Today: Outline

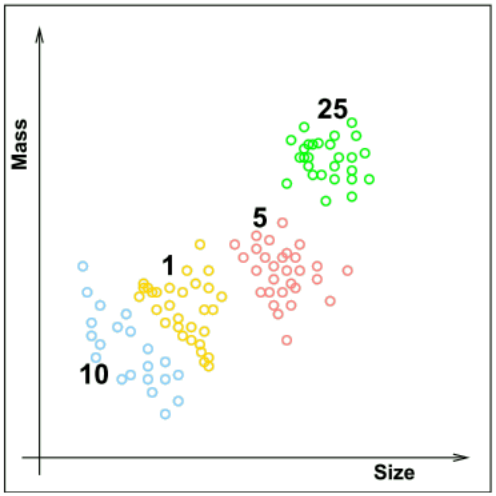
- Piazza Walk Through
- Machine Learning Review 1
  - Cost Functions
  - Regression
  - Classification
- Reminder: Please email me ASAP if you have not received a Piazza enrollment email.

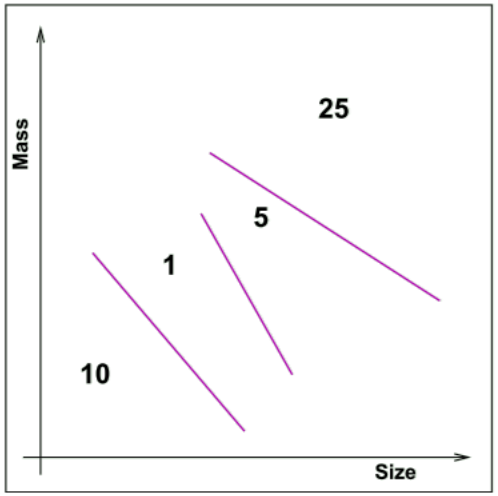
Supervised Learning

Task Definitions: Classification and Regression Tasks

Example of Supervised Learning:

Classification: recognize coins

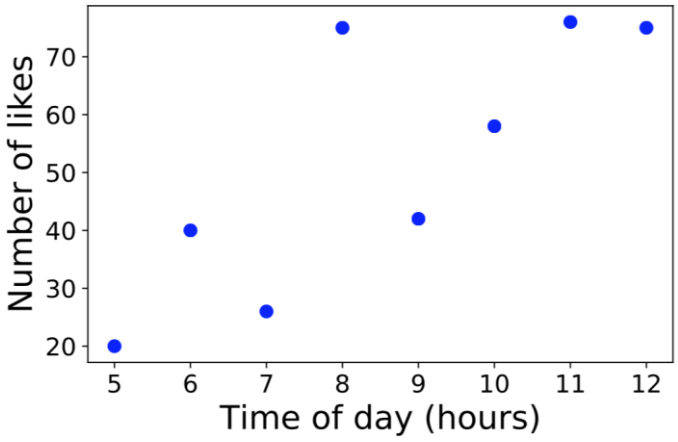


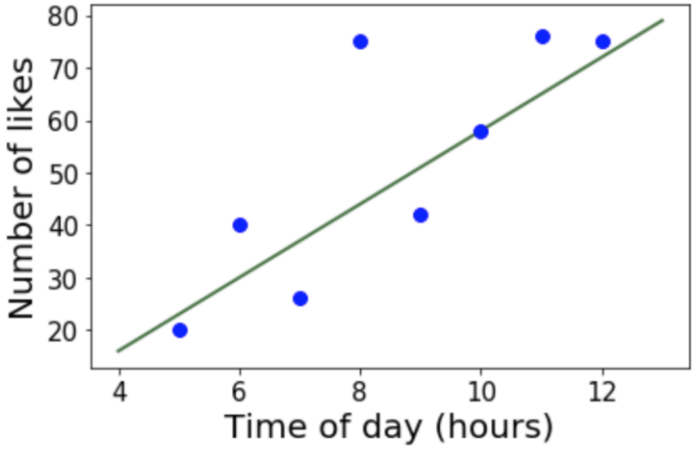


- Given training set consisting of coin denomination (penny, nickel, dime, quarter), mass and size
- Learn to predict denomination
- What is input? Output?

Example of Supervised Learning:
Linear Regression: predict number of likes

- When the label is a real number
- Training a **model** to find a relationship between input and output values
- Learning a **line of best fit**



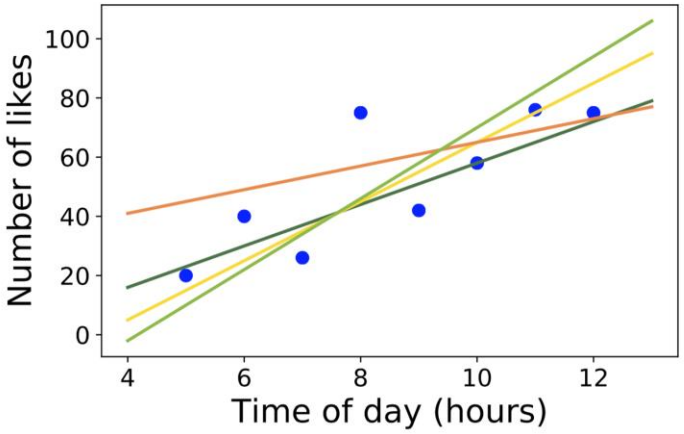


Linear Regression: Model Parameters

- Learning a best fit line means learning parameters (or weights) for our model.

$$h_\theta(x) = \theta_0 + \theta_1 x$$

θ_i's: Parameters

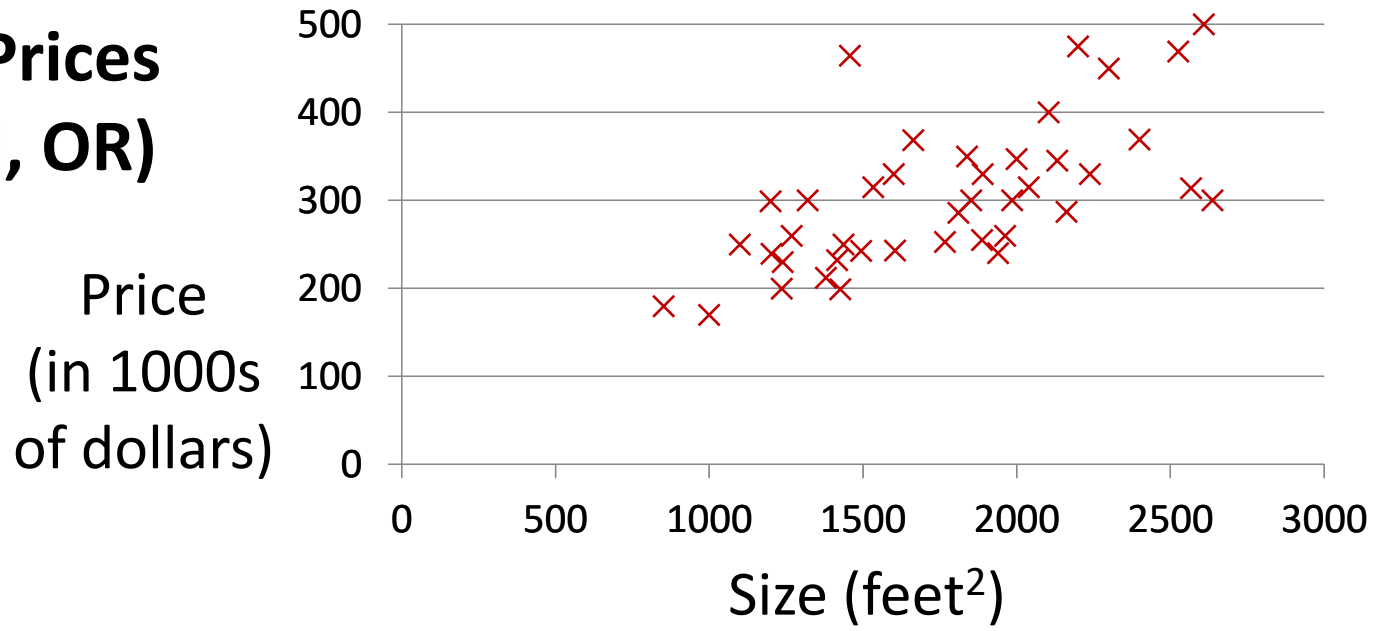


Supervised Learning

Regression

Example: house price prediction

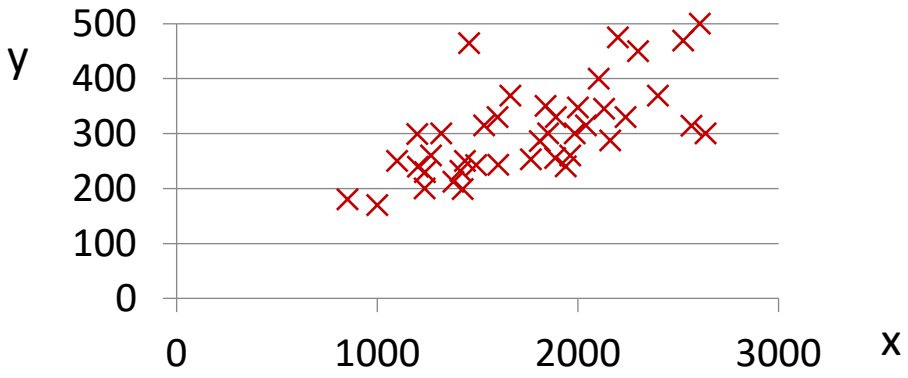
Housing Prices (Portland, OR)



Supervised Learning

What should the learner be??

