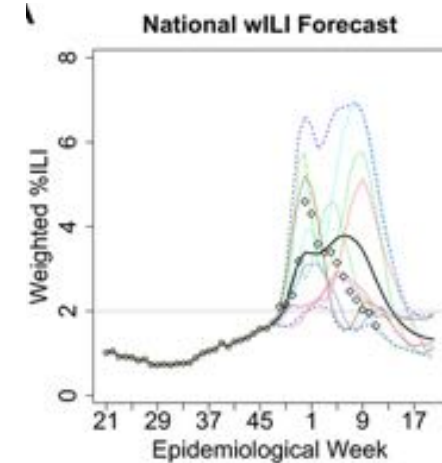
# Regression

## Goal:

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

## Example Applications:

- Stock price prediction
- Forecasting epidemics
- Speech synthesis
- Generation of images (e.g. *Deep Dream*)
- Predicting the number of tourists on Machu Picchu on a given day



# Regression

## Example Application: Forecasting Epidemics

- Input features,  $x$ : attributes of the epidemic
- Output,  $y$ : Weighted %ILI, prevalence of the disease
- Setting: observe past prevalence to predict future prevalence

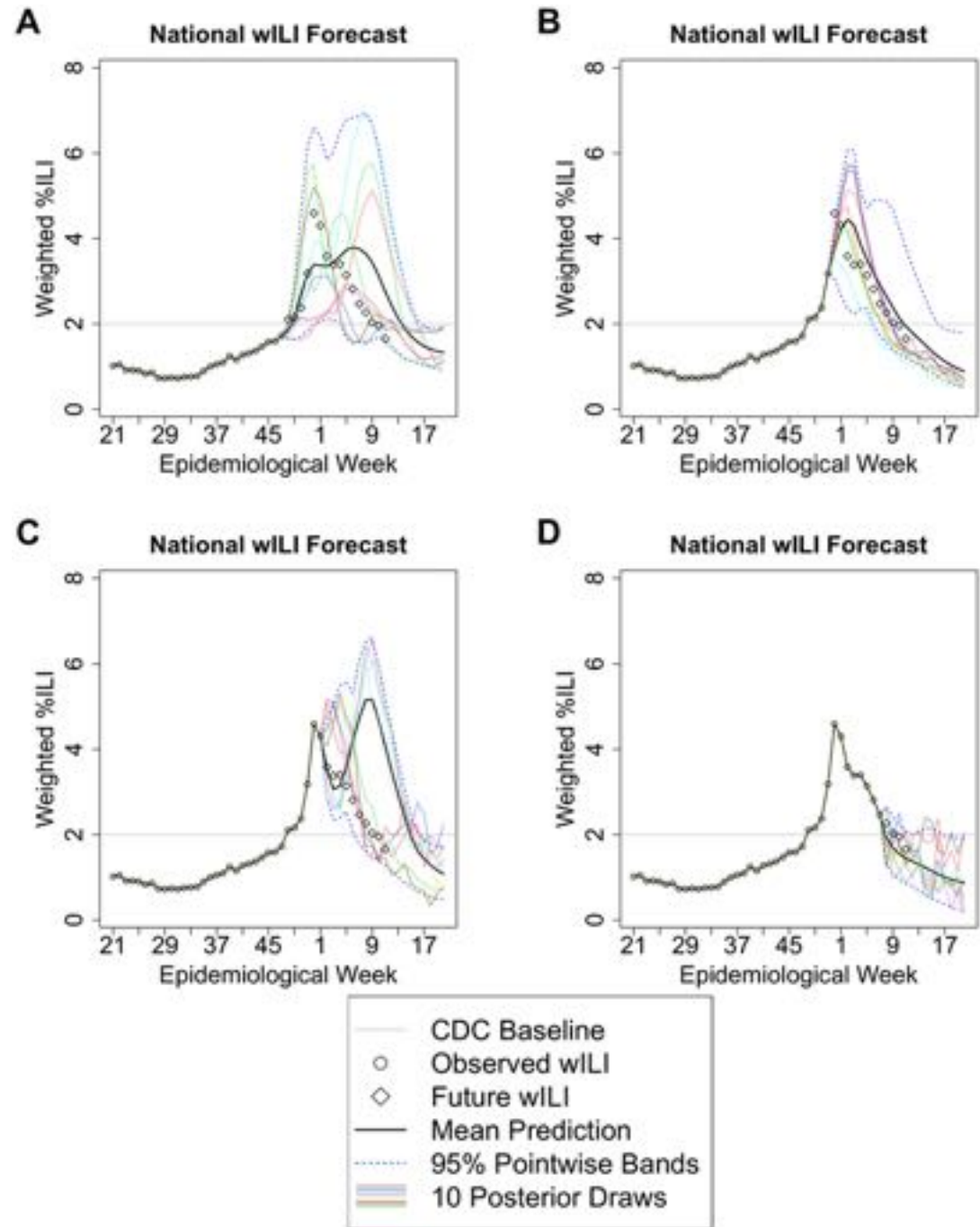
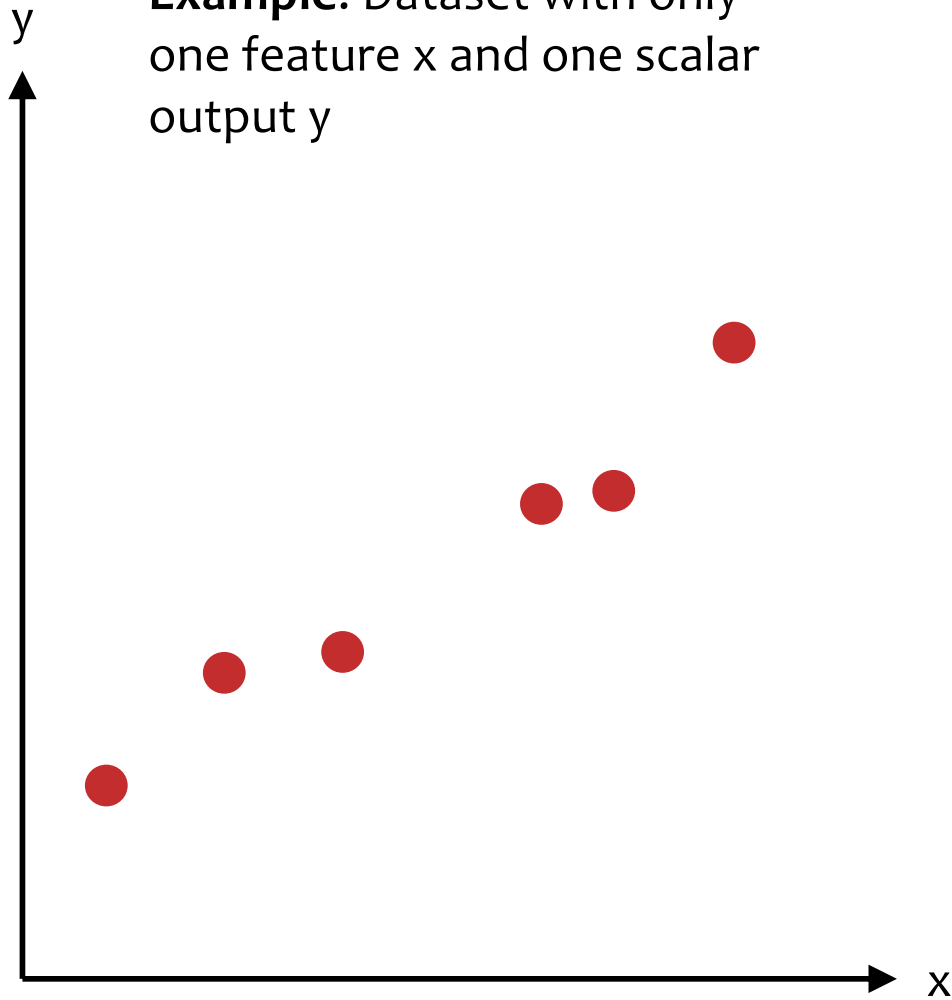


Fig 2. 2013–2014 national forecast, retrospectively, using the final revisions of wILI values, using revised wILI data through epidemiological weeks (A) 47, (B) 51, (C) 1, and (D) 7.

