

INFO 251: Applied Machine Learning

# Gradient Descent

Please grab a pen and paper for today's lecture (not a quiz, just to jot down notes!)

# Announcements

- PS3 due in one week
- Next week's classes will be on Zoom only
- Enrollment update

# Course Outline

- Causal Inference and Research Design
  - Experimental methods
  - Non-experiment methods
- Machine Learning
  - Design of Machine Learning Experiments
  - **Linear Models and Gradient Descent**
  - Non-linear models
  - Neural models
  - Unsupervised Learning
  - Practicalities, Fairness, Bias
- Special topics

# Key Concepts (previous lecture)

- Decision boundaries
- Voronoi diagrams
- ( $K$ -)Nearest Neighbors
- Similarity and Distance metrics
- Normalization and Standardization
- Feature weighting

# Key Concepts (today's lecture)

- Cost Functions
- Gradient Descent
- Local and global minima
- Convex functions
- Incremental vs. Batch GD
- Learning rates
- Feature scaling

# Outline

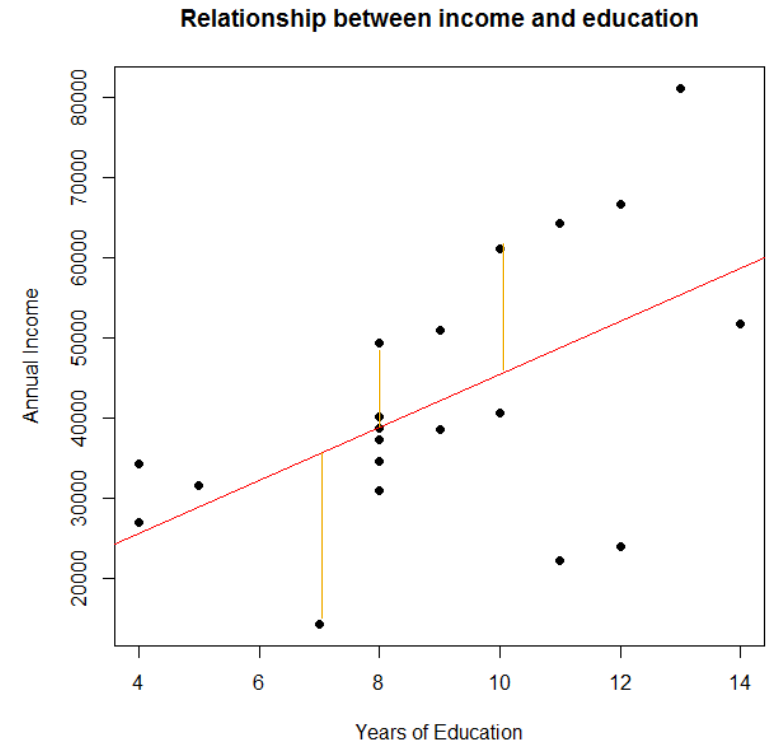
- **Cost functions**
- Gradient descent
- Feature scaling

# Cost minimization

- In general:
  - We make a prediction of  $Y$  using some function  $f(X)$
  - To choose the best model:
    - Define a loss function  $J(Y, f(X))$
    - Minimize the expected loss of  $J$
- Linear regression:
  - $f$  is a linear function (e.g.,  $\alpha + \beta X$ )
  - Minimize squared-error loss  $E(Y - f(X))^2$

# Linear Regression

- OLS as Maximum Likelihood Estimation:
  - $Y_i = \alpha + \beta X_i + \epsilon_i$
  - Idea: Choose  $\alpha$  and  $\beta$  so that  $\alpha + \beta X_i$  is “as close as possible” to  $Y_i$  for training data
- In other words
  - $\min_{\alpha, \beta} \sum_{i=1}^N (\alpha + \beta X_i - Y_i)^2$
- In general, we are minimizing a **Cost Function  $J$** 
  - $\min_{\alpha, \beta} J(\alpha, \beta)$
  - In the case of OLS, we use a “squared error” cost function
  - $J(\alpha, \beta) = \frac{1}{2N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i)^2$





# General formulation (OLS)

- Model ("hypothesis")

- $Y_i = \alpha + \beta X_i$

- Parameters

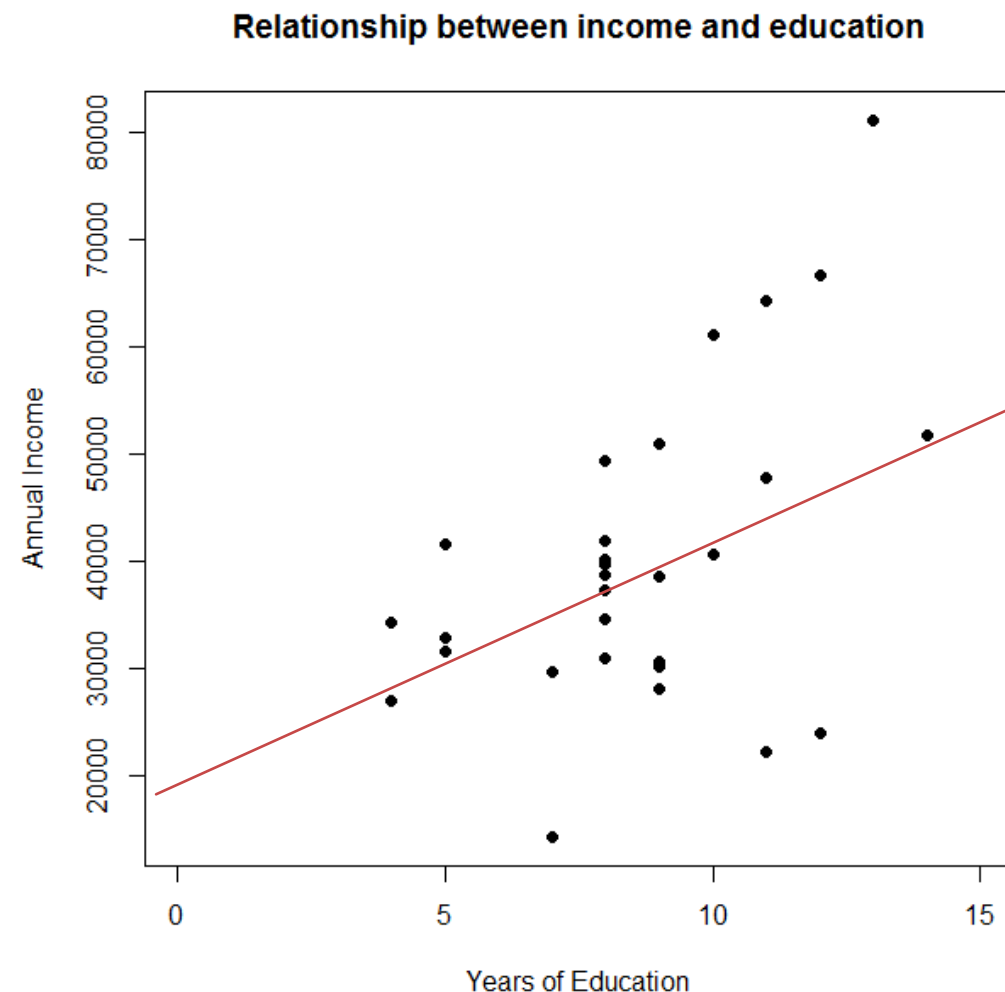
- $\alpha, \beta$

- Cost Function

- $J(\alpha, \beta) = \frac{1}{2N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i)^2$

- Objective

- $\min_{\alpha, \beta} J(\alpha, \beta)$



# OLS with no intercept

- Model ("hypothesis")