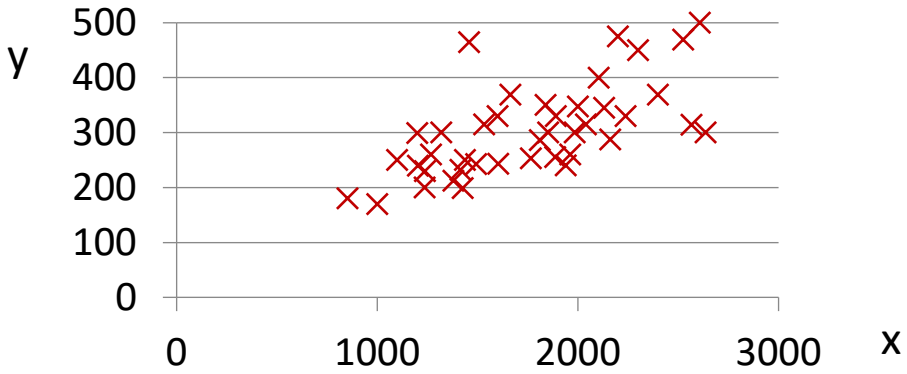
Want: input x → **?** → output y



Hypothesis *h*

h : a function parametrized by ϑ

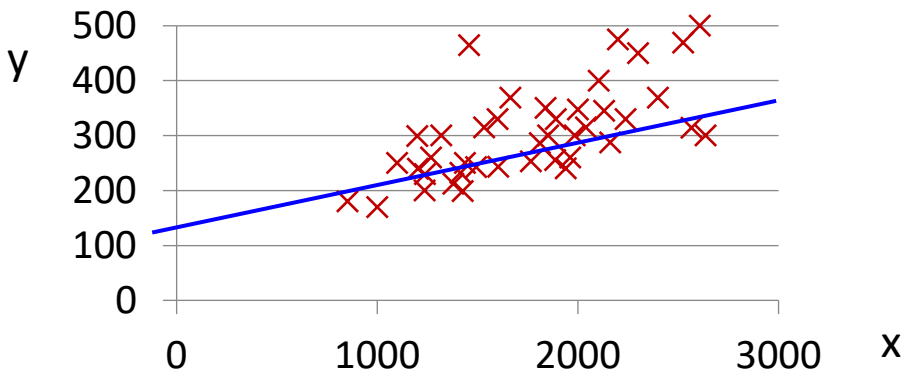
Want:



How to learn ϑ ?

Given: Training Set $\{x^i, y^i\}$ **But what if $y \neq y^i$??**

Want: input x^i → h_θ → output y



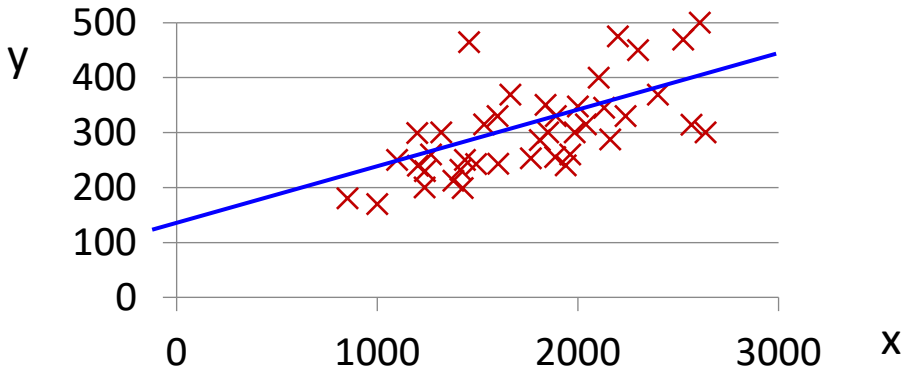
Cost function

Given: Training Set $\{x^i, y^i\}$

Cost function $Cost(y, y^i)$

learning == minimizing cost

Want: input x^i → h_θ → output y



Supervised Learning

Given: Training Set $\{x^i, y^i\}$

Cost function $\text{Cost}(y, y^i)$

learning == minimizing cost

Learn θ^*: $\min_{\theta} \text{Cost}(h_\theta(x^i), y^i)$

Want: input x^i → h_{θ^*} → output y

Training set

Training set:

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Notation:

m = Number of training examples

$x^{(i)}$ = "input" variable / features

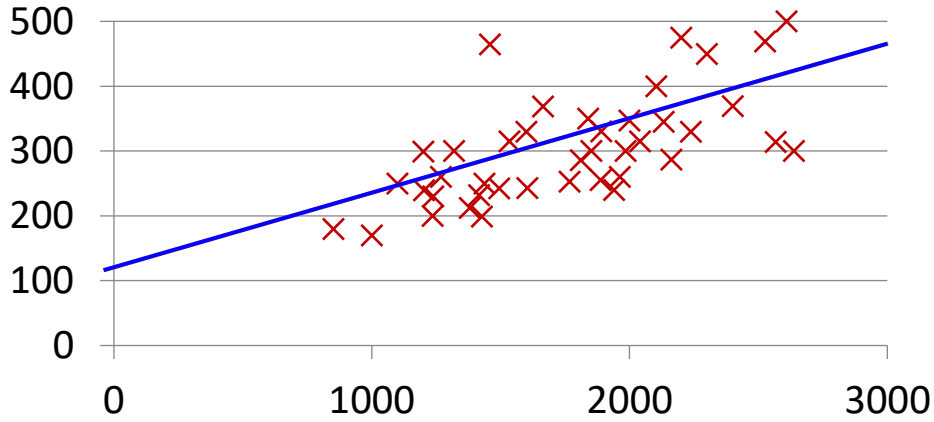
$y^{(i)}$ = "output" variable / "target" variable

What should *h* be?

Linear hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

θ_i's: Parameters



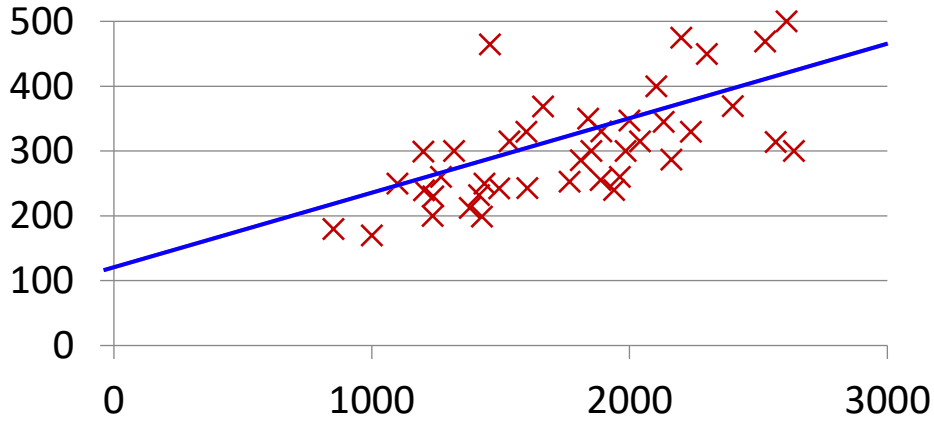
$$\min_{\theta} \text{Cost}(h_\theta, \{x^i, y^i\})$$

What's a good cost function for this problem?