# Regression

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

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

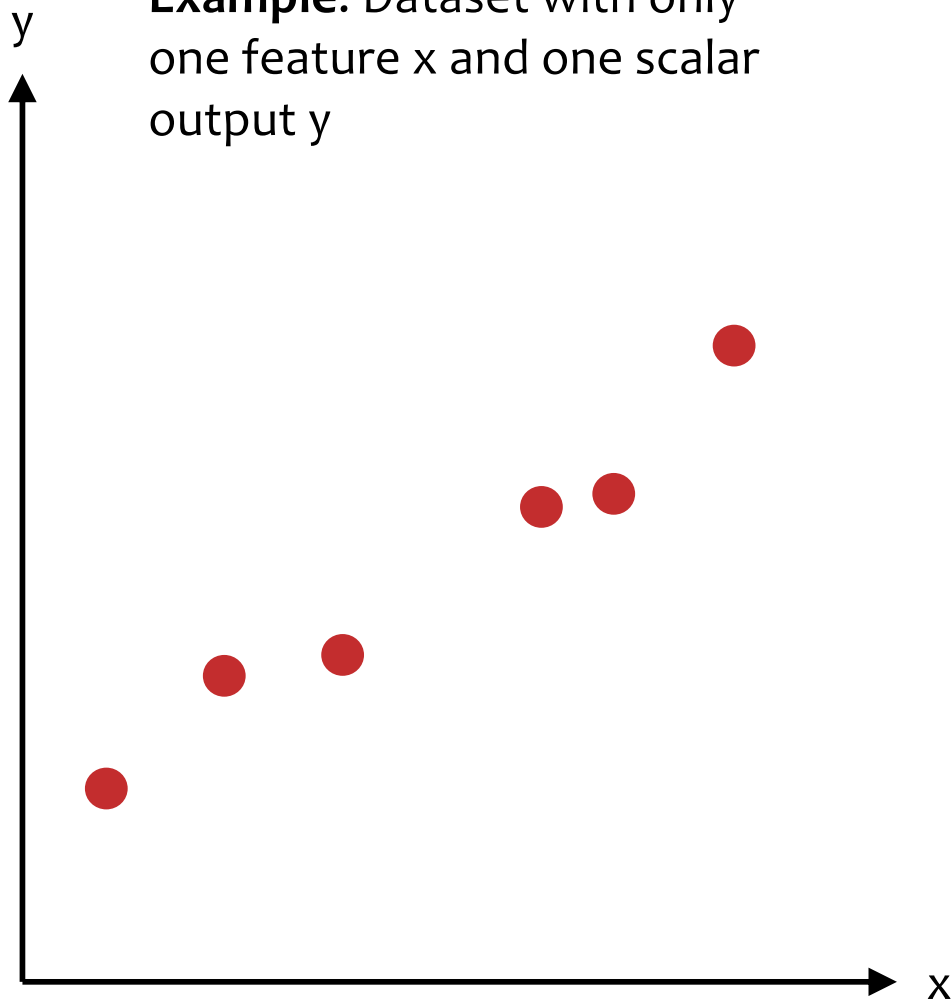




# **K-NEAREST NEIGHBOR REGRESSION**

# k-NN Regression

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



## k=1 Nearest Neighbor Regression

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

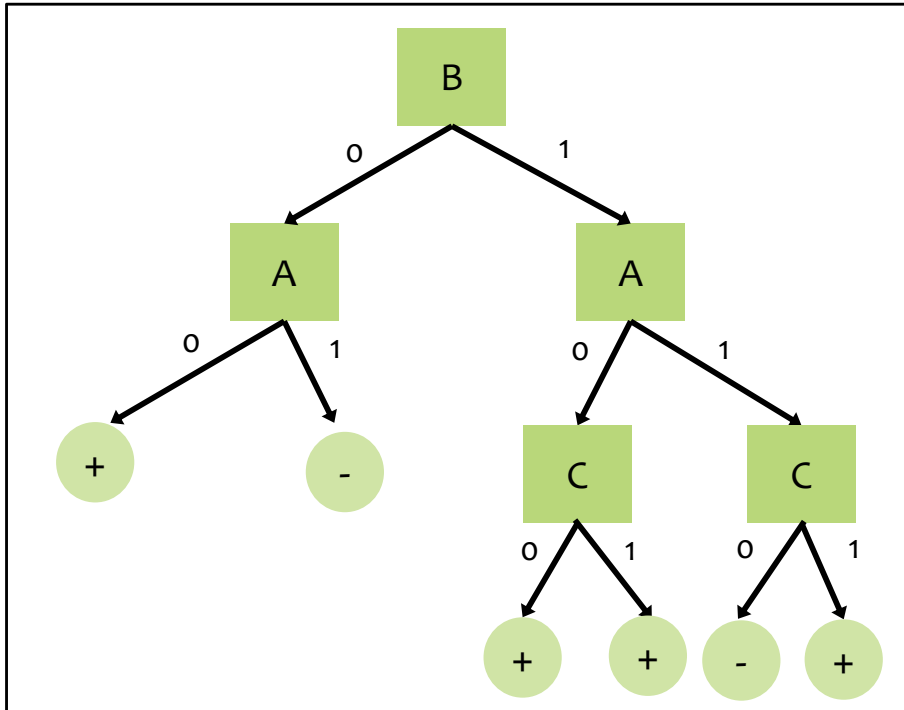
## k=2 Nearest Neighbor Distance Weighted Regression