- $Y_i = \beta X_i$

- Parameters

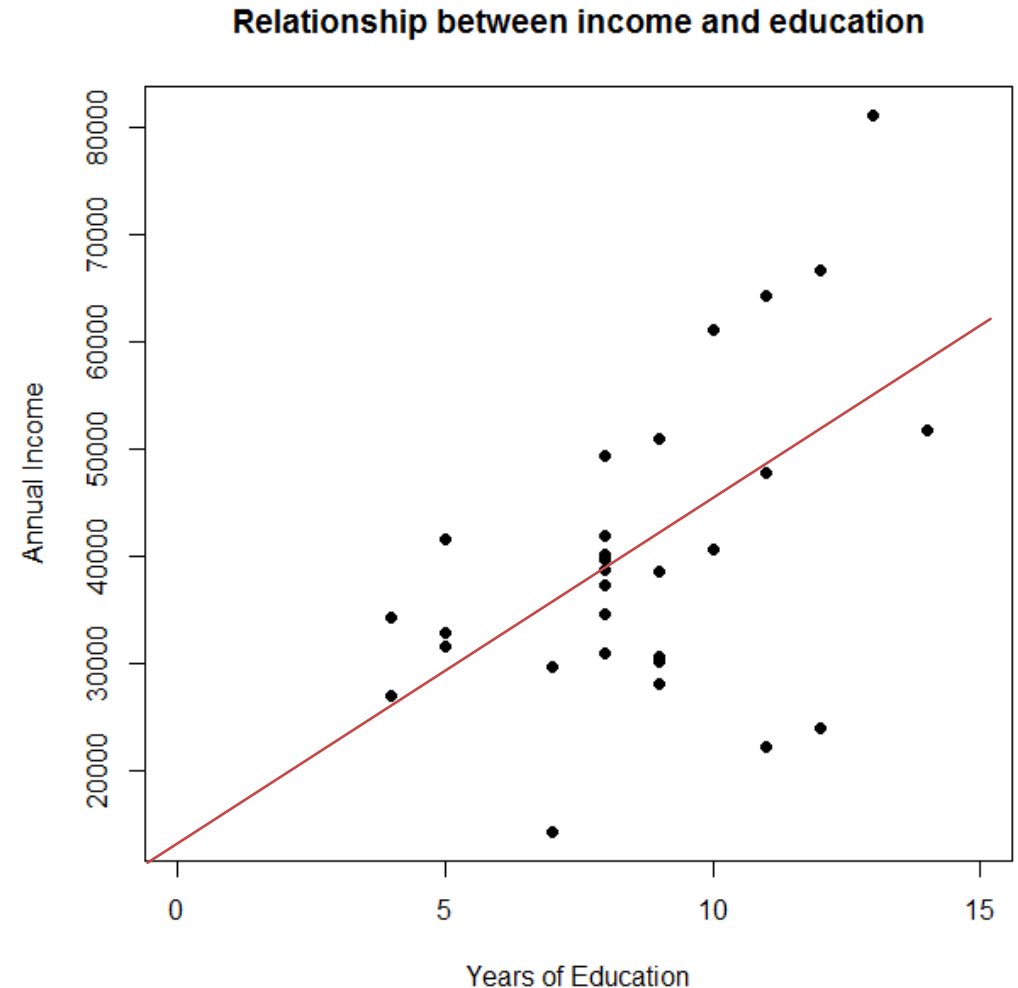
- $\beta$

- Cost Function

- $J(\beta) = \frac{1}{2N} \sum_{i=1}^N (\beta X_i - Y_i)^2$

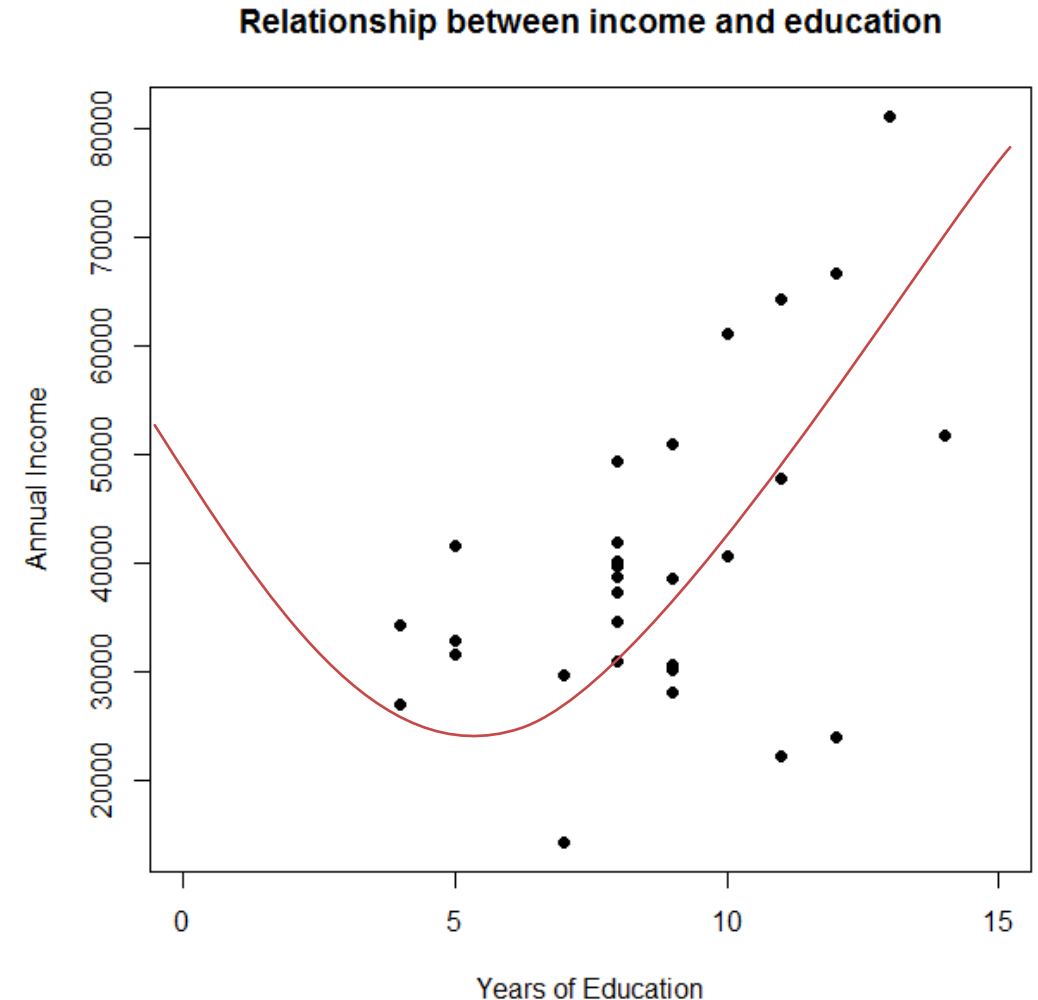
- Objective

- $\min_{\beta} J(\beta)$



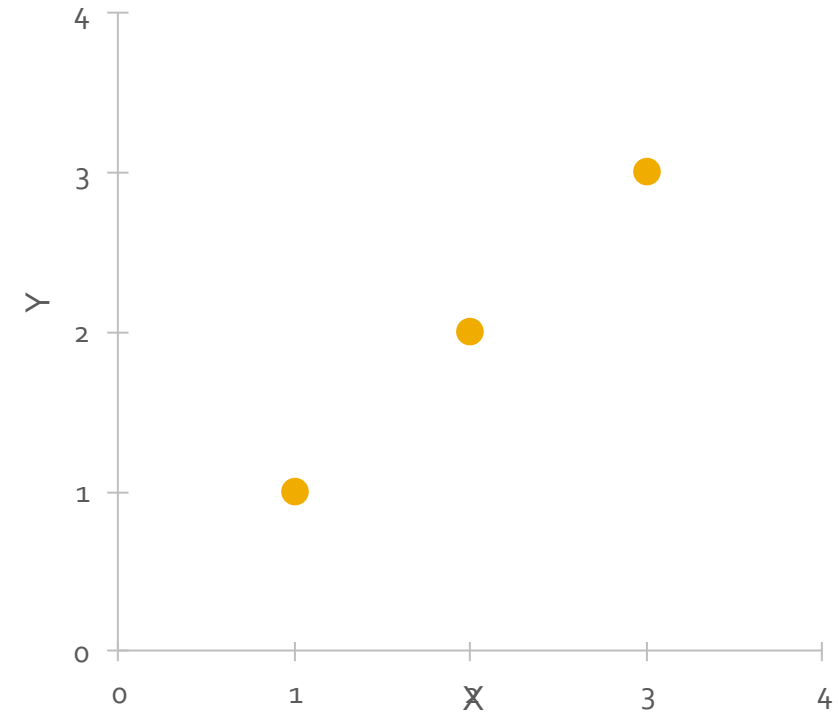
# Fill in the blanks

- Model (“hypothesis”)
  - Income depends on education, and also on education<sup>2</sup>, i.e. nonlinearities exist
  - $Y_i = \alpha + \beta X_i + \gamma X_i^2$
- Parameters
  - $\alpha, \beta, \gamma$
- Cost Function
  - Use “absolute error” cost function
  - $J(\alpha, \beta, \gamma) = \frac{1}{N} \sum_{i=1}^N |\alpha + \beta X_i + \gamma X_i^2 - Y_i|$
- Objective
  - $\min_{\alpha, \beta, \gamma} J(\alpha, \beta, \gamma)$