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

θ_i's: Parameters



How about "Sum of squared differences"

Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

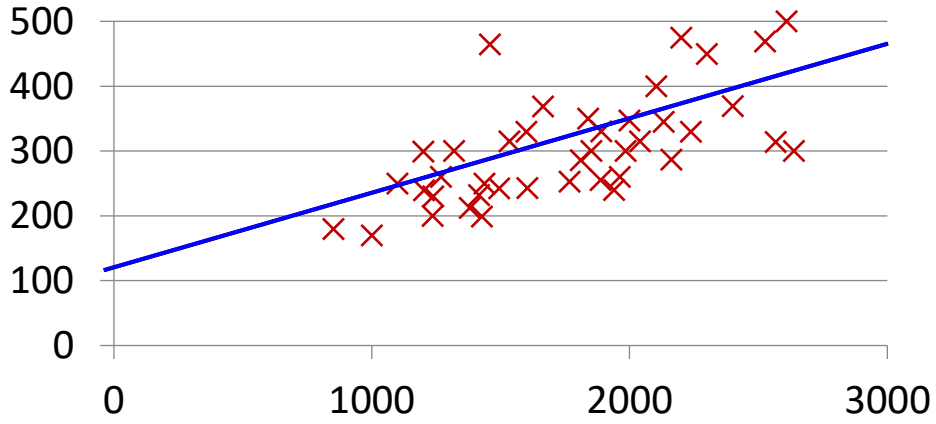
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

2-dimensional θ

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

θ_i's: Parameters



Cost Function:

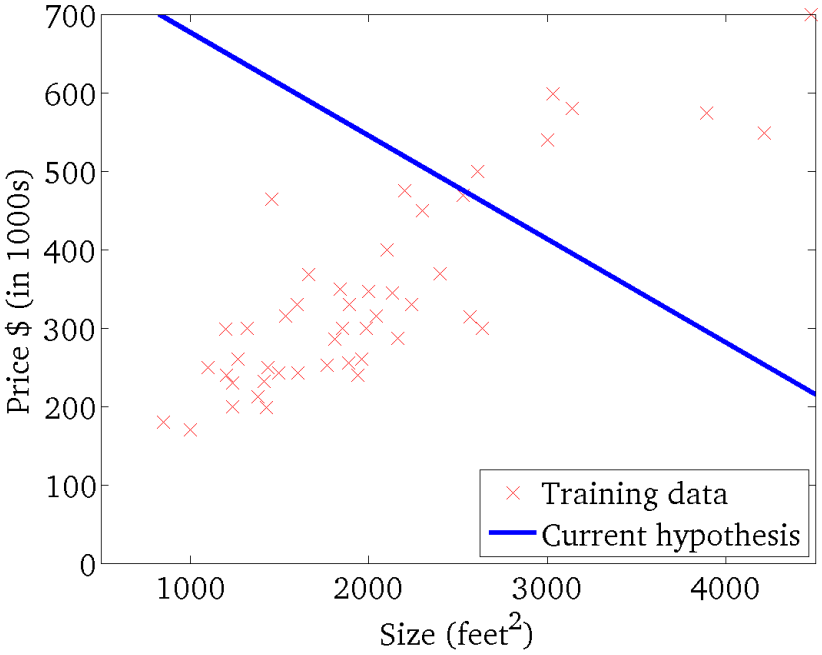
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

SSD = sum of squared differences, also
SSE = sum of squared errors

Plotting cost for 2-dimensional θ

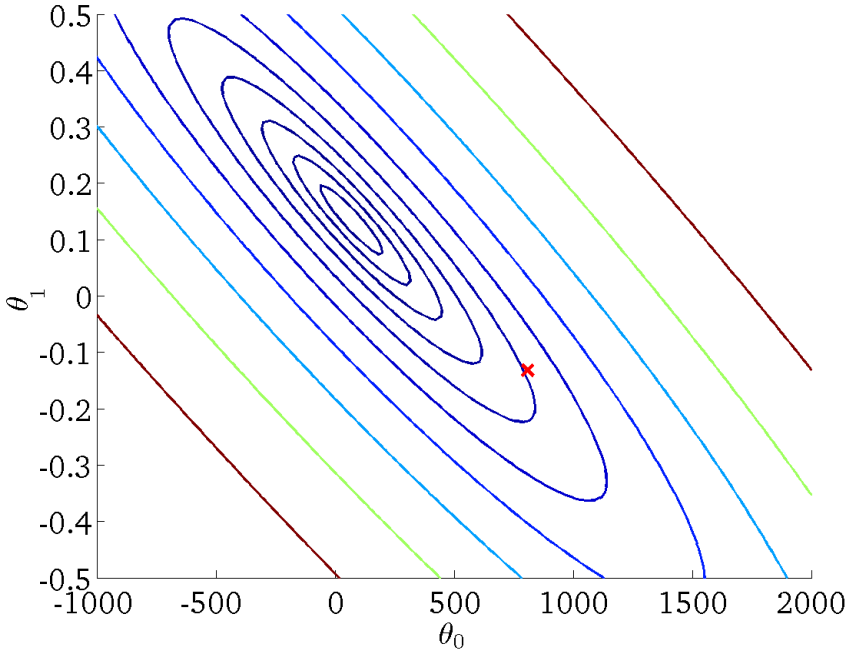
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

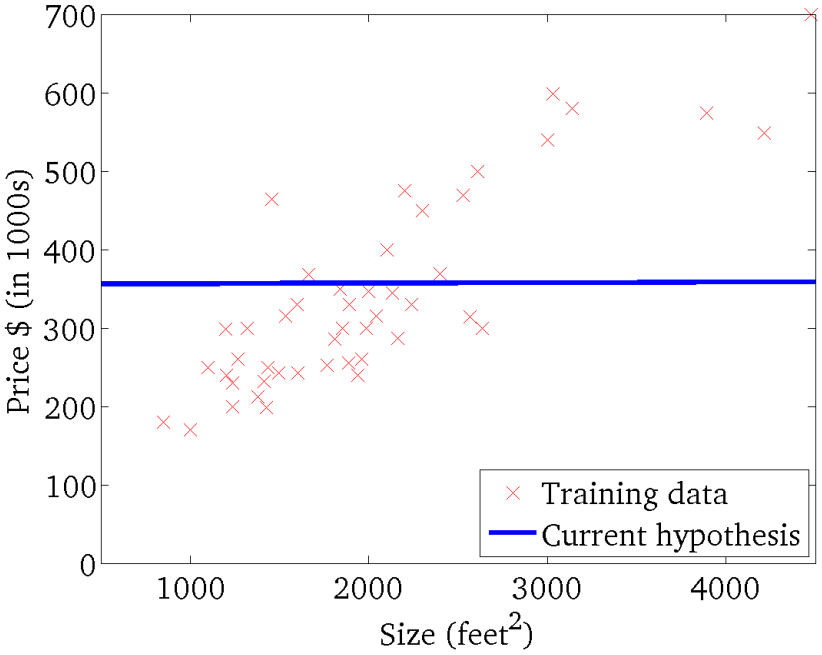
(function of the parameters θ_0, θ_1)



Plotting cost for 2-dimensional θ

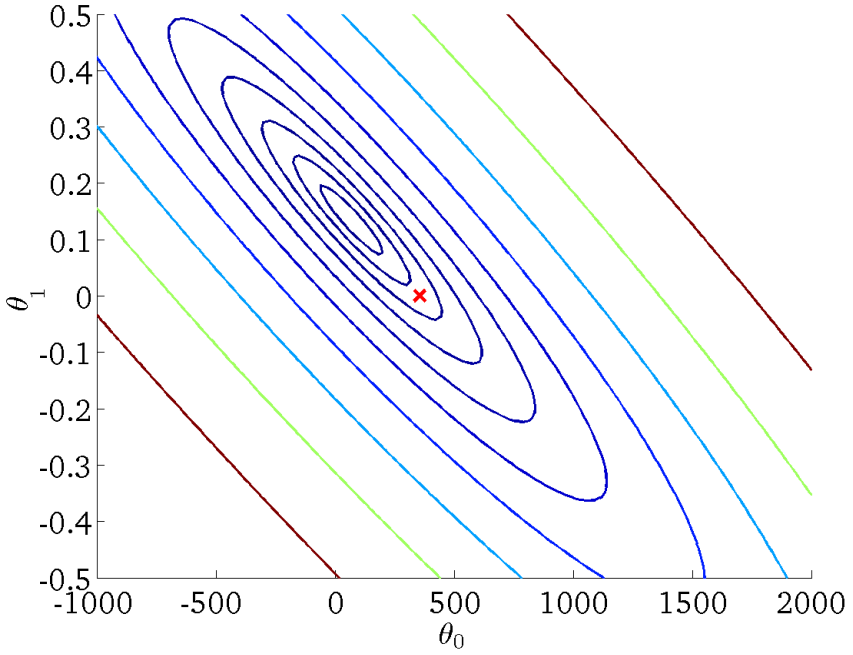
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