- *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

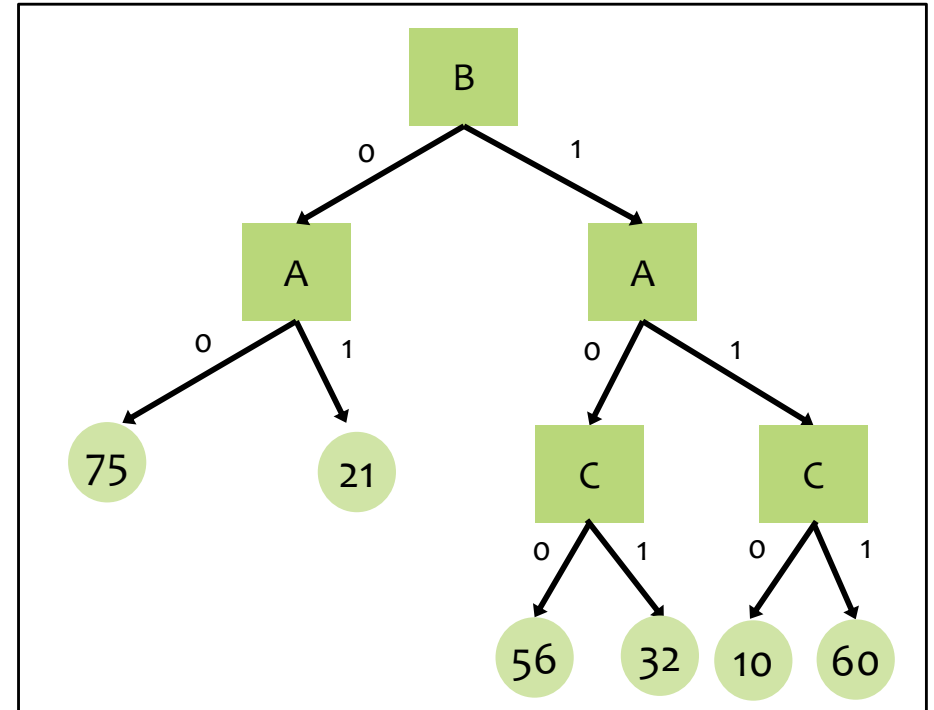
# DECISION TREE REGRESSION

# Decision Tree Regression

Decision Tree for Classification



Decision Tree for Regression

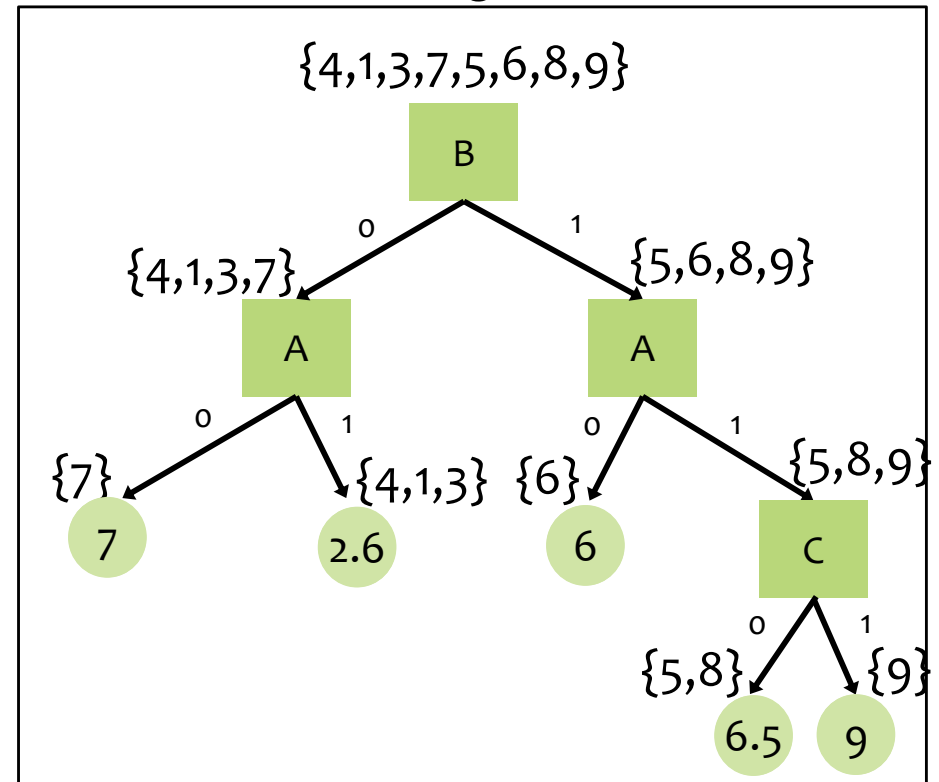


# Decision Tree Regression

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)

# **LINEAR FUNCTIONS, RESIDUALS, AND MEAN SQUARED ERROR**

# Regression Problems

## *Chalkboard*

- Definition of Regression
- Linear functions
- Linear function vs. linear decision boundary
- Residuals
- Mean squared error

The Big Picture

# **OPTIMIZATION FOR ML**



# Optimization for ML

Not quite the same setting as other fields...

- Function we are optimizing might not be the true goal

(e.g. likelihood vs generalization error)

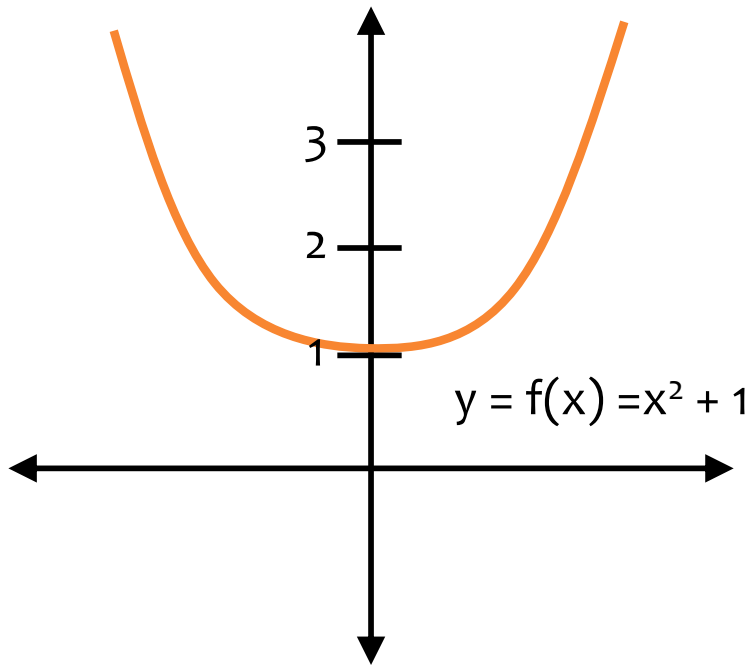
- Precision might not matter

(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



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

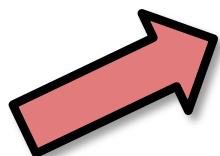
$$x^* = \operatorname{argmin}_x f(x)$$

- Q: What is  $v^*$ ?
- Q: What is  $x^*$ ?

# Notation Trick: Folding in the Intercept Term

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

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



*Notation Trick:* fold the bias  $b$  and the weights  $\mathbf{w}$  into a single vector  $\boldsymbol{\theta}$  by prepending a constant to  $\mathbf{x}$  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:

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

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)

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

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  $\hat{y}$

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





Lantern Riddles:

1. Riddle: 物生来身穿三百多件衣,  
每天脱一件, 年底剩张皮。

Hint: (猜一日常用品)