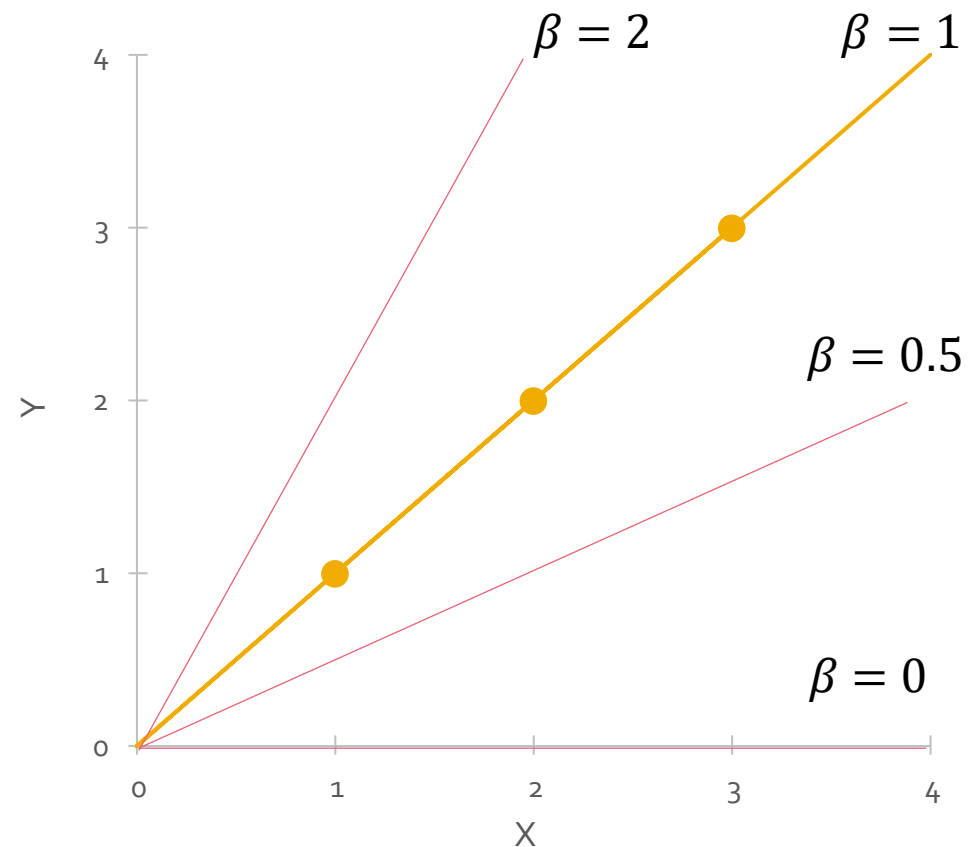
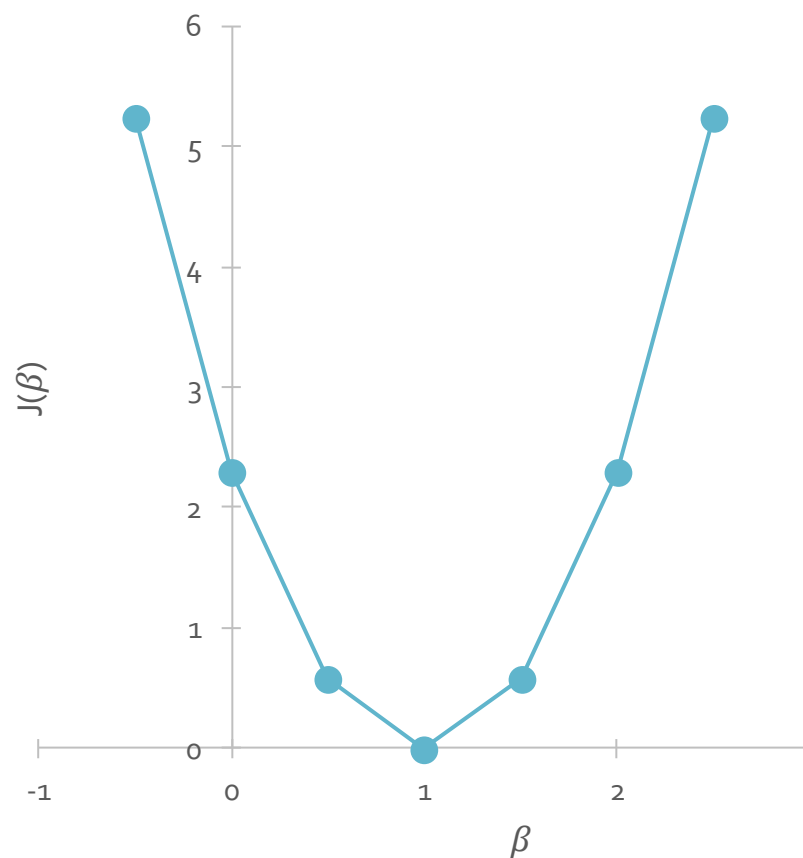
# Exercise: Computing Cost

- Assume our data look like this ( $N = 3$ )
  - $X_1 = 1, Y_1 = 1$
  - $X_2 = 2, Y_2 = 2$
  - $X_3 = 3, Y_3 = 3$
- Our model is  $Y_i = \beta X_i$ 
  - Our cost function is squared error:  $J(\beta) = \sum_{i=1}^N (\beta X_i - Y_i)^2$
- Your task is to compute  $J(\beta)$ , given these 3 points, for:
  - $\beta = 1$
  - $\beta = 0$
  - $\beta = 2$
  - $\beta = 0.5$  (you might need a calculator)
- Draw a plot of  $J(\beta)$  as a function of  $\beta$



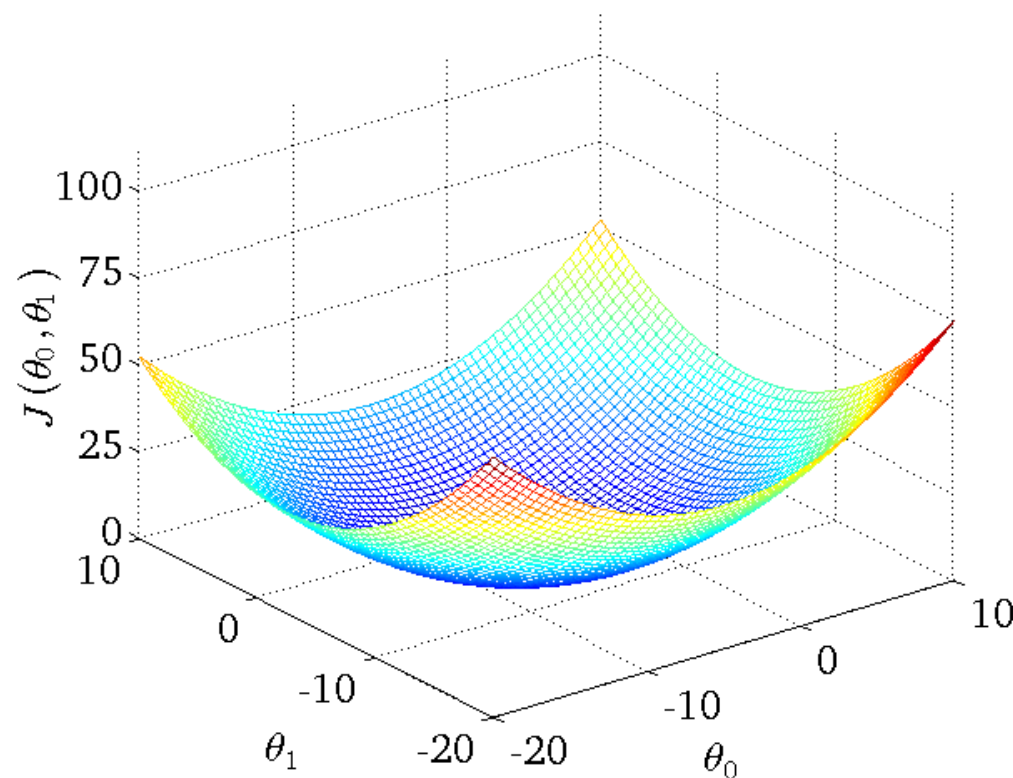
# Visualizing Cost (1 parameter)

- Where is  $J(\beta)$  minimized?



# Visualizing Cost (2 parameters)

- Generalizing to a multi-dimensional loss surface
  - Model ("hypothesis"):  $Y_i = \theta_1 + \theta_2 X_i$
  - Objective:  $\min_{\theta_1, \theta_2} J(\theta_1, \theta_2)$



# Outline

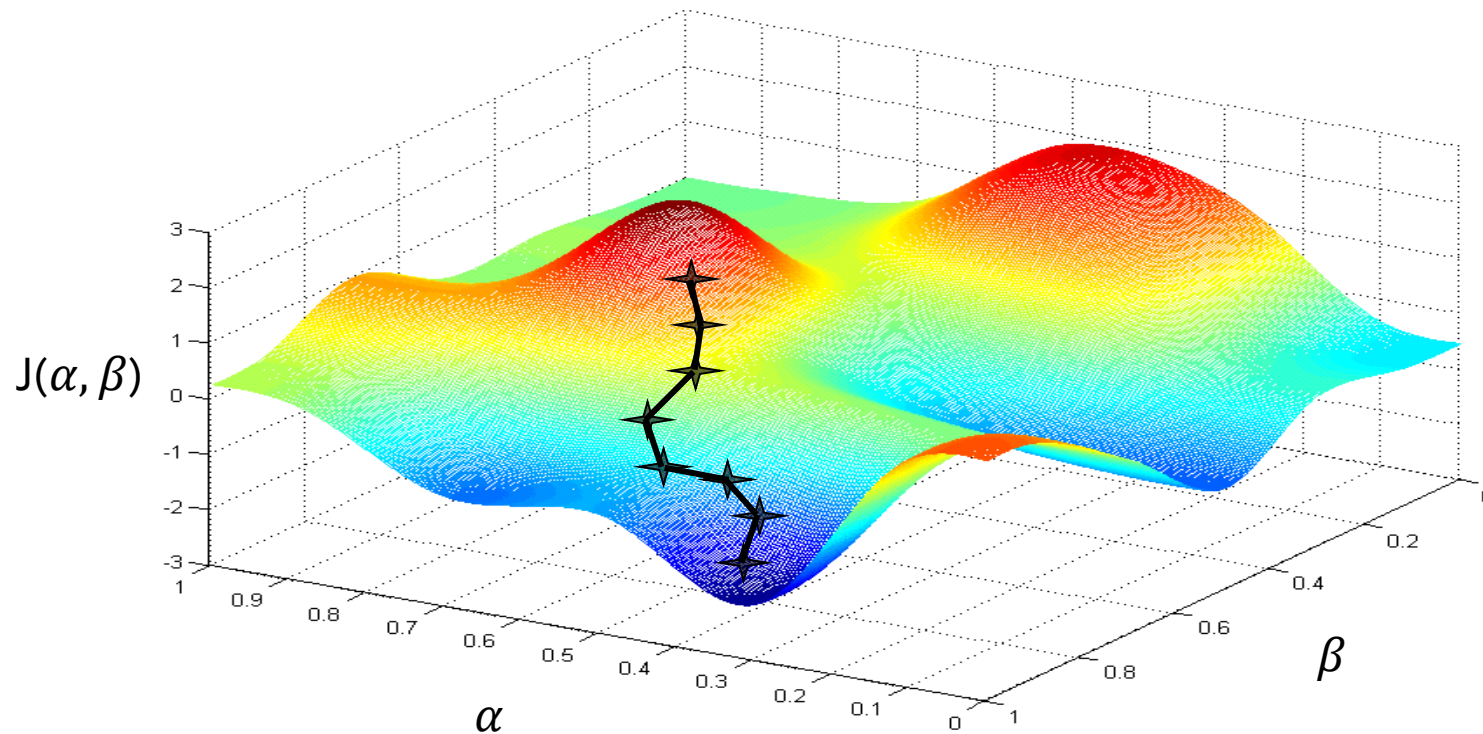
- Cost functions
- **Gradient descent**
- Feature scaling