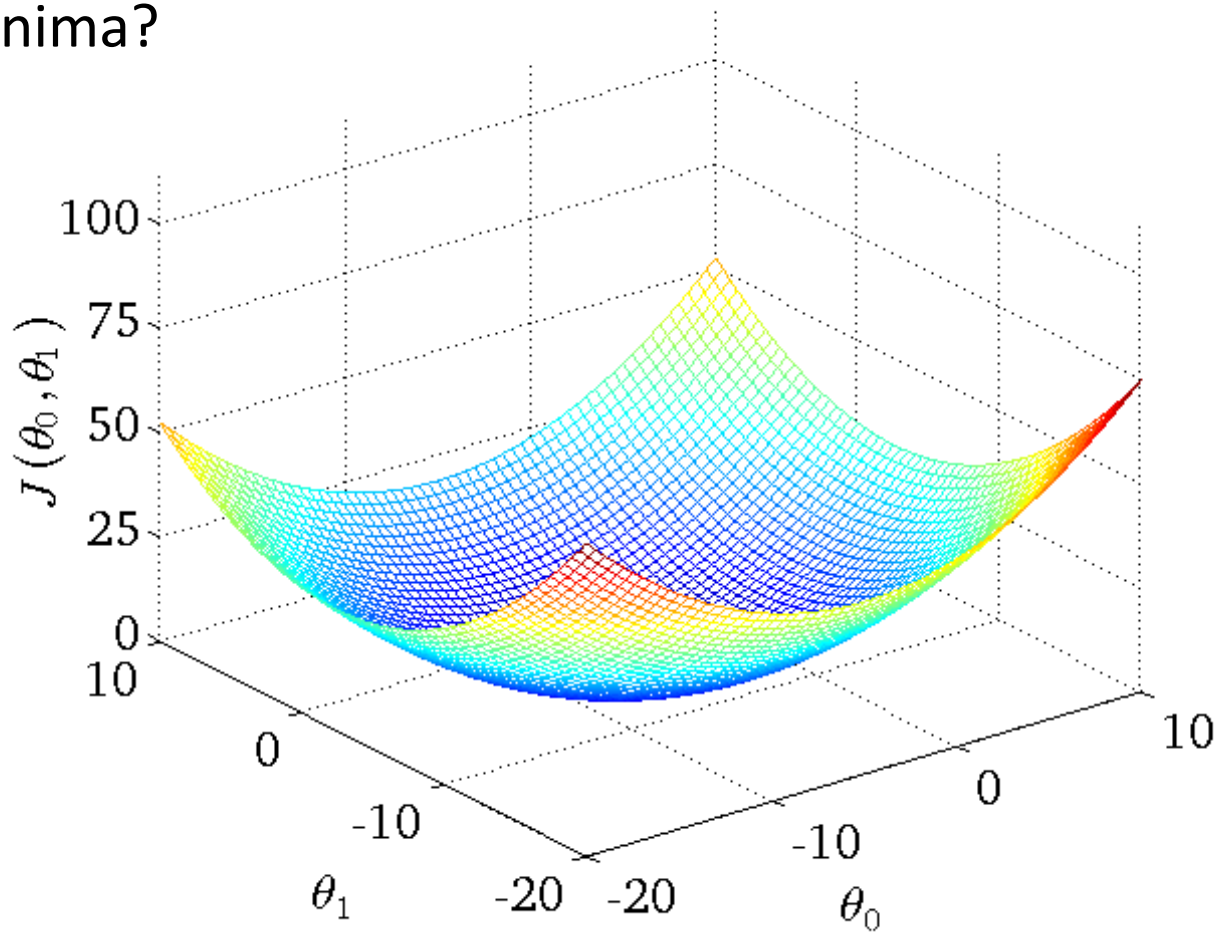
(function of the parameters θ_0, θ_1)



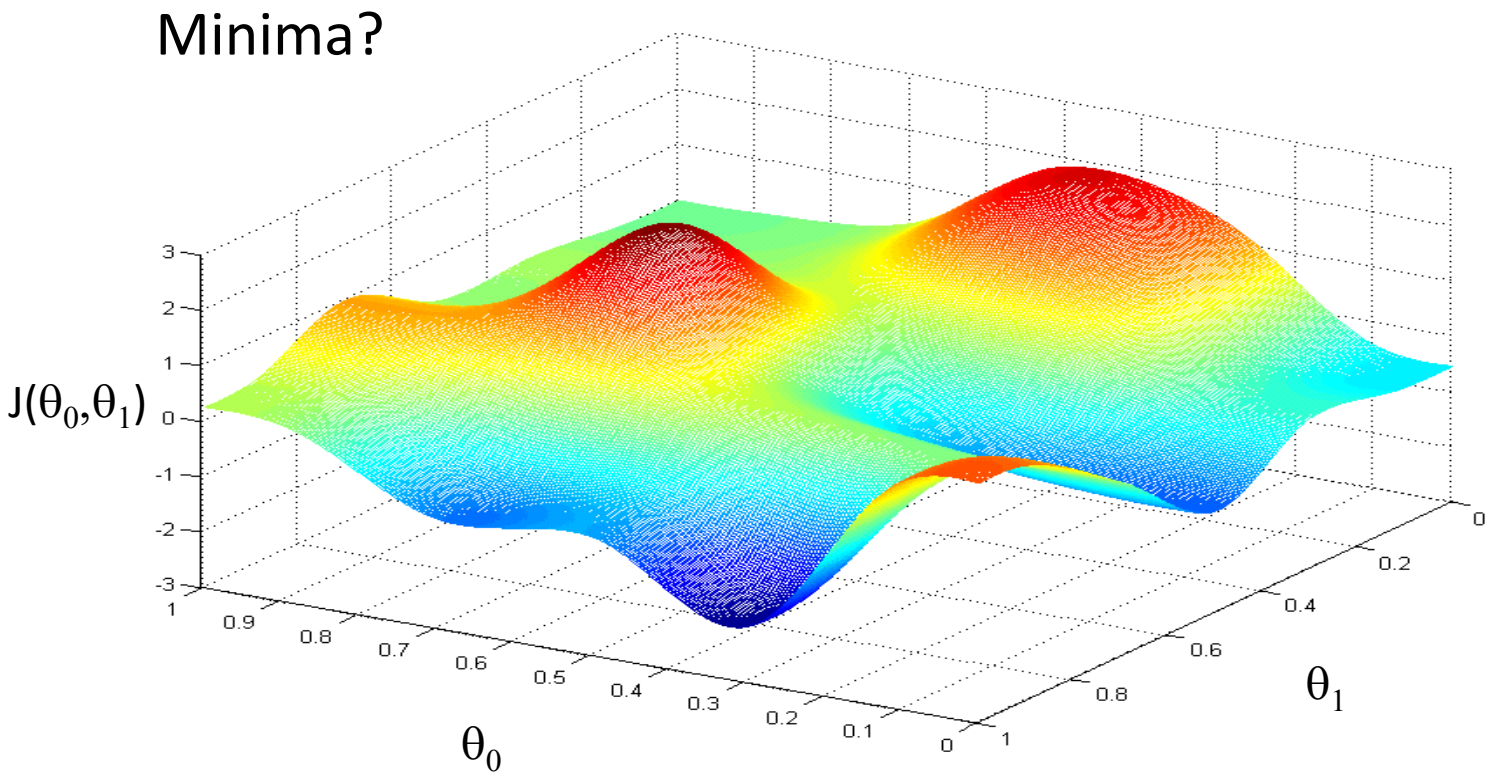
Note, squared loss cost is convex in parameters

SSD cost function is convex

Minima?



Non-convex cost function



Multivariate Linear Regression

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

For convenience of notation, define $x_0 = 1$.

θ_i 's: Parameters

Cost Function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Goal: $\underset{\theta_0, \theta_1, \ldots, \theta_n}{\text{minimize}} \; J(\theta_0, \theta_1, \ldots, \theta_n)$

Supervised Learning: Classification

Classification

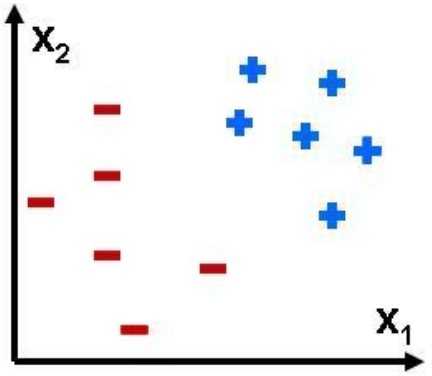
Binary Classification

$y \in \{0,1\}$

0: “Negative Class” (e.g., No COVID19)

1: “Positive Class” (e.g., COVID19)

Emotion: Smile/ No Smile
Email: Spam / Not Spam?
Video: Viral / Not Viral?
Tumor: Malignant / Benign?