Answer: 日曆

2. Riddle: 无底洞

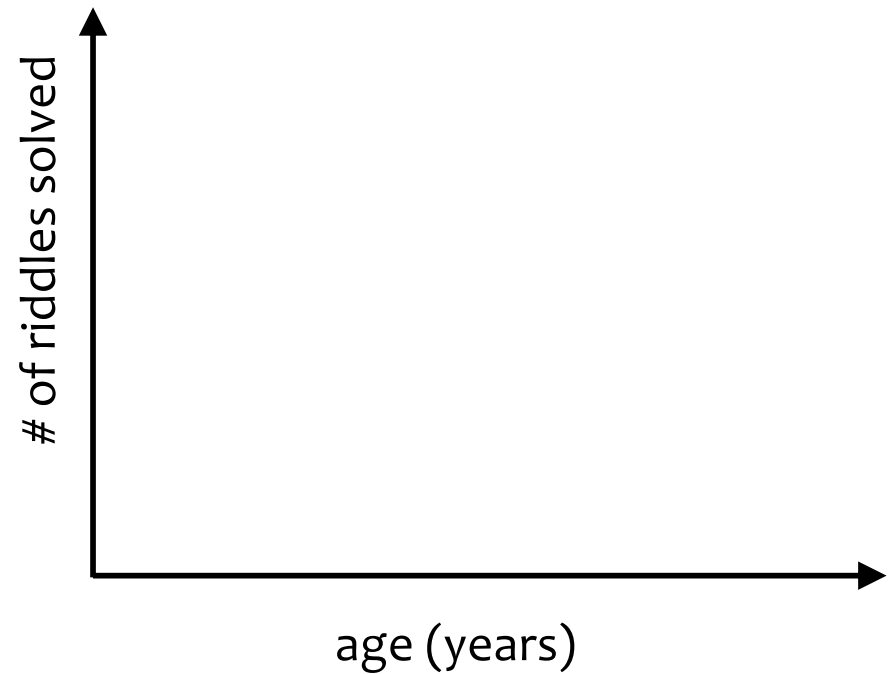
Hint: (猜成语)

Answer: 深不可测

3. Riddles: ‘不负中秋夜明月’

Hint: 成语

Answer: 胜利在望



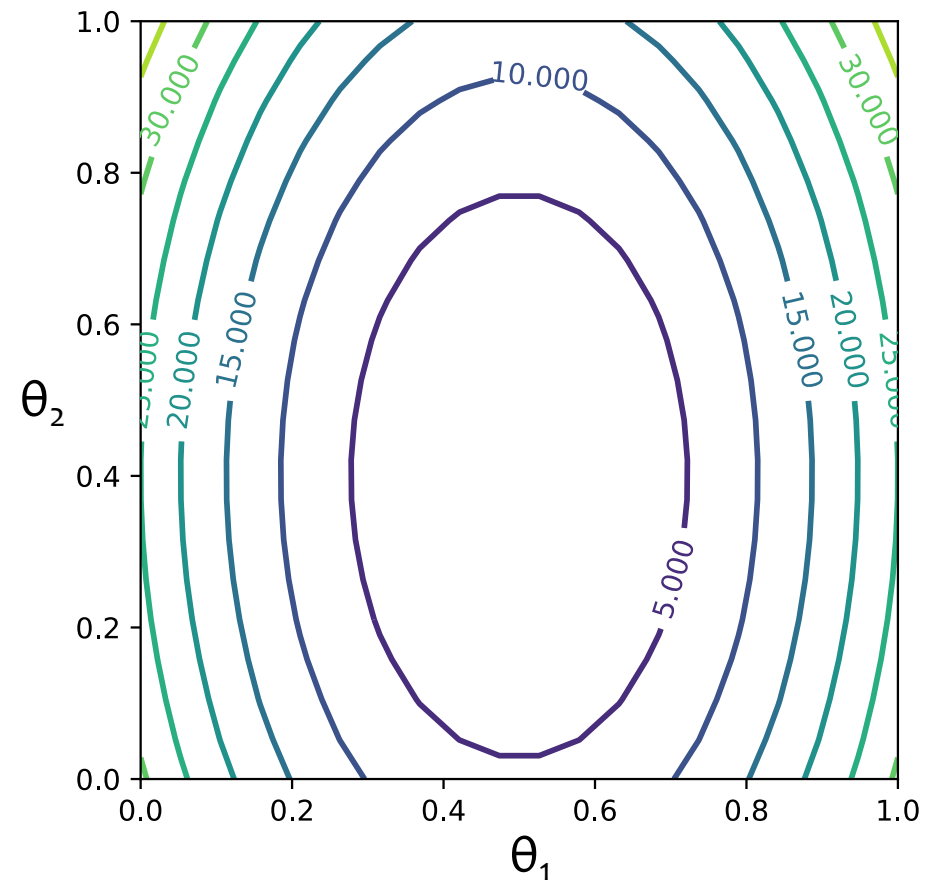
# Contour Plots

## 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



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

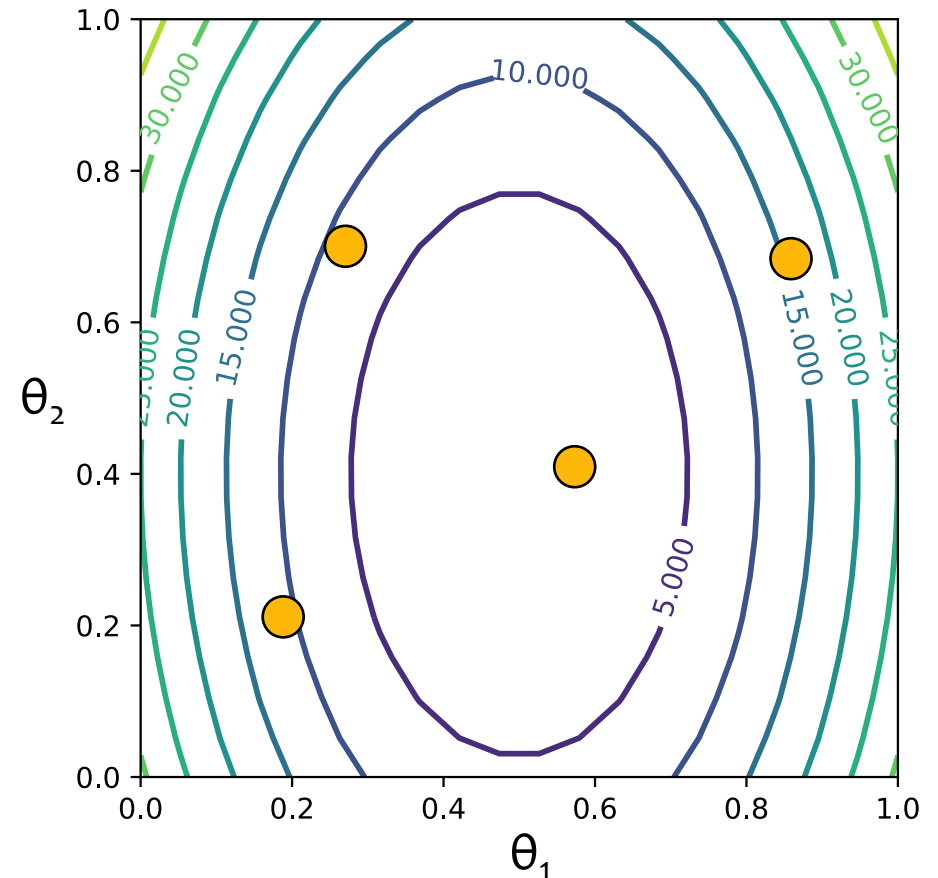


# Optimization by Random Guessing

## Optimization Method #0: Random Guessing

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

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



t	$\theta_1$	$\theta_2$	$J(\theta_1, \theta_2)$
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	19.2