# Gradient Descent

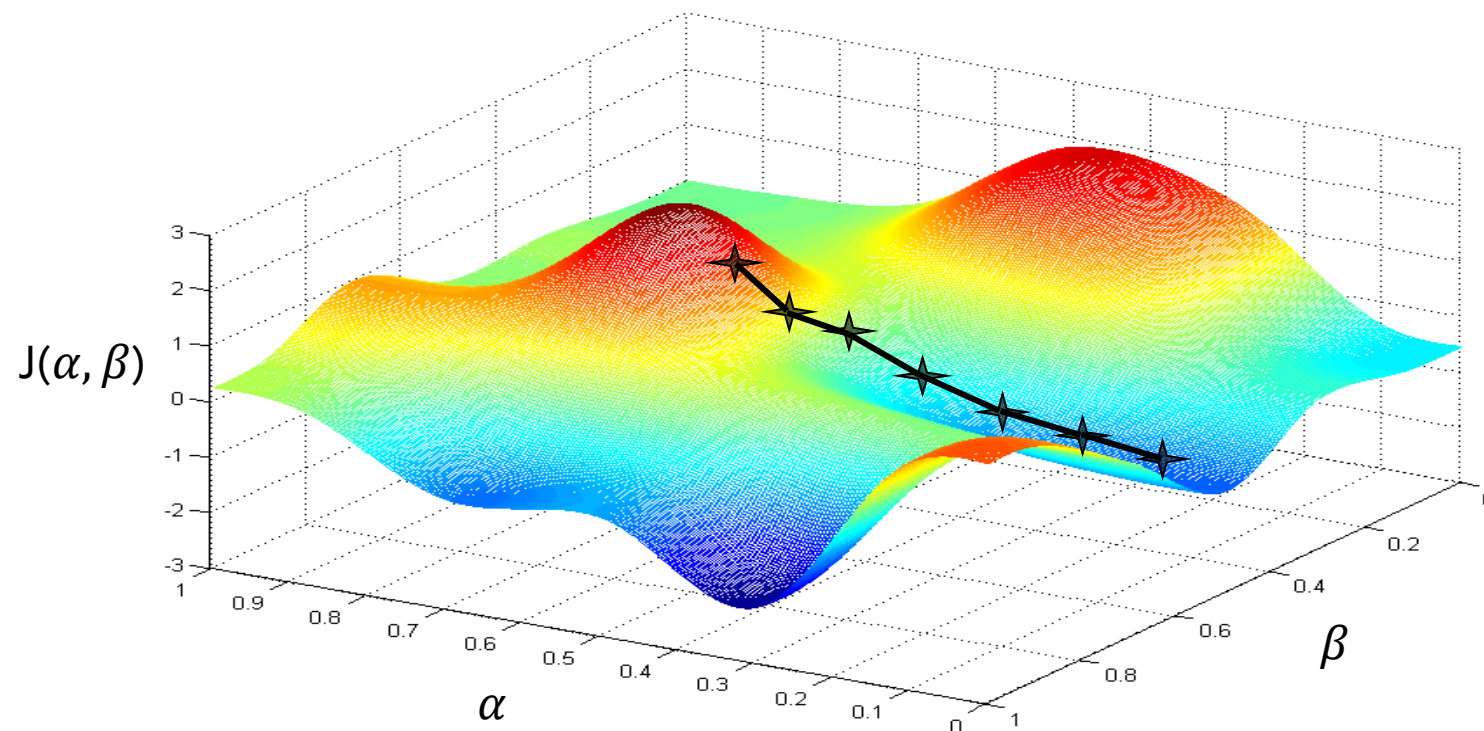
- Gradient descent provides a principled method/algorithm to minimize the cost function  $J$
- Idea: to solve  $\min_{\alpha, \beta} J(\alpha, \beta)$ 
  - Initialize  $\alpha, \beta$
  - Change  $\alpha, \beta$  in some way that reduces  $J(\alpha, \beta)$
  - Eventually we will end up at a minimum
- What about analytic solution:  $(X'X)^{-1}X'Y$ 
  - Sometimes not practically feasible (too much data, multicollinearity)



# Gradient Descent: Visualization



# Local Minima



# Gradient Descent Algorithm

- In pseudo-code:

Choose an initial vector of parameters  $\alpha, \beta$

Choose learning rate  $R$

Repeat until convergence (i.e., until an approximate minimum is obtained):

For each example  $i$  in training set:

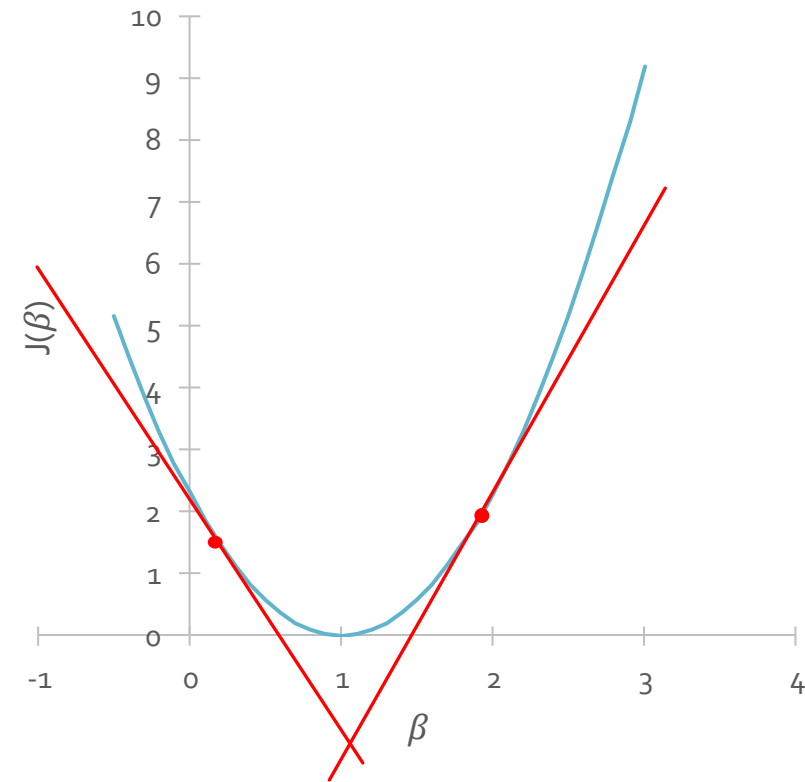
$$\left. \begin{aligned} \alpha &\leftarrow \alpha - R \frac{\partial}{\partial \alpha} J(\alpha, \beta) \\ \beta &\leftarrow \beta - R \frac{\partial}{\partial \beta} J(\alpha, \beta) \end{aligned} \right\} \text{ Simultaneous update}$$

- With multiple predictors/regressors...