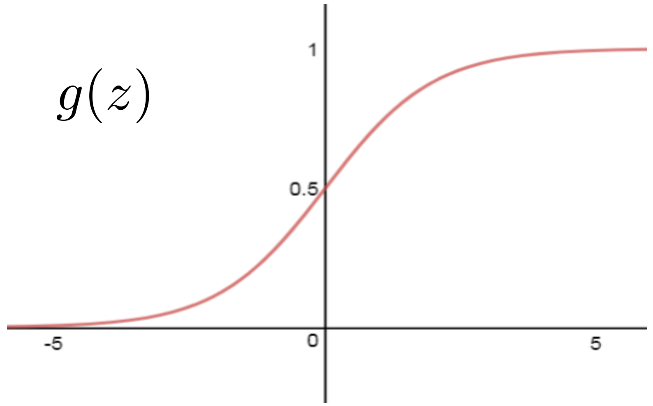
Logistic Regression

$$0 \leq h_\theta(x) \leq 1$$

map to (0, 1) with “sigmoid” function

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$



$h_\theta(x) = p(y = 1|x)$ “probability of class 1 given input”

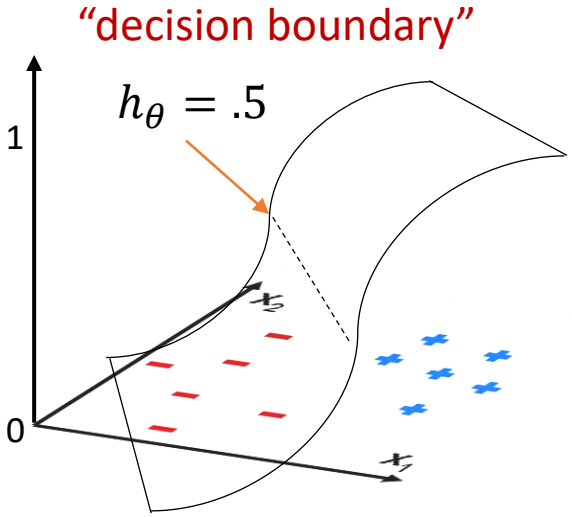
Logistic Regression

Hypothesis:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Predict "$y = 1$" if $h_\theta(x) \geq 0.5$

Predict "$y = 0$" if $h_\theta(x) < 0.5$



Logistic Regression Cost Function

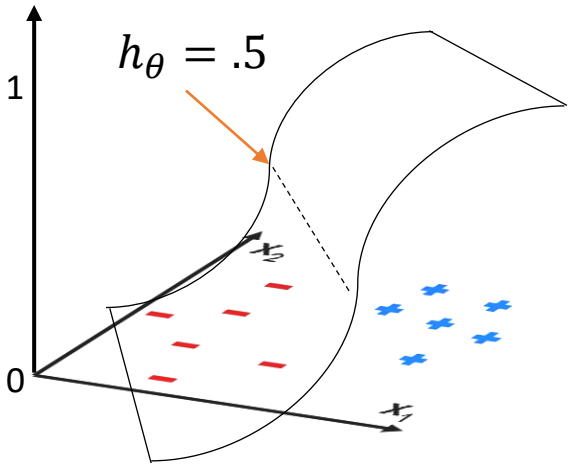
Hypothesis:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

θ: parameters

$D = \left(x^{(i)}, y^{(i)}\right)$: data

"decision boundary"



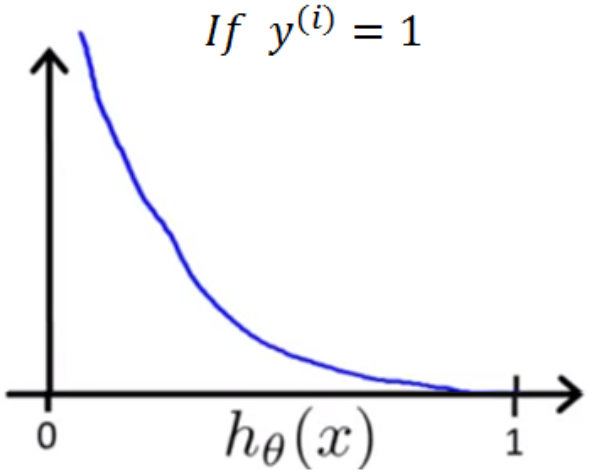
Cost Function: cross entropy

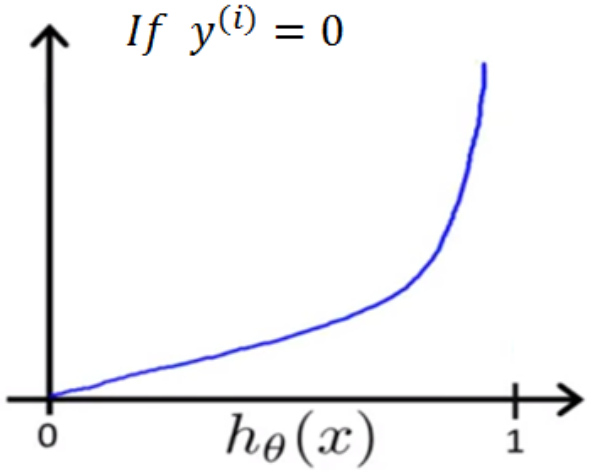
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

Goal: minimize cost $\min_\theta J(\theta)$

Logistic Regression Cost Function

$$\text{Cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(h_\theta\left(x^{(i)}\right)\right) & if\ y^{(i)} = 1 \\ -log\left(1 - h_\theta\left(x^{(i)}\right)\right) & if\ y^{(i)} = 0 \end{cases}$$





$\text{Cost} = 0\ if\ y^{(i)} = 1, h_\theta\left(x^{(i)}\right) = 1$

Similarly desirable behavior

But as $h_\theta\left(x^{(i)}\right) \to 0$

$\text{Cost} \to \infty$

[Andrew Ng's Illustration]

Cross Entropy Cost

$$\text{Cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(h_\theta\left(x^{(i)}\right)\right) & if\ y^{(i)} = 1 \\ -log\left(1 - h_\theta\left(x^{(i)}\right)\right) & if\ y^{(i)} = 0 \end{cases}$$

Can be written more compactly as:

$$\text{Cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = -\,y^{(i)} \log h_\theta(x^{(i)}) - (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)$$

Cross Entropy Cost

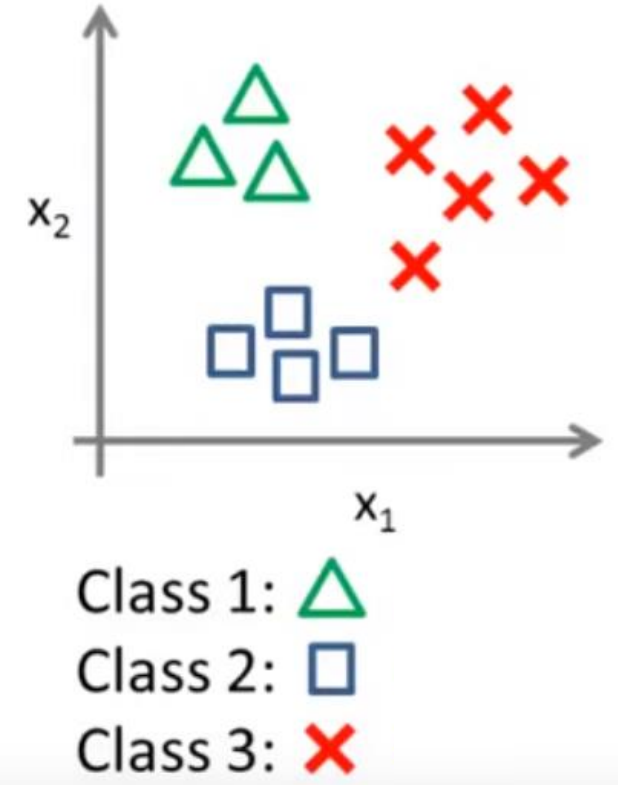
- Cross entropy compares distribution *q* to reference *p*

$$H(p,q) = -\sum_x p(x)\,\log q(x).$$

- Here q is predicted probability of $y = 1$ given x, reference distribution is $p = y^{(i)}$, which is either *1 or 0*

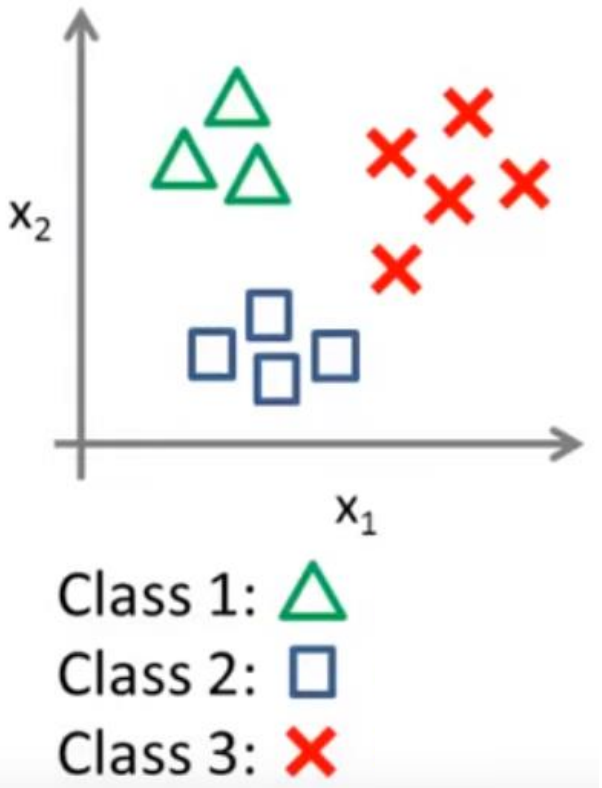
$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