# Optimization by Random Guessing

## Optimization Method #0: Random Guessing

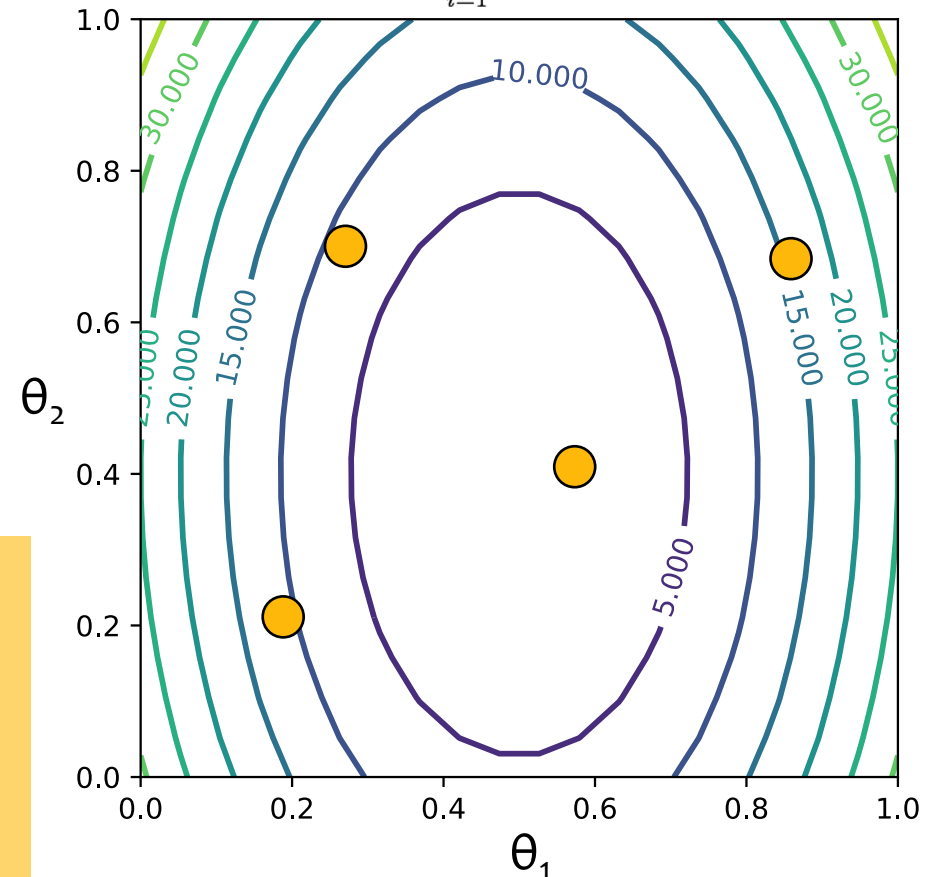
1. Pick a random  $\theta$
2. Evaluate  $J(\theta)$
3. Repeat steps 1 and 2 many times
4. Return  $\theta$  that gives smallest  $J(\theta)$

## For Linear Regression:

- **objective function** is Mean Squared Error (MSE)
- $MSE = J(w, b)$   

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

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

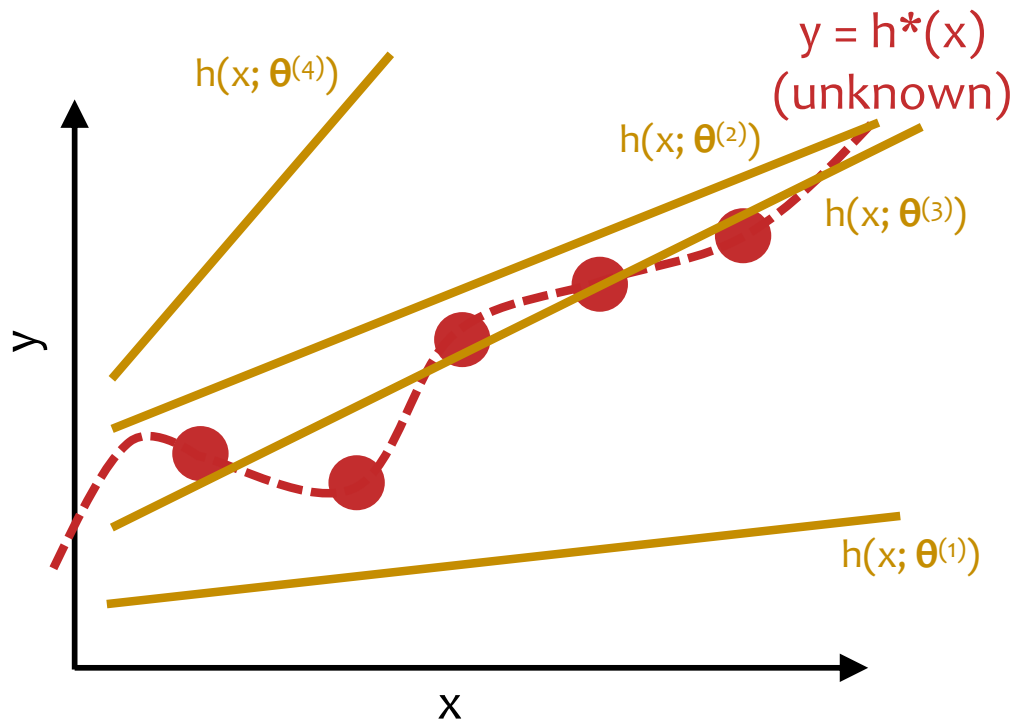


t	$\theta_1$	$\theta_2$	$J(\theta_1, \theta_2)$
1	0.2	0.2	10.4
2	0.3	0.7	7.2
3	0.6	0.4	1.0
4	0.9	0.7	19.2

