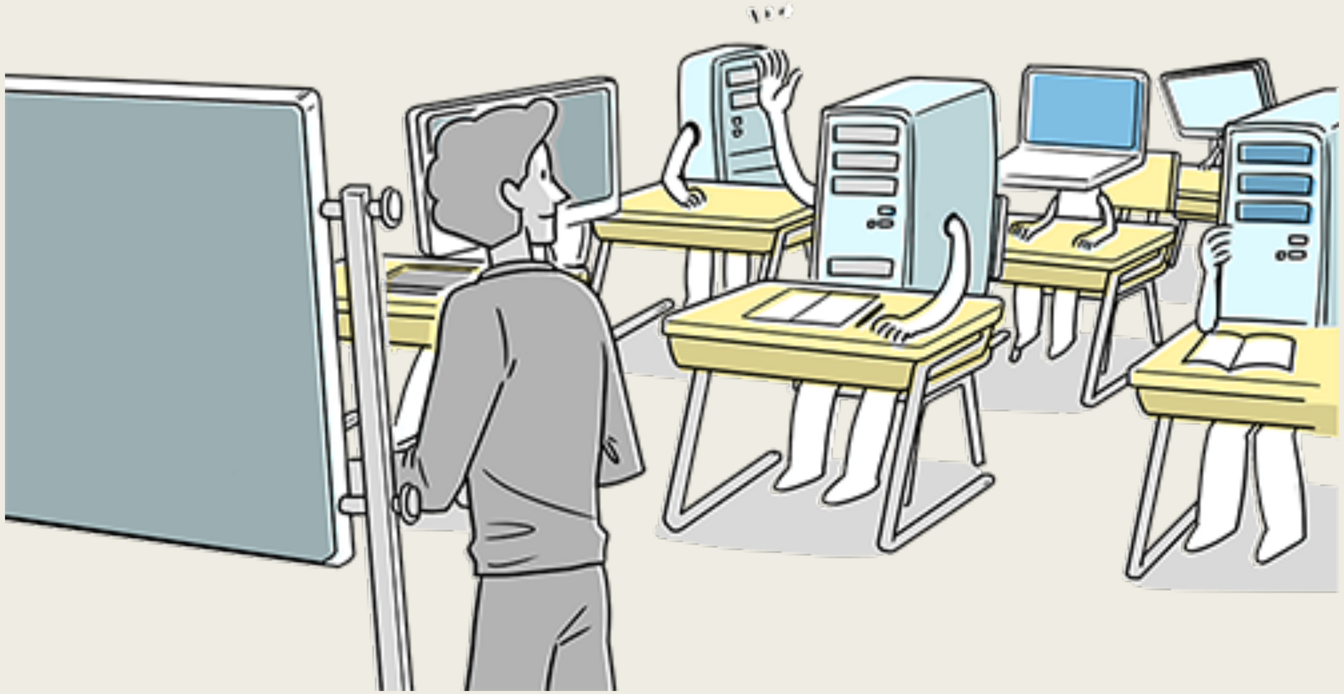


MORE ON
LINEAR REGRESSION

Recap: Linear Regression

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

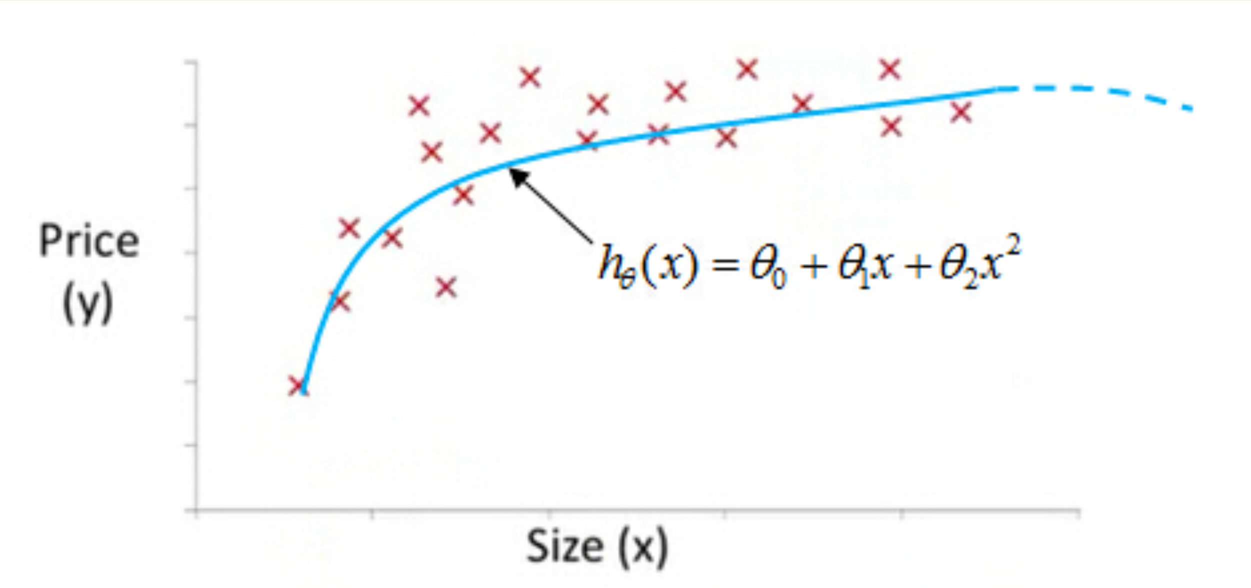
Parameters: θ_0, θ_1

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

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

Polynomial Regression

- A polynomial function may fit the data better than a linear function



<u>Set</u>

- $x_1 = x$
- $x_2 = x^2$

Model representation: $h(x) = \boldsymbol{\theta_0} + \boldsymbol{\theta_1} x_1 + \boldsymbol{\theta_1} x_2$

Polynomial Regression

Model representation: $\mathrm{h}(x) = \boldsymbol{\theta_0} + \boldsymbol{\theta_1} x_1 + \boldsymbol{\theta_1} x_2$

<u>Cost Function</u>

$$J(\boldsymbol{\theta_0}, \boldsymbol{\theta_1}, \boldsymbol{\theta_2}) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}\left(x^i\right) - y^i\right)^2$$

<u>Gradient Descent (Parameters Learning)</u>

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

...

}

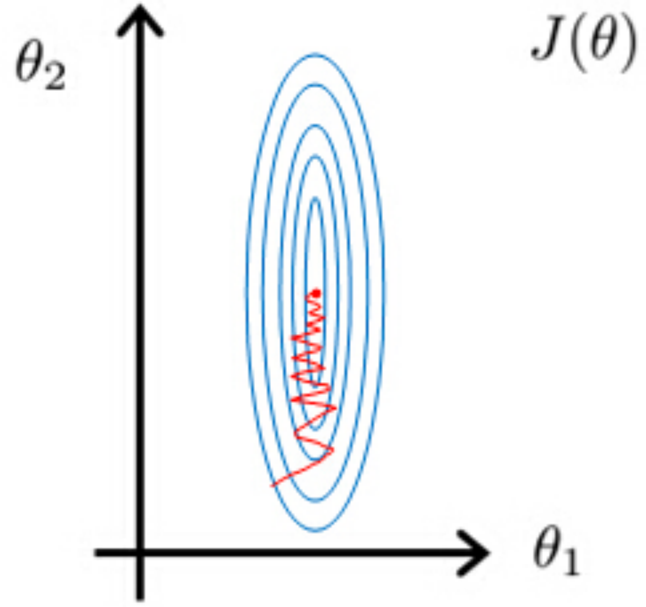
Feature Scaling

Idea: make sure features are on a similar scale

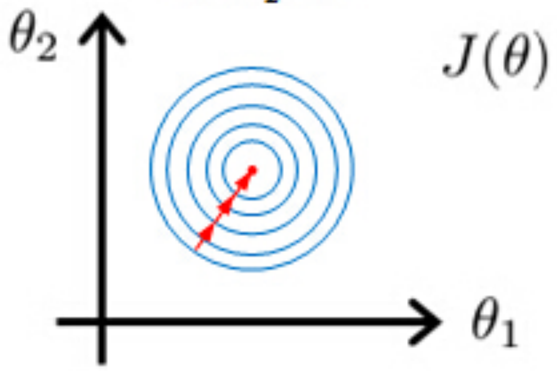
x_1 = size (0-2000 feet^2)
x_2 = number of bedrooms (1-5)

$$x_1 = \frac{\text{size(feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$



$$0 \le x_1 \le 1$$
$$0 \le x_2 \le 1$$



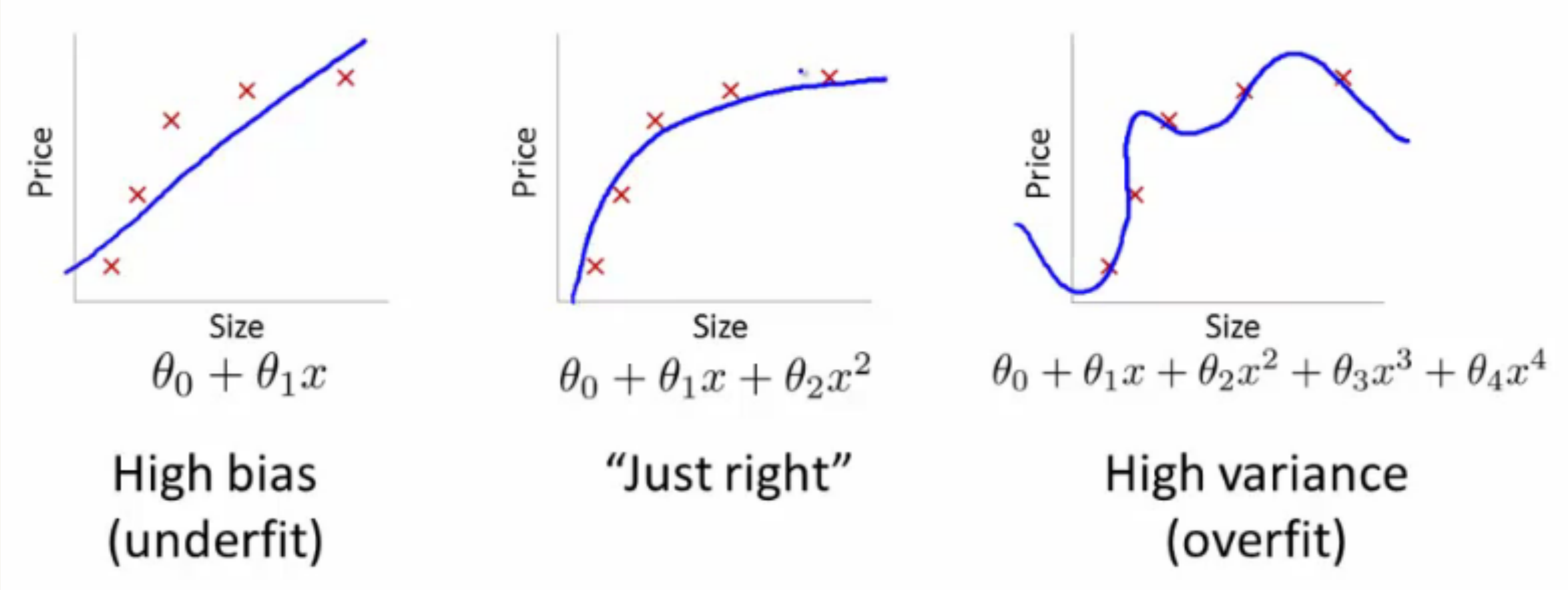
Feature Scaling

<u>Mean Normalization</u>

$$x_i \leftarrow \frac{x_i - mean}{std}$$

So all features would have an average of zero and similar range

The problem of overfitting



Addressing overfitting

1) **Reduce number of features**

- Manually select which features to keep
- Model selection algorithms are discussed later (good for reducing number of features)
- But, in reducing the number of features we lose some information

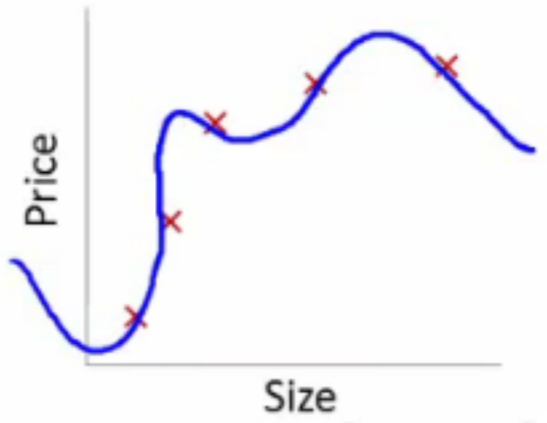
2) **Regularization**

- Keep all features, but reduce magnitude of parameters $\boldsymbol{\theta}$
- Works well when we have a lot of features, each of which contributes a bit to predicting y

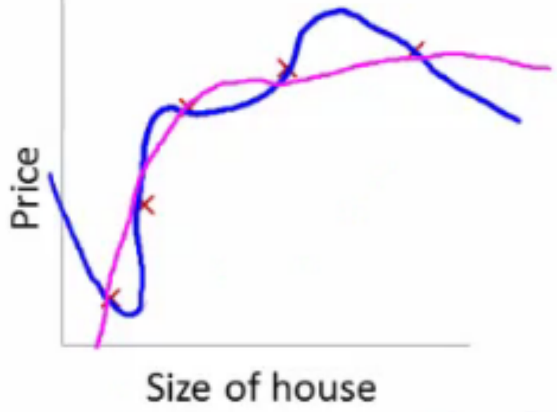
Regularization

Intuition: making theta so small that is almost equivalent to zero

Penalize and make some of the **θ** parameters really smalle.g. here $\boldsymbol{\theta_3}$ and $\boldsymbol{\theta_4}$



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



$$\underline{\theta_0 + \theta_1 x + \theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

Ridge regression (L2 Regularization)

Can't be ZERO

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{\boldsymbol{\theta}}\left(x^{i}\right) - y^{i}\right)^{2} + \boldsymbol{\lambda}\sum_{j=1}^{n}\boldsymbol{\theta}_j^2$$

- Small values for parameters corresponds to a simpler hypothesis (you effectively get rid of some of the terms)

- A simpler hypothesis is less prone to overfitting

- λ is the **regularization parameter**

  Controls a trade off between our two goals

  1. Want to fit the training set well
  2. Want to keep parameters small

This is what we call "Hyper-parameters"

Lasso regression (L1 Regularization)