- $Y_i = \alpha + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots + \beta_k X_{ik}$

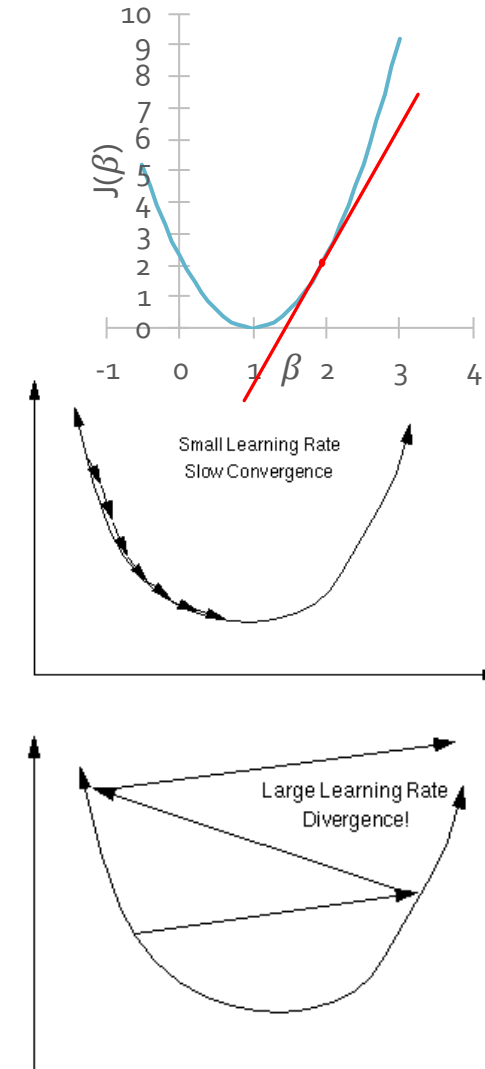
# Gradient Descent: Derivative

- In 1-Dimension
- Update Rule:
  - $\beta \leftarrow \beta - \mathbb{R} \frac{\partial}{\partial \beta} J(\alpha, \beta)$
- Initialize  $\beta$  at 1.9
  - What's the derivative  $\frac{\partial}{\partial \beta} J(\alpha, \beta)$ ?
  - How does  $\beta$  update?
- Initialize  $\beta$  at 0.2
  - What's the derivative  $\frac{\partial}{\partial \beta} J(\alpha, \beta)$ ?
  - How does  $\beta$  update?



# Gradient Descent: Learning Rate

- $\beta \leftarrow \beta - R \frac{\partial}{\partial \beta} J(\alpha, \beta)$
- What does R do?
- Small R:
  - Gradient descent can be slow
- Large R:
  - Can overshoot the minimum
  - May fail to converge
  - May diverge!



# Gradient Descent: Convergence

- Do we need to change the learning rate?

Choose an initial vector of parameters  $\alpha, \beta$

Choose learning rate  $R$

Repeat until convergence:

For each example  $i$ :

$$\alpha \leftarrow \alpha - R \frac{\partial}{\partial \alpha} J(\alpha, \beta)$$

$$\beta \leftarrow \beta - R \frac{\partial}{\partial \beta} J(\alpha, \beta)$$

- No! Gradient descent can converge to a local minimum, even with the learning rate fixed
  - As we approach a local minimum, gradient descent takes smaller steps

# Gradient Descent: Regression

- Gradient Descent

Repeat until convergence:

$$\alpha \leftarrow \alpha - \text{R} \frac{\partial}{\partial \alpha} J(\alpha, \beta)$$

$$\beta \leftarrow \beta - \text{R} \frac{\partial}{\partial \beta} J(\alpha, \beta)$$

- Regression cost function

- $J(\alpha, \beta) = \frac{1}{2N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i)^2$

- The missing pieces:  $\frac{\partial}{\partial \alpha} J(\alpha, \beta)$  and  $\frac{\partial}{\partial \beta} J(\alpha, \beta)$

- $\frac{\partial}{\partial \alpha} J(\alpha, \beta) = \frac{\partial}{\partial \alpha} \frac{1}{2N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i)^2$

- $\frac{\partial}{\partial \beta} J(\alpha, \beta) = \frac{\partial}{\partial \beta} \frac{1}{2N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i)^2$

# Gradient Descent: Regression

- Partial derivatives:

$$\begin{aligned}
 \frac{\partial}{\partial \alpha} J(\alpha, \beta) &= \frac{\partial}{\partial \alpha} \frac{1}{2N} \sum_{i=1}^N (\hat{Y}_i - Y_i)^2 \\
 &= \frac{\partial}{\partial \alpha} \frac{1}{2N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i)^2 \\
 &= \frac{1}{2N} \sum_{i=1}^N \frac{\partial}{\partial \alpha} (\alpha + \beta X_i - Y_i)^2 \\
 &= \frac{1}{2N} \sum_{i=1}^N 2(\alpha + \beta X_i - Y_i) \frac{\partial}{\partial \alpha} (\alpha + \beta X_i - Y_i) \\
 &= \frac{1}{N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i) \\
 &= \frac{1}{N} \sum_{i=1}^N (\hat{Y}_i - Y_i)
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial}{\partial \beta} J(\alpha, \beta) &= \frac{\partial}{\partial \beta} \frac{1}{2N} \sum_{i=1}^N (\hat{Y}_i - Y_i)^2 \\
 &= \frac{1}{2N} \sum_{i=1}^N \frac{\partial}{\partial \beta} (\alpha + \beta X_i - Y_i)^2 \\
 &= \frac{1}{2N} \sum_{i=1}^N 2(\alpha + \beta X_i - Y_i) \frac{\partial}{\partial \beta} (\alpha + \beta X_i - Y_i) \\
 &= \frac{1}{N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i) (X_i) \\
 &= \frac{1}{N} \sum_{i=1}^N (\hat{Y}_i - Y_i) X_i
 \end{aligned}$$



# Gradient Descent: Regression

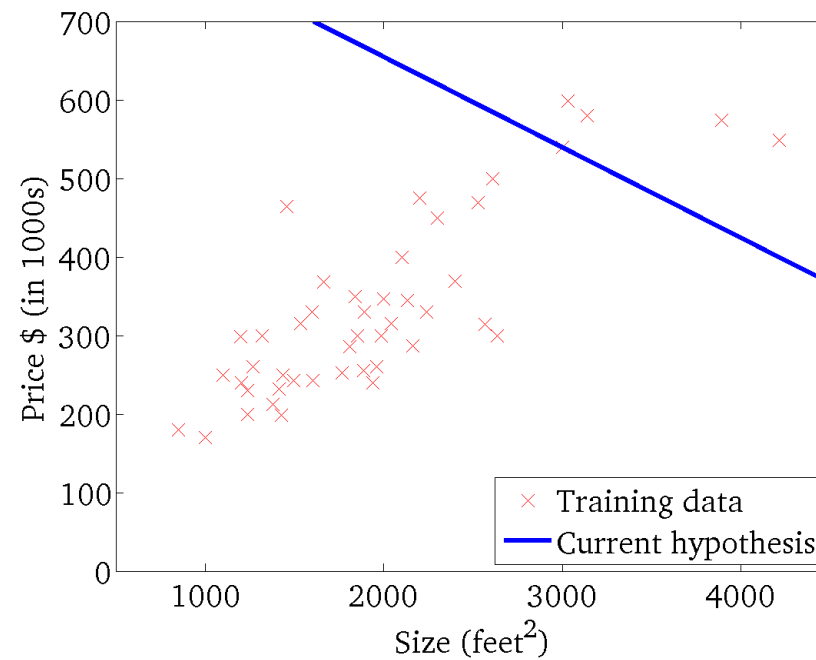
- Gradient Descent Algorithm (linear regression)

Repeat until convergence:

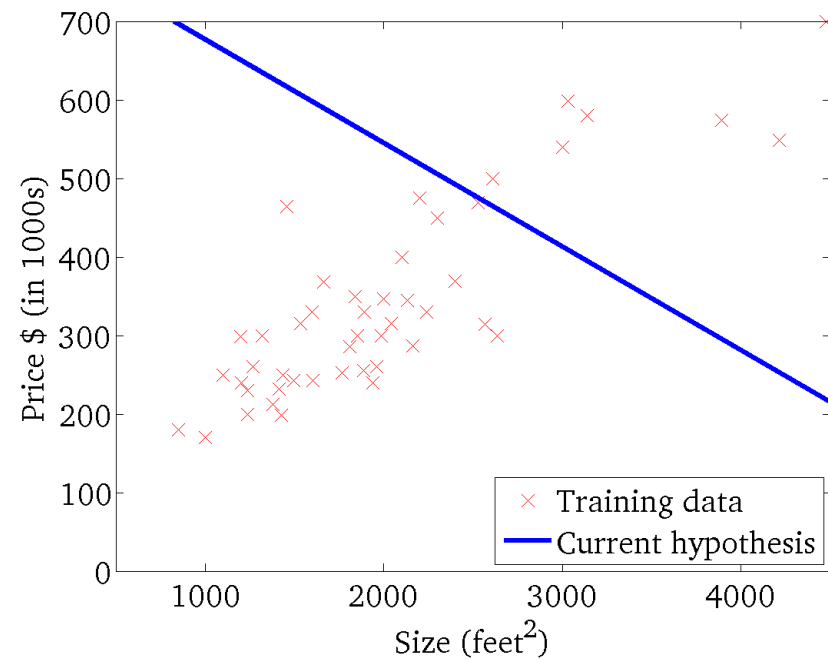
$$\alpha \leftarrow \alpha - \mathbb{R} \frac{1}{N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i)$$

$$\beta \leftarrow \beta - \mathbb{R} \frac{1}{N} \sum_{i=1}^N (\alpha + \beta X_i - Y_i) X_i$$