# Linear Regression by Rand. Guessing

## Optimization Method #0: Random Guessing

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



## For Linear Regression:

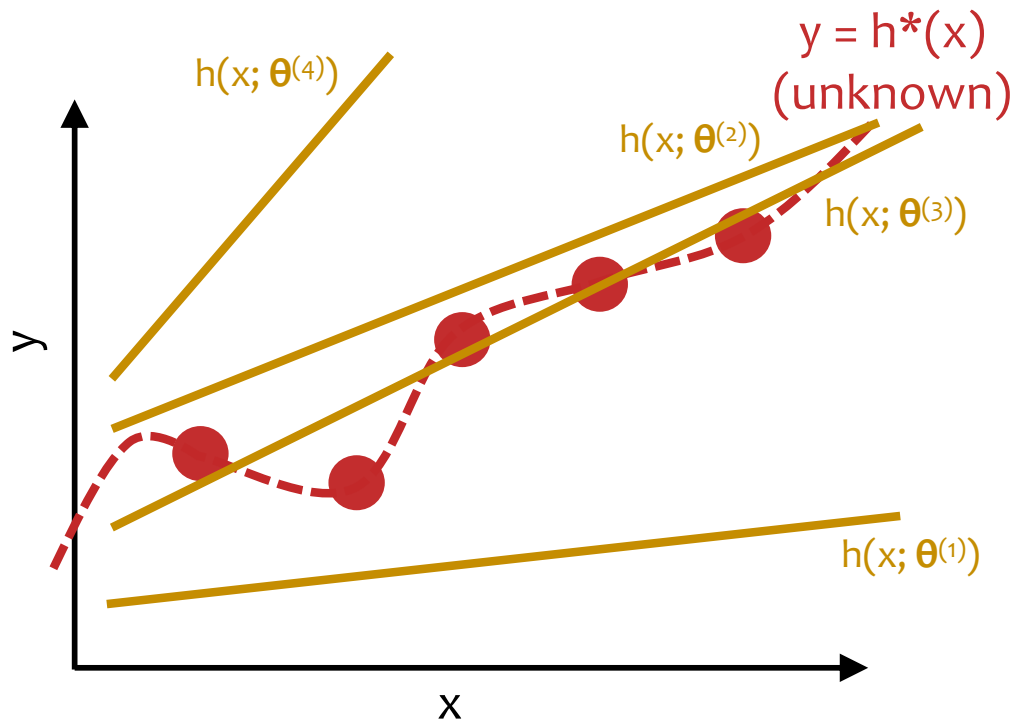
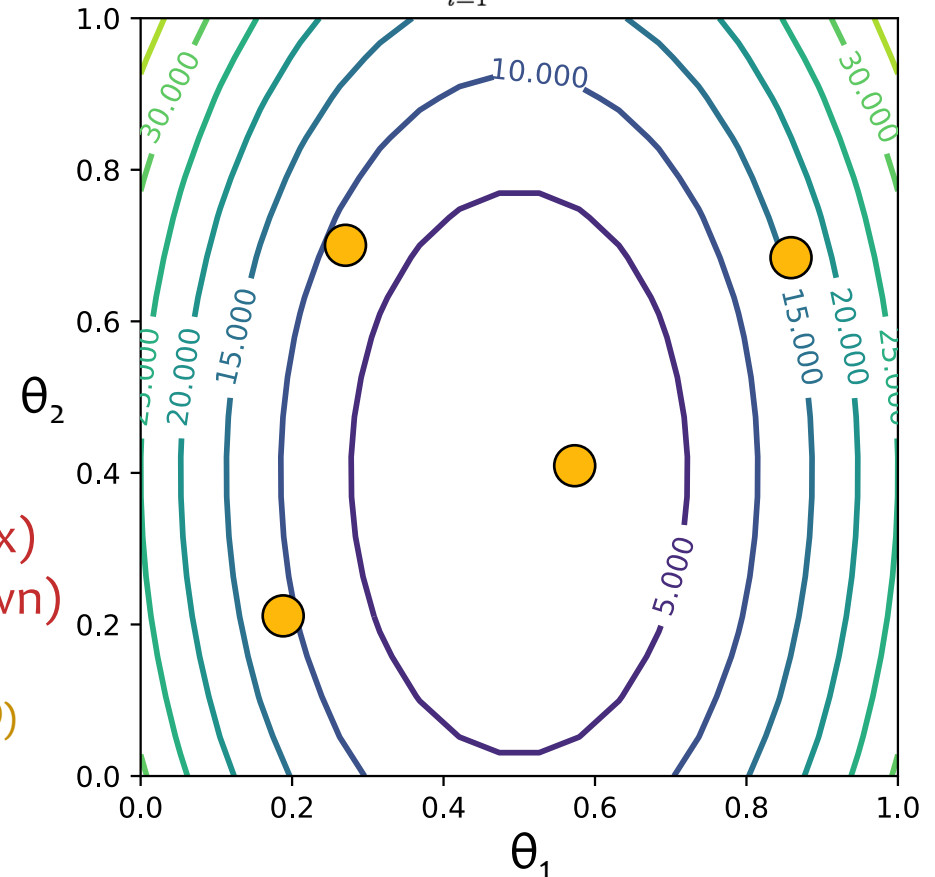
- target function  $h^*(x)$  is **unknown**
- only have access to  $h^*(x)$  through **training examples**  $(x^{(i)}, y^{(i)})$
- want  $h(x; \theta^{(t)})$  that **best approximates**  $h^*(x)$
- **enable generalization** w/inductive bias that restricts hypothesis class to **linear functions**

# Linear Regression by Rand. Guessing

## Optimization Method #0: Random Guessing

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

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



t	$\theta_1$	$\theta_2$	$J(\theta_1, \theta_2)$
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	19.2