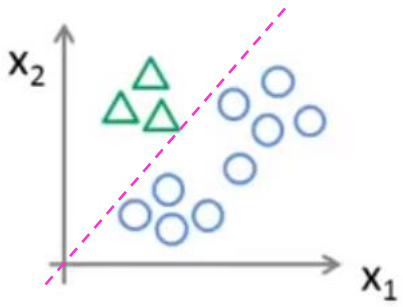
Logistic Regression for Multi-class Classification



Logistic Regression for Multi-class Classification

One-vs-all (One-vs-rest)



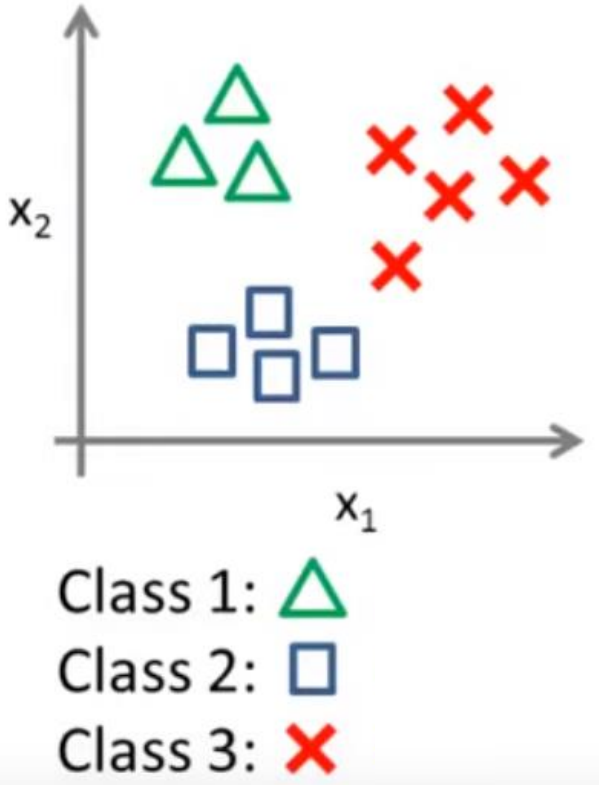


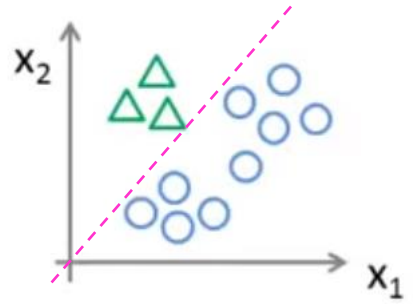
$h_\theta^1(x)$

$P(y = 1|x)$

Logistic Regression for Multi-class Classification

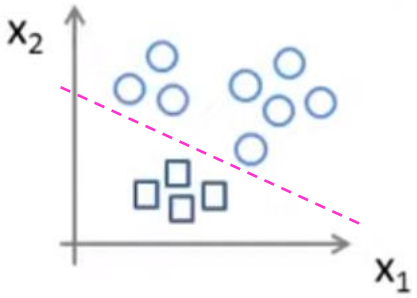
One-vs-all (One-vs-rest)





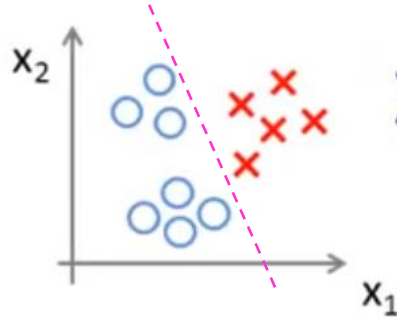
$h_\theta^1(x)$

$P(y = 1|x)$



$h_\theta^2(x)$

$P(y = 2|x)$



$h_\theta^3(x)$

$P(y = 3|x)$

Andrew Ng

Logistic Regression for Multi-class Classification

- Trained a logistic regression classifier $h_\theta^i(x)$ for each class i to predict the probability that $y = i$.

- On a new input x, to make a prediction, pick the class i that maximizes:

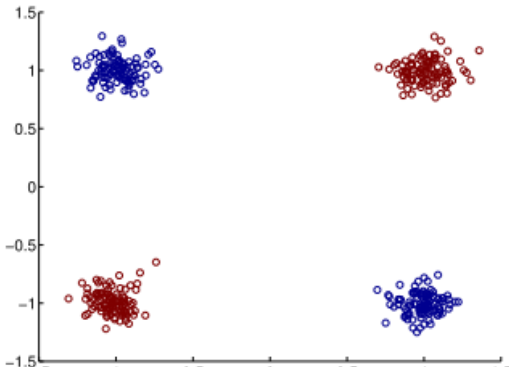
$$\max_i h_\theta^i(x)$$

Supervised Learning

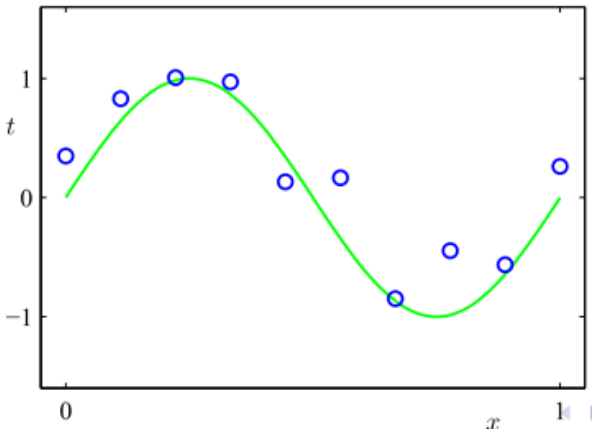
Non-linear Features

What to do if data is nonlinear?

Example of nonlinear classification



Example of nonlinear regression



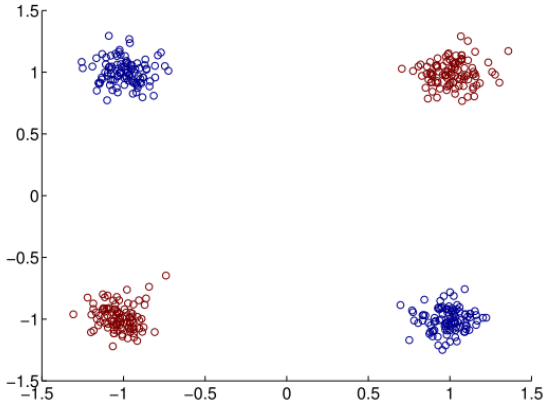
Nonlinear basis functions

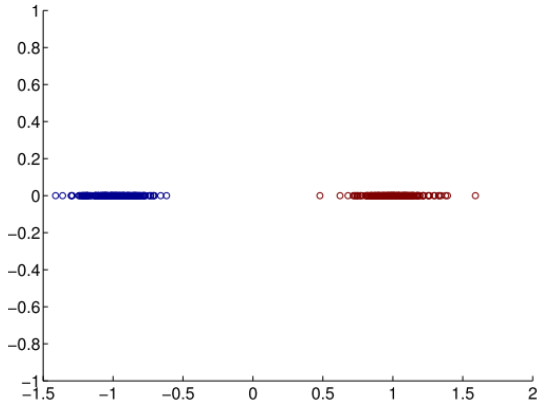
Transform the input/feature

$$\phi(x) : x \in R^2 \rightarrow z = x_1 \cdot x_2$$

How do we select such features?

Transformed training data: linearly separable!





Supervised Learning

Gradient Descent

Gradient Descent Algorithm

Set $\theta = 0$ ***or random***

Repeat {

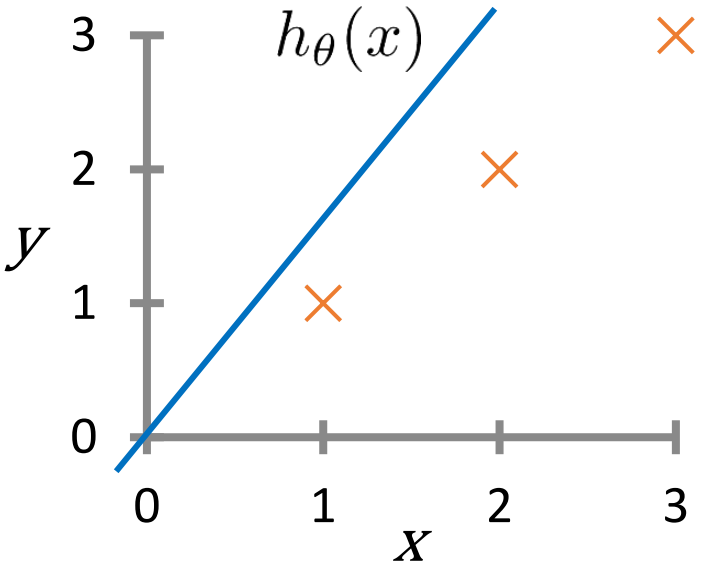
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$ simultaneously for all $j = 0, \ldots, n$

} until convergence

Gradient Descent: Intuition

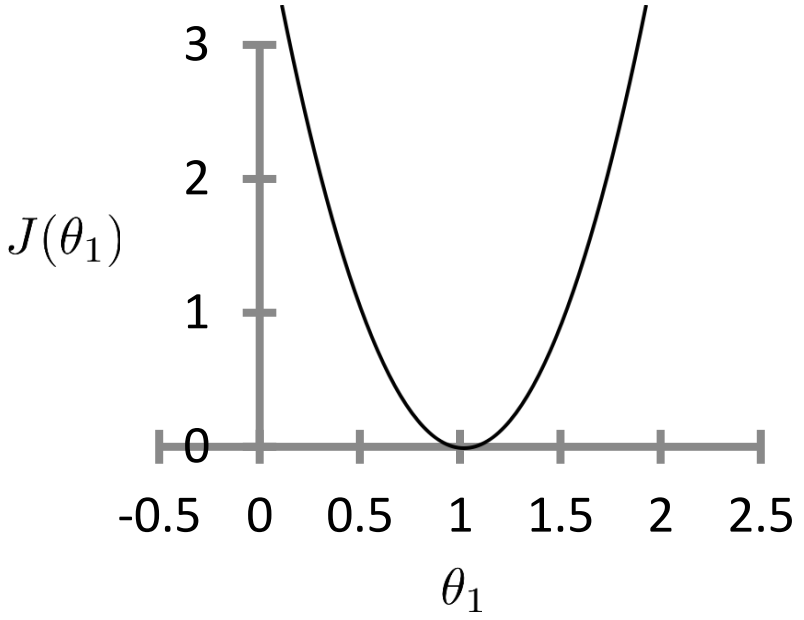
$h_\theta(x)$

(for fixed θ_1, this is a function of x)

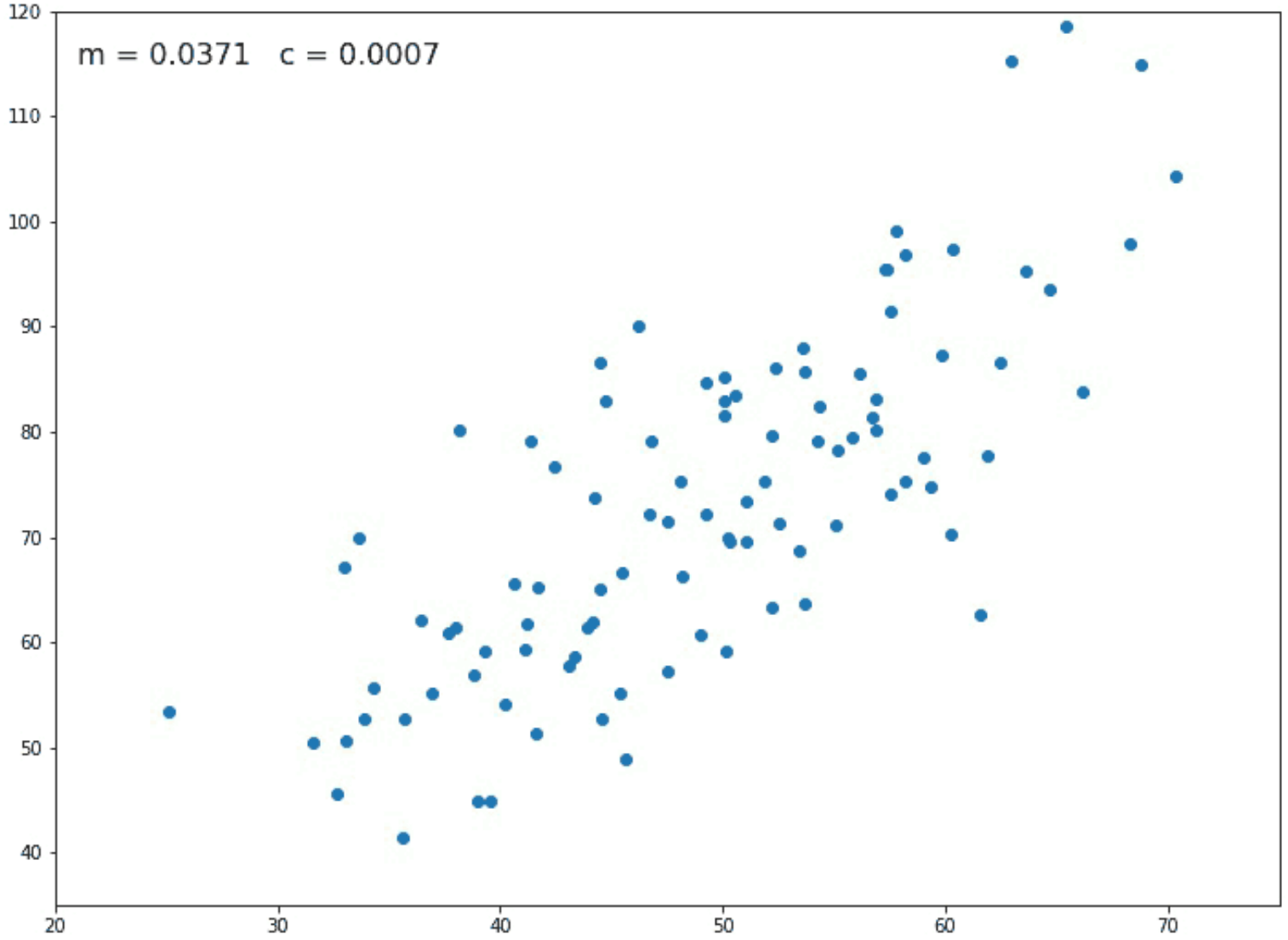


$J(\theta_1)$

(function of the parameter θ_1)



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$



Gradient descent illustration (credit: https://towardsdatascience.com/

Gradient for Least Squares Cost

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta) \quad =$$

Gradient for Least Squares Cost

For one example

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y \right)^2$$

Gradient for Least Squares Cost

For one example

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y\right)^2 \\ &= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y\right)\end{aligned}$$

Gradient for Least Squares Cost

For one example

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y \right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y \right) \\
&= \left(h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right)
\end{aligned}
$$

Gradient for Least Squares Cost

For one example

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y \right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y \right) \\
&= \left(h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y \right) x_j
\end{aligned}
$$

Gradient Descent Algorithm

Set $\boldsymbol{\theta} = 0$

Repeat {

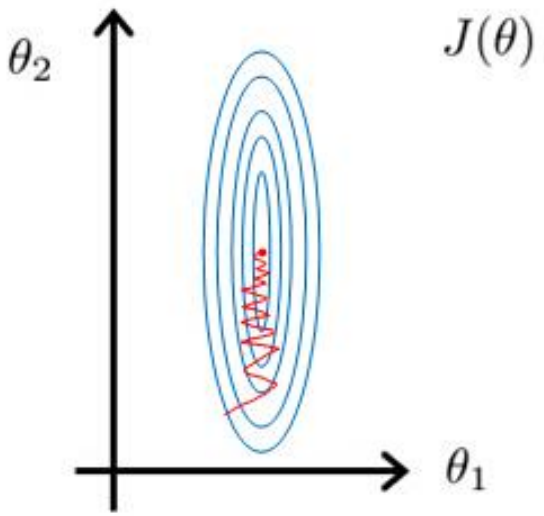
$$\boldsymbol{\theta}_j := \boldsymbol{\theta}_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) \boldsymbol{x}_j^{(i)}$$

simultaneously for all $j = 0, \ldots, n$

} until convergence

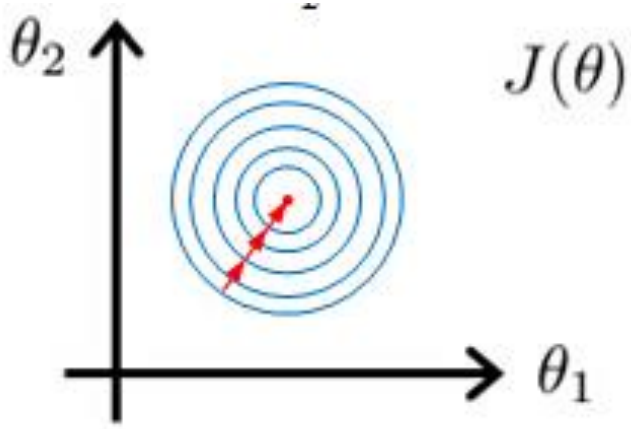
Feature normalization

- If features have very different scale, GD can get "stuck" since x_j affects size of gradient in the direction of j[th] dimension
- Normalizing features to be zero-mean (μ) and same-variance (σ) helps gradient descent converge faster