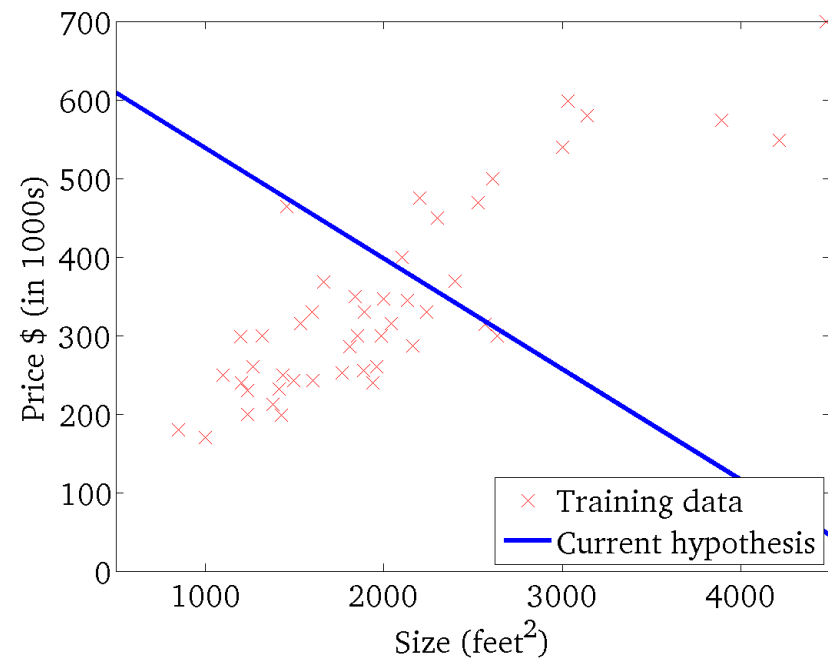
# Gradient Descent: In Action



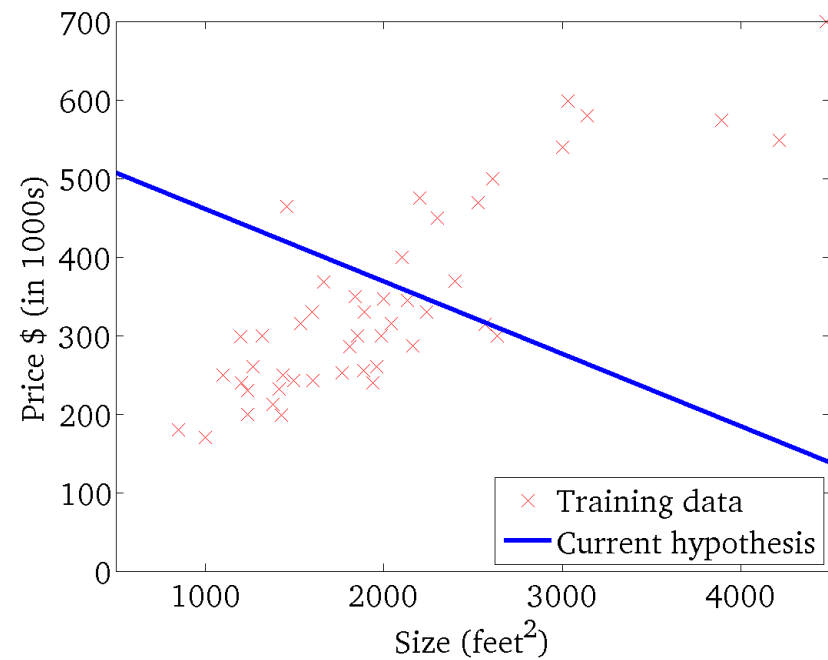
# Gradient Descent: In Action



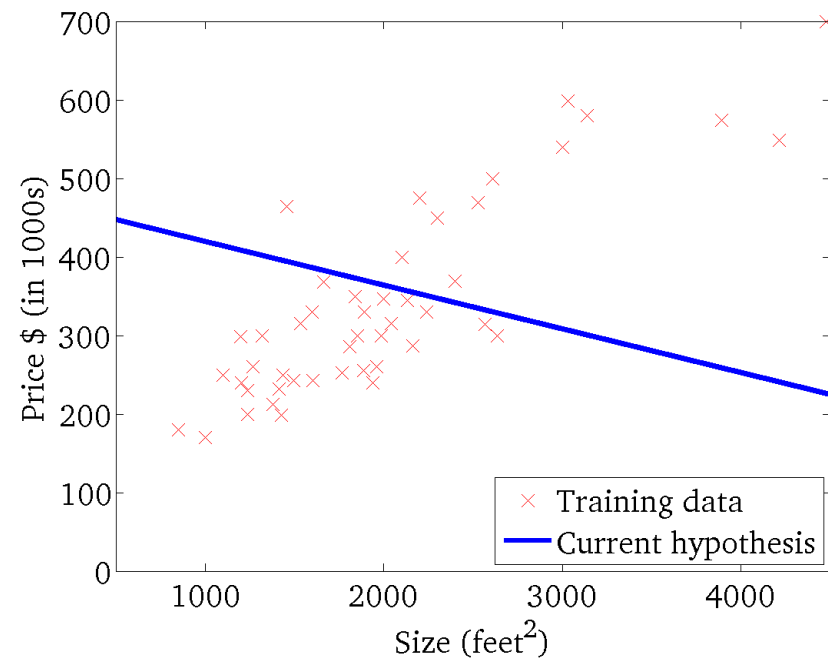
# Gradient Descent: In Action



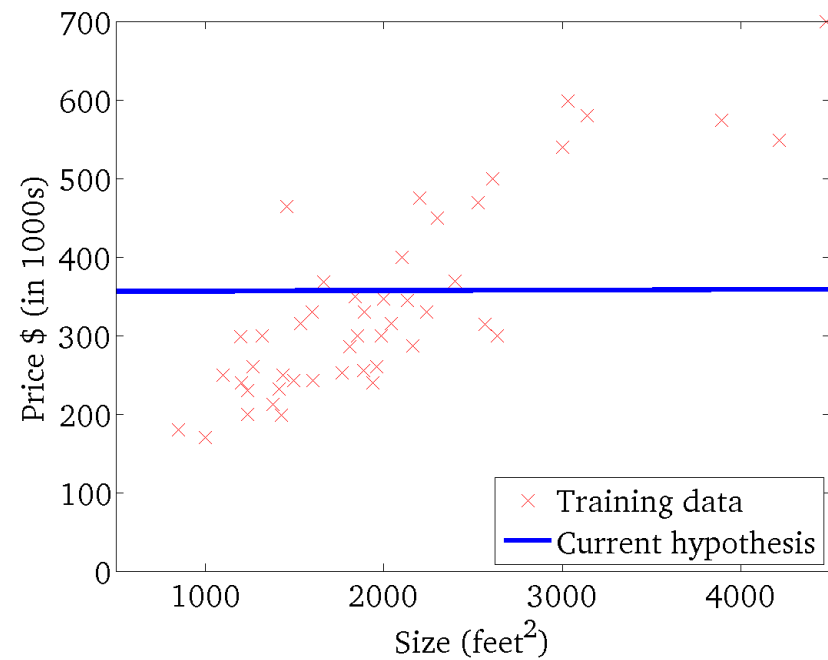
# Gradient Descent: In Action



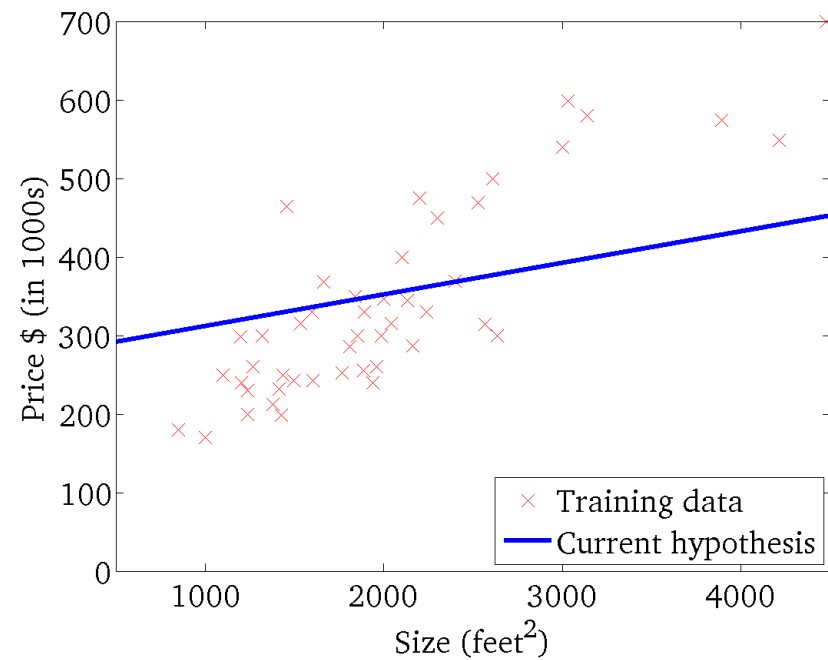
# Gradient Descent: In Action



# Gradient Descent: In Action

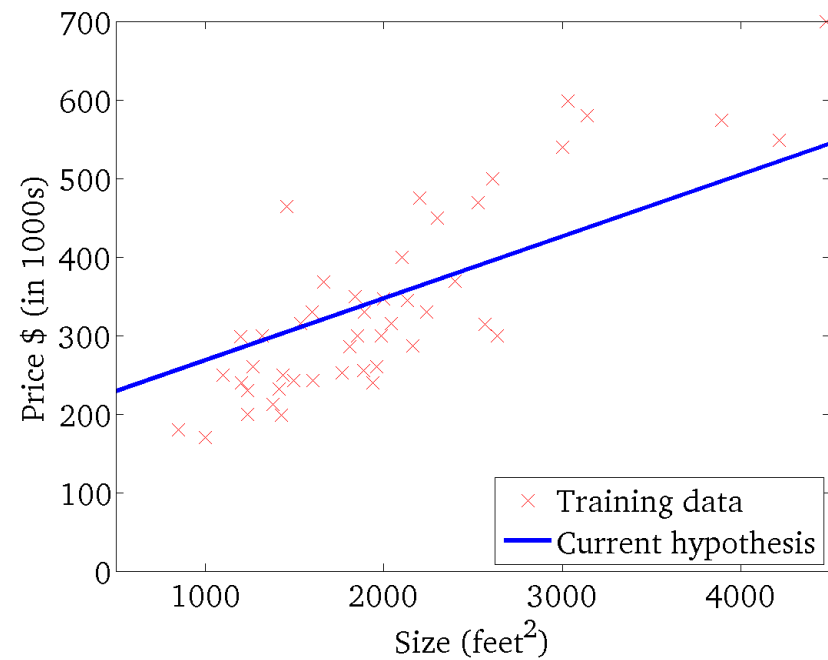


# Gradient Descent: In Action

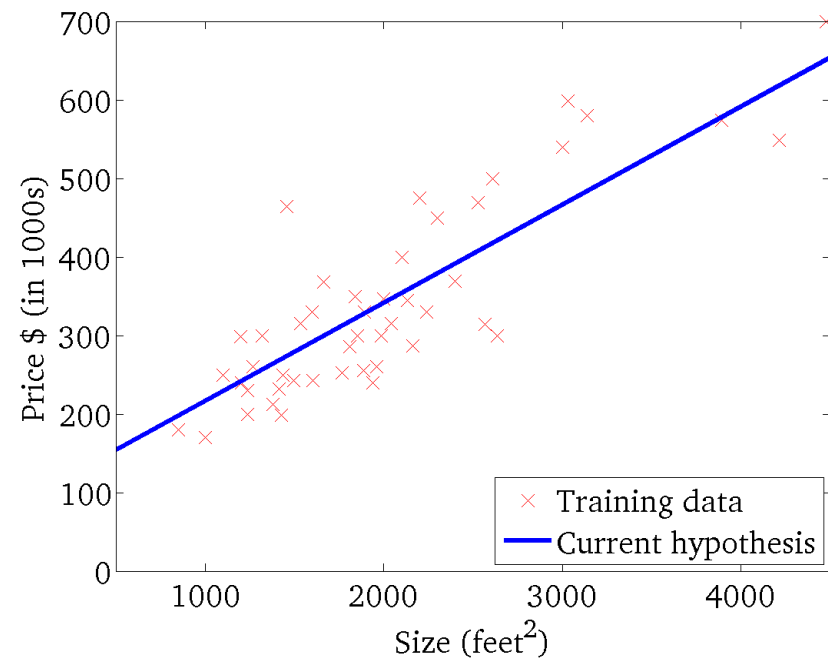




# Gradient Descent: In Action



# Gradient Descent: In Action

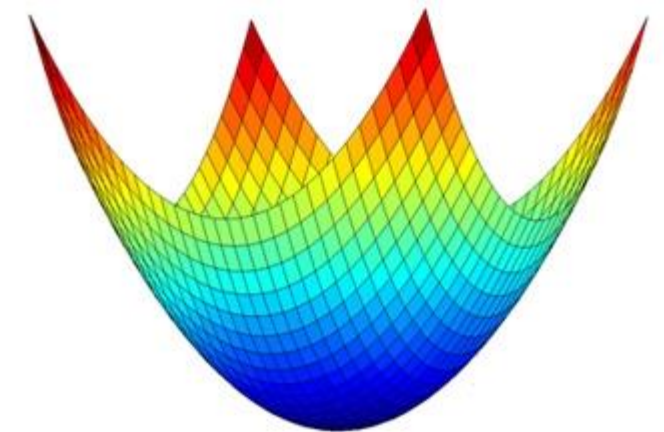
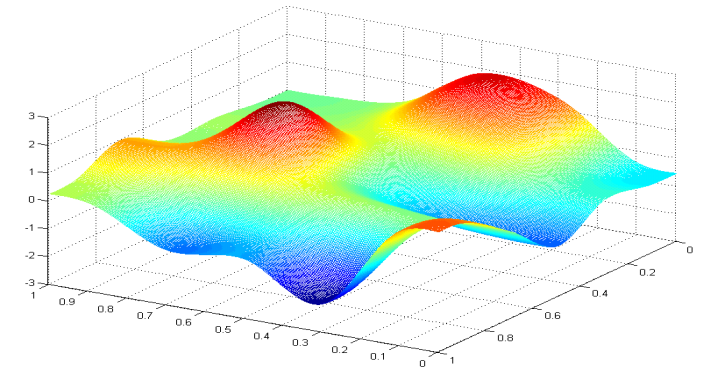