# **OPTIMIZATION METHOD #1: GRADIENT DESCENT**

# Optimization for ML

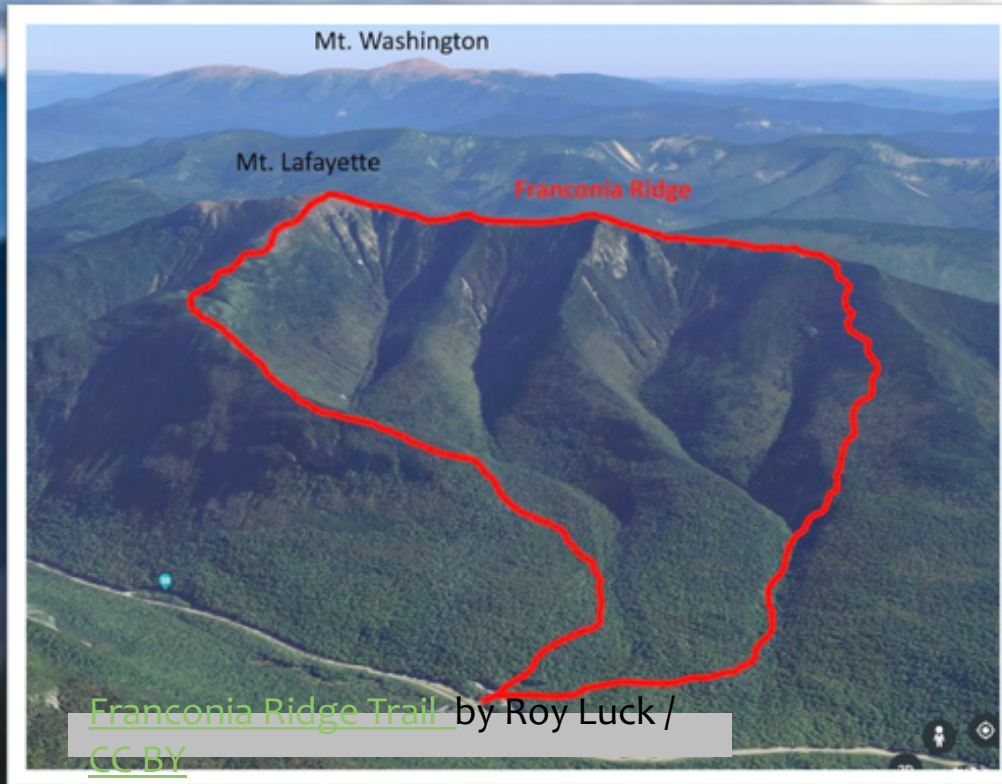
## *Chalkboard*

- Unconstrained optimization
- Derivatives
- Gradient

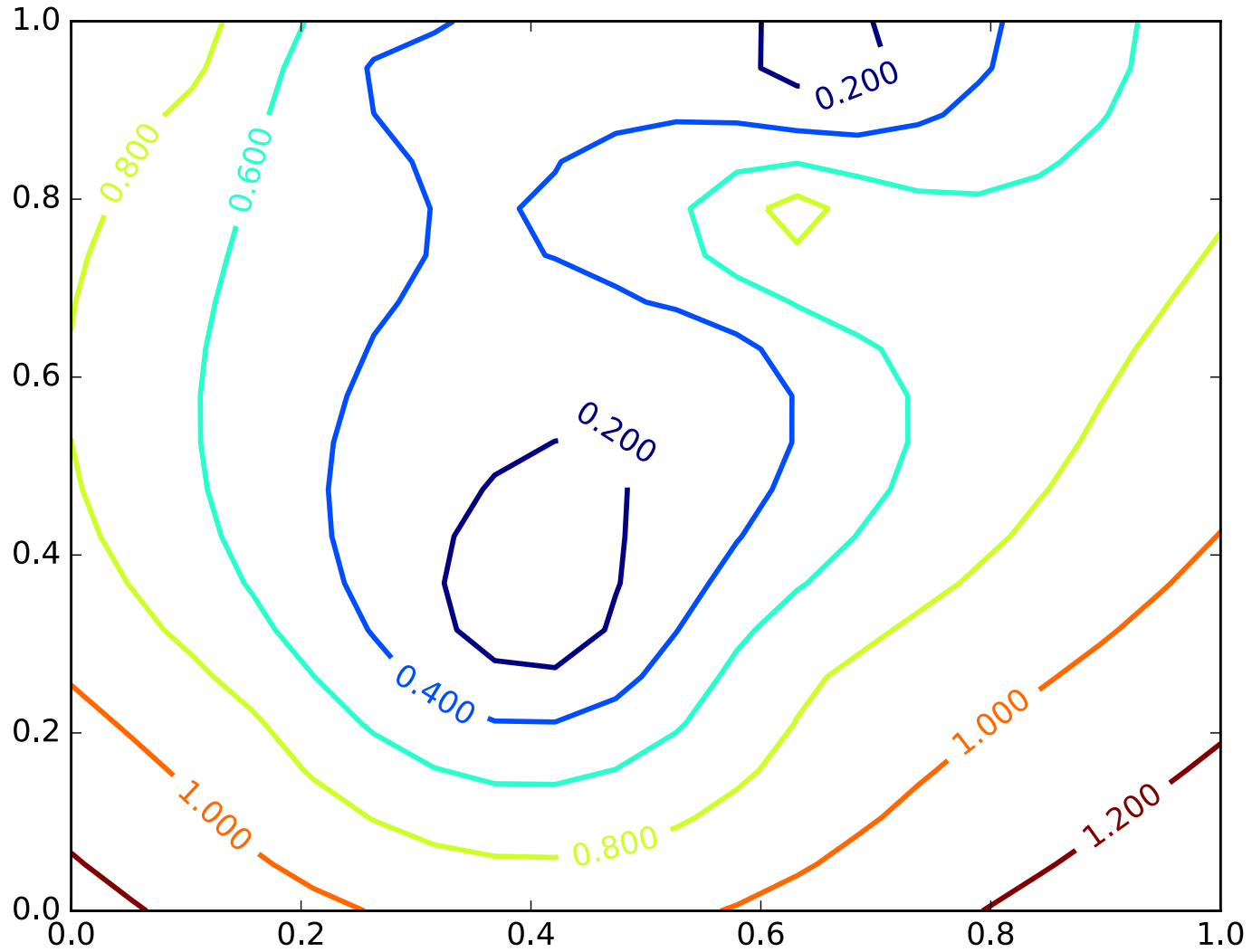
# Topographical Maps



# Topographical Maps

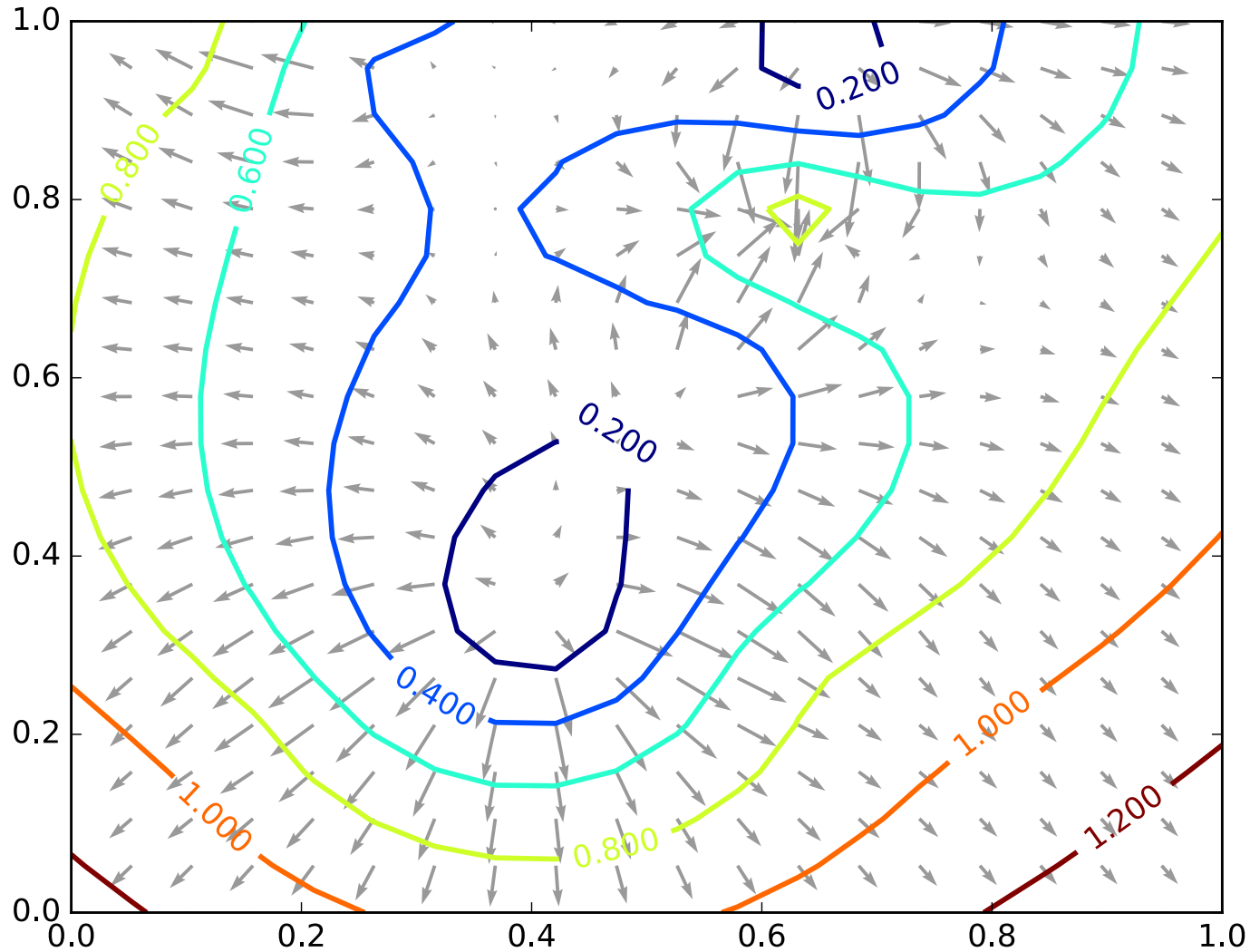