normalize

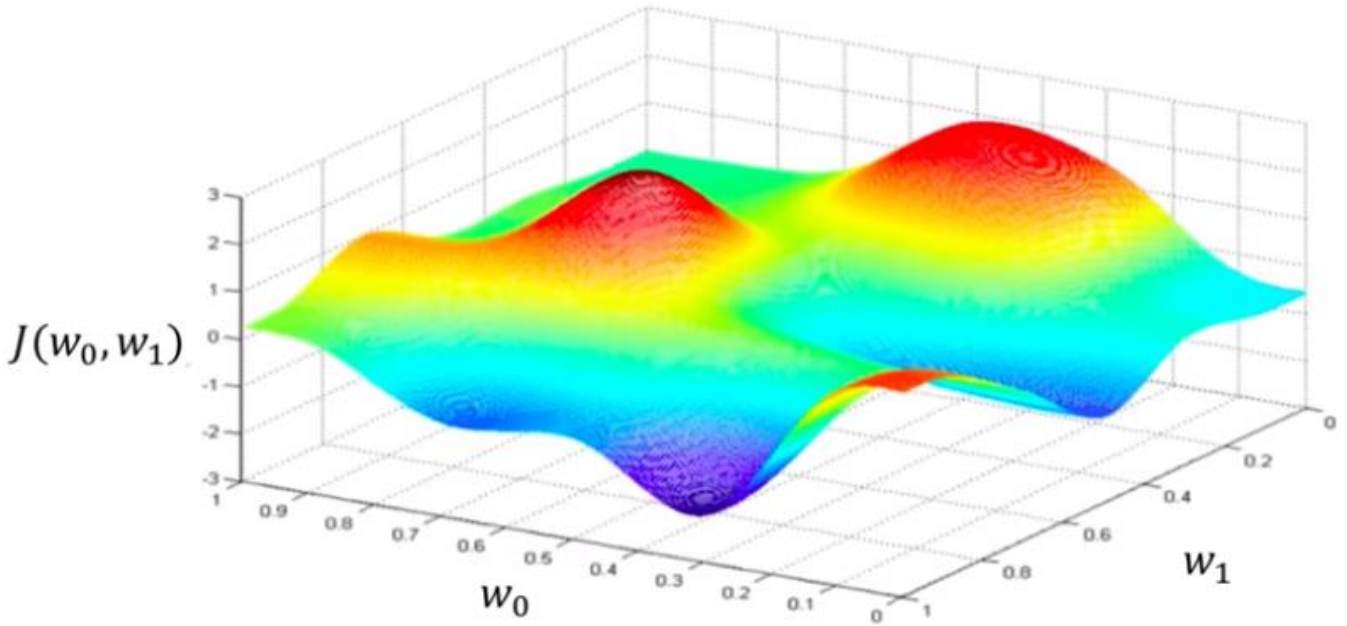
$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$



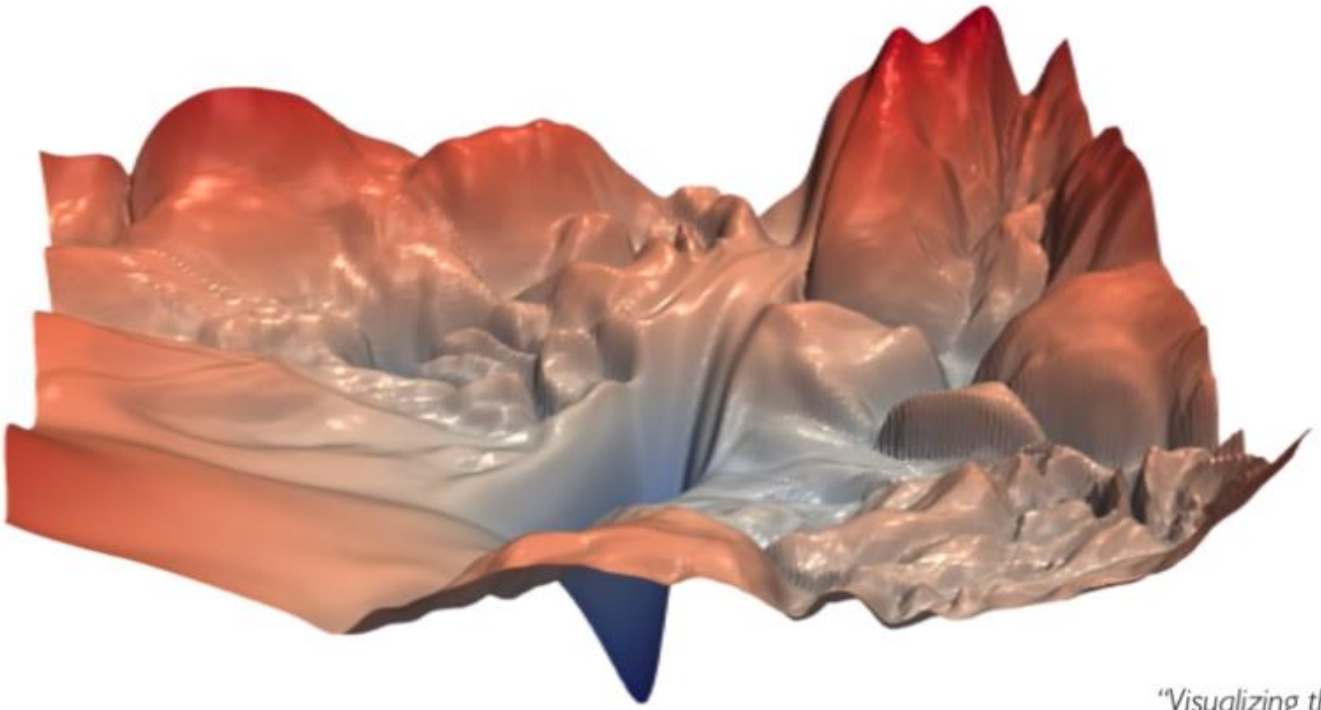
Supervised Learning

Loss Function and Learning Rate

Loss/Cost Function

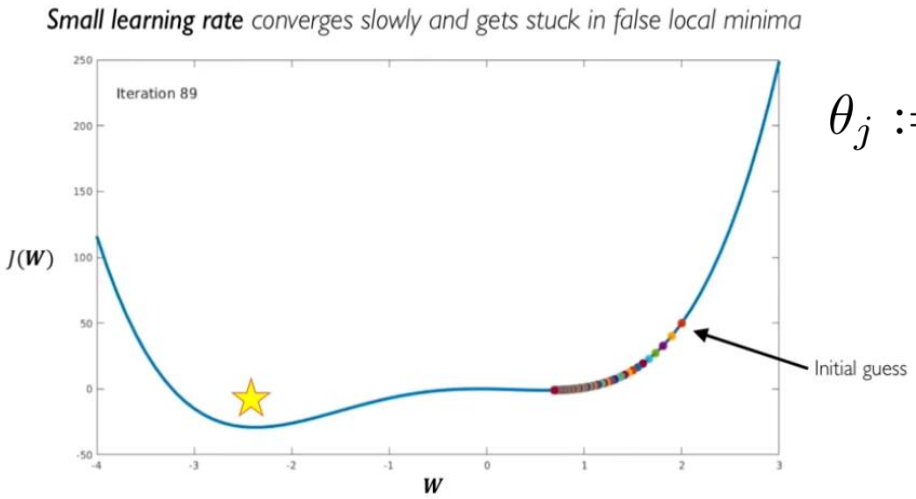


Landscape Visualization



"Visualizing the loss landscape of neural nets". Dec 2017.

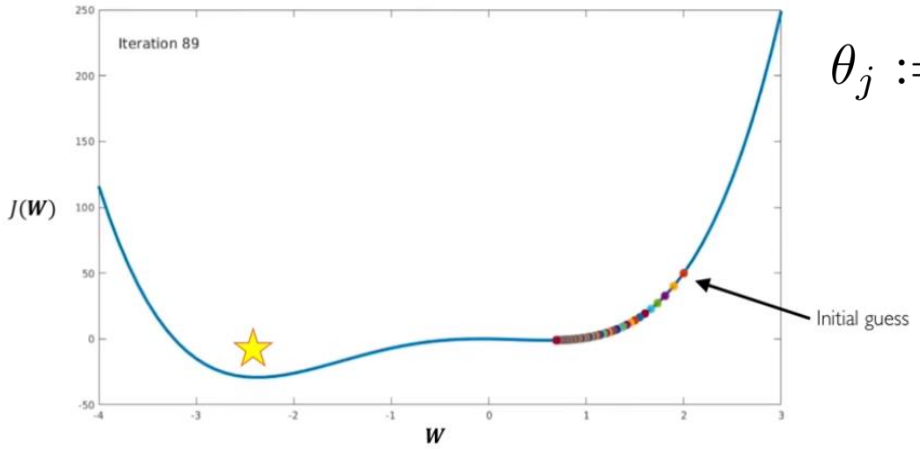
Setting the Learning Rate



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Setting the Learning Rate

Small learning rate *converges slowly and gets stuck in false local minima*



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Large learning rates *overshoot, become unstable and diverge*