# Outline

- Cost functions
- Gradient descent
  - **Local Minima**
  - Batch and Incremental versions
- Feature scaling

# Local Minima?

- What about local minima in gradient descent for regression?
- No problem!
- Cost function in regression is **convex**
  - Convex: a continuous function where the midpoint of any interval doesn't exceed the mean of the endpoints
  - (second derivative is non-negative)



# Outline

- Cost functions
- Gradient descent
  - Local Minima
  - **Batch and Incremental versions**
- Feature scaling

# Incremental vs. Batch Gradient Descent

- In “Batch” gradient descent

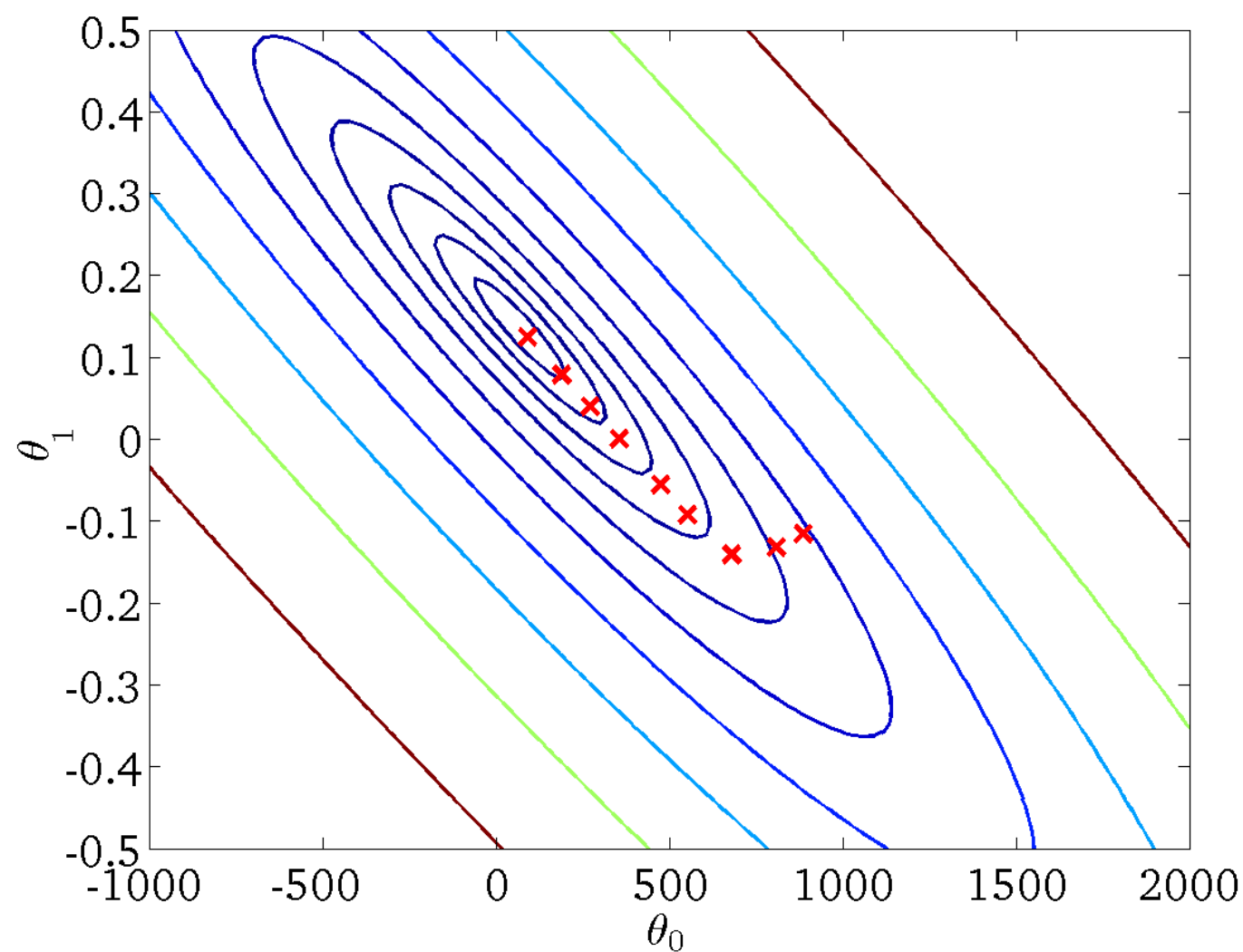
Repeat until convergence:

$$\begin{aligned} \text{Compute } \nabla\alpha &= \frac{\partial}{\partial\alpha}J(\alpha, \beta) = \frac{1}{N}\sum_{i=1}^N(\alpha + \beta X_i - Y_i) \\ \text{Compute } \nabla\beta &= \frac{\partial}{\partial\beta}J(\alpha, \beta) = \frac{1}{N}\sum_{i=1}^N(\alpha + \beta X_i - Y_i)X_i \end{aligned} \quad \left. \vphantom{\begin{aligned} \text{Compute } \nabla\alpha \\ \text{Compute } \nabla\beta \end{aligned}} \right\} \text{Global gradient, computed over all training data}$$

$$\begin{aligned} \alpha &\leftarrow \alpha - R\nabla\alpha \\ \beta &\leftarrow \beta - R\nabla\beta \end{aligned} \quad \left. \vphantom{\begin{aligned} \alpha &\leftarrow \alpha - R\nabla\alpha \\ \beta &\leftarrow \beta - R\nabla\beta \end{aligned}} \right\} \text{Single, simultaneous update}$$

- Note: each step uses all training examples!

# Batch Gradient Descent



# Incremental vs. Batch Gradient Descent

- “Iterative” (stochastic) version of gradient descent:

Repeat until an approximate minimum is obtained:

Randomly shuffle examples in the training set

For each example  $i$ :

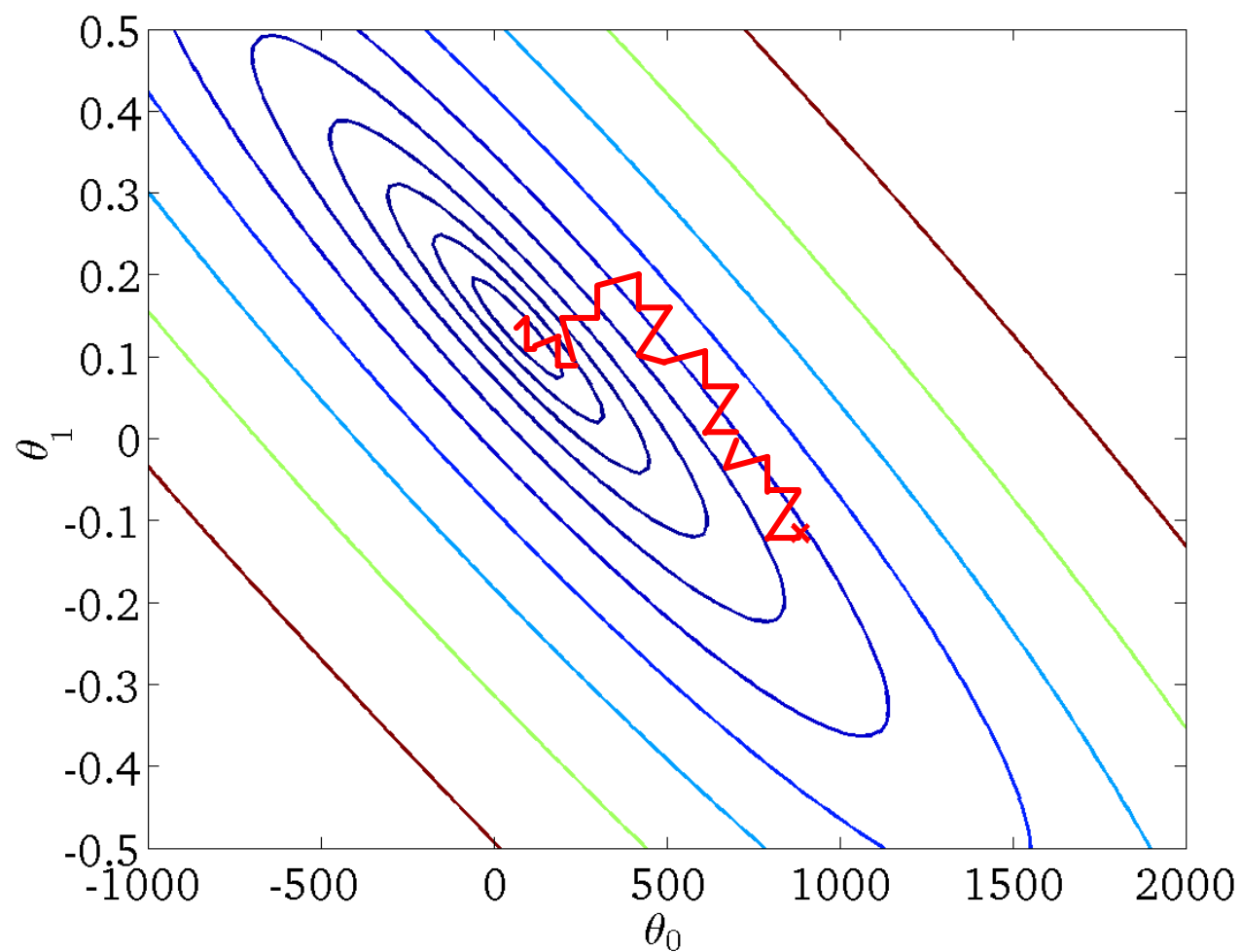
$\alpha \leftarrow \alpha - \mathbb{R} \frac{\partial}{\partial \alpha} J(\alpha, \beta)$       // evaluate  $\frac{\partial}{\partial \alpha} J(\alpha, \beta)$  at  $\mathbf{x}_i$  and update  $\alpha$

$\beta \leftarrow \beta - \mathbb{R} \frac{\partial}{\partial \beta} J(\alpha, \beta)$       // evaluate  $\frac{\partial}{\partial \beta} J(\alpha, \beta)$  at  $\mathbf{x}_i$  and update  $\beta$

- The parameters are adjusted with each training instance, iteratively
- Also: Mini-batch



# Stochastic Gradient Descent

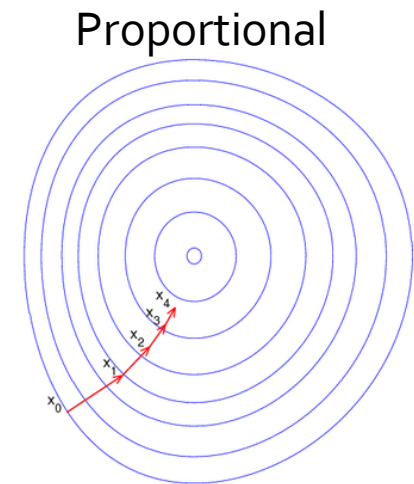
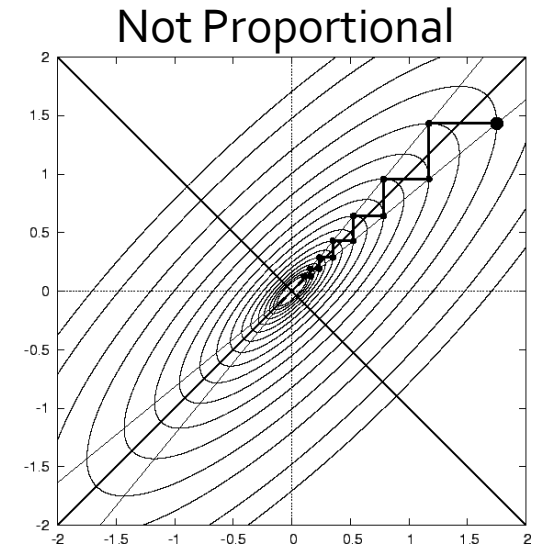


# Outline

- Cost functions
- Gradient descent
- **Feature scaling**

# Feature scaling

- In gradient descent, we “take a step” in the direction where the decrease in cost is greatest
- When some features (axes) are on different scales, gradient descent can be inefficient
  - Putting different features on same scale can make gradient descent much faster



# Feature scaling

- Feature scaling is an important pre-processing step for many common machine learning algorithms

- Standardization: (mean 0 standard dev. 1)

$$x'_{ik} = \frac{x_{ik} - \bar{x}_k}{s_k}$$

- $s_k$  is standard deviation of  $x_k$ , or range (max-min) of  $x_k$

- Also common: force feature to be roughly between -1 and 1:

$$x'_{ik} = \frac{x_{ik}}{\max(|x_k|)}$$

# Key Concepts (today's lecture)

- Cost Functions
- Gradient Descent
- Local and global minima
- Convex functions
- Incremental vs. Batch GD
- Learning rates
- Feature scaling