# Gradients



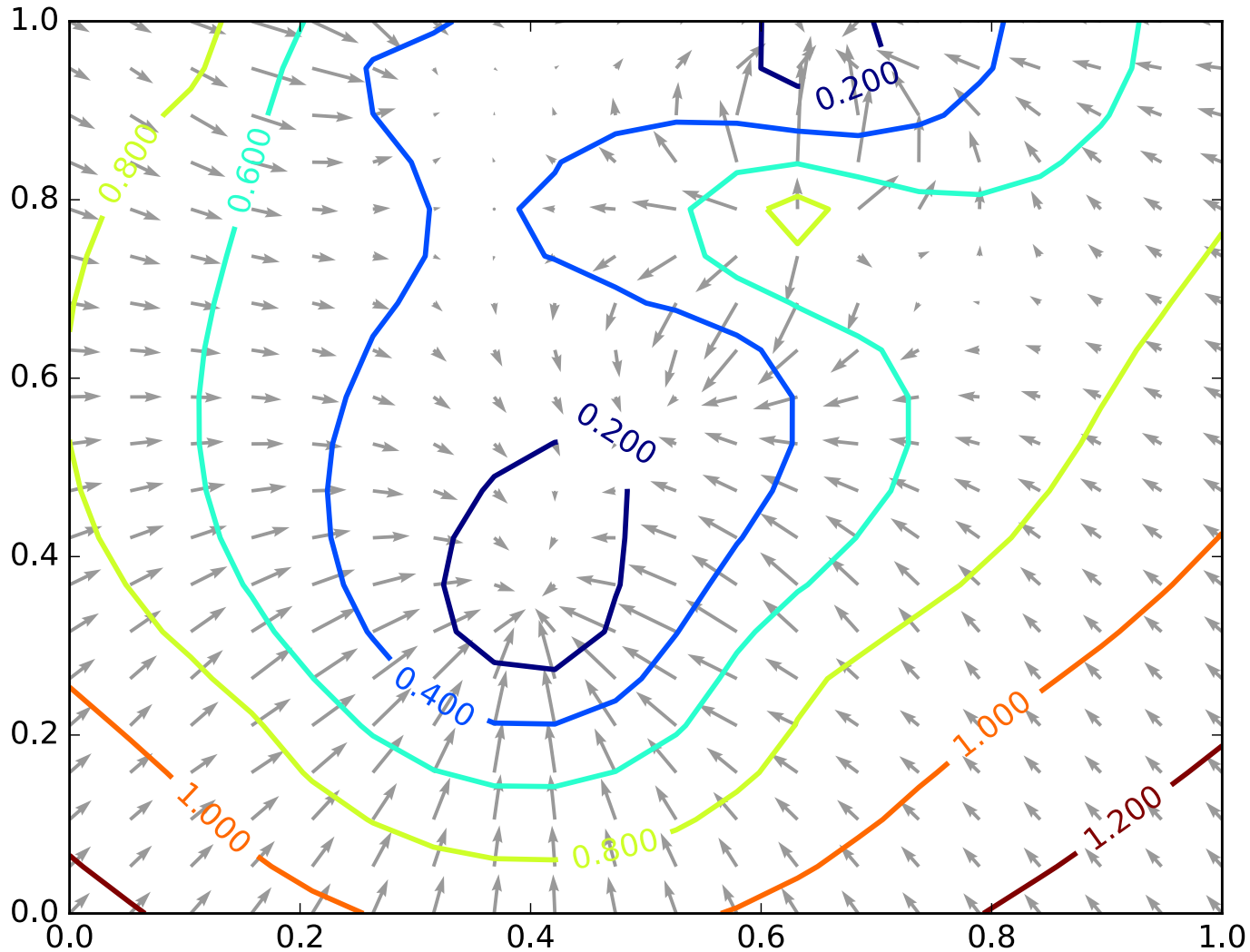


# Gradients



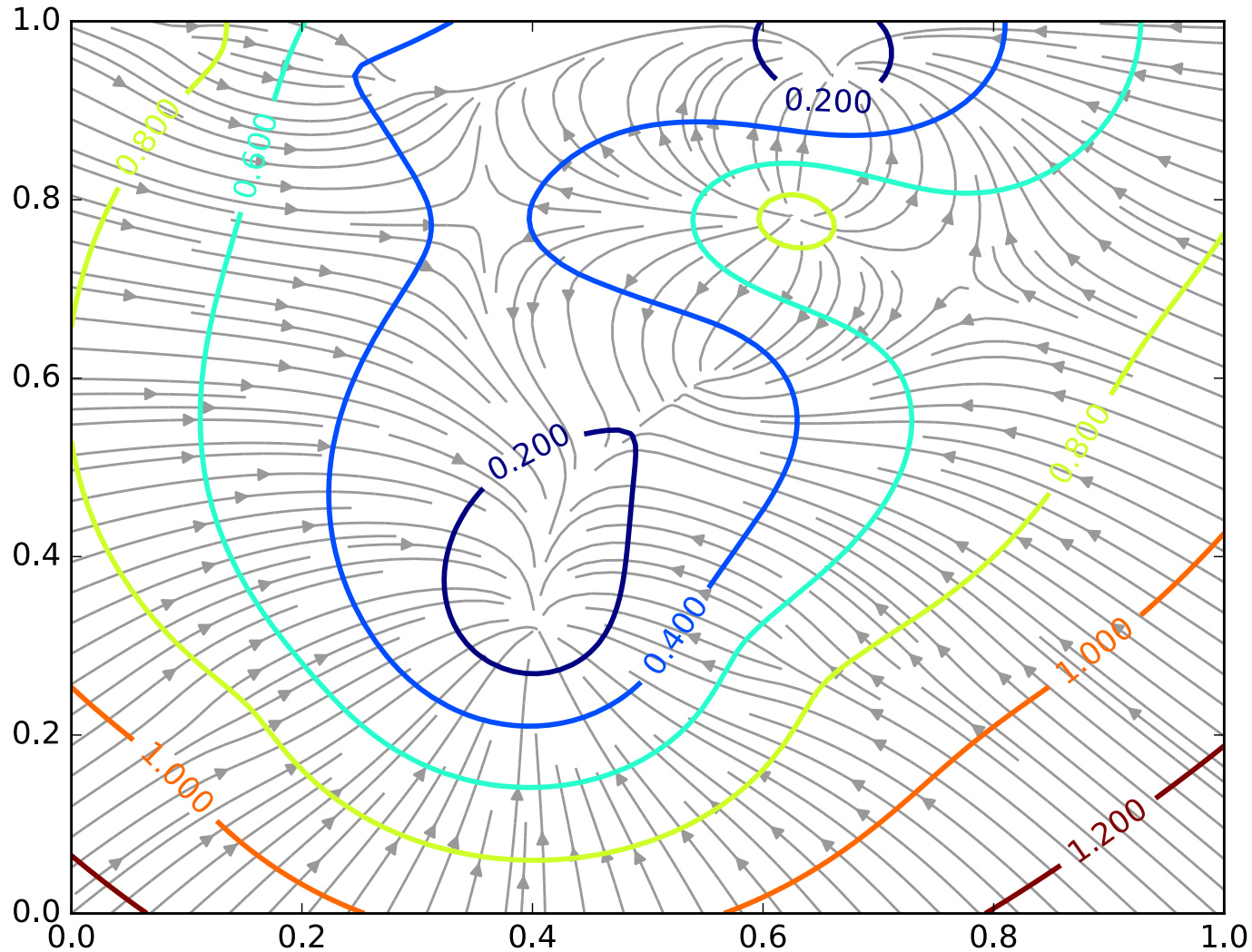
These are the **gradients** that Gradient **Ascent** would follow.

# (Negative) Gradients



These are the **negative** gradients that Gradient **D**escent would follow.

# (Negative) Gradient *Paths*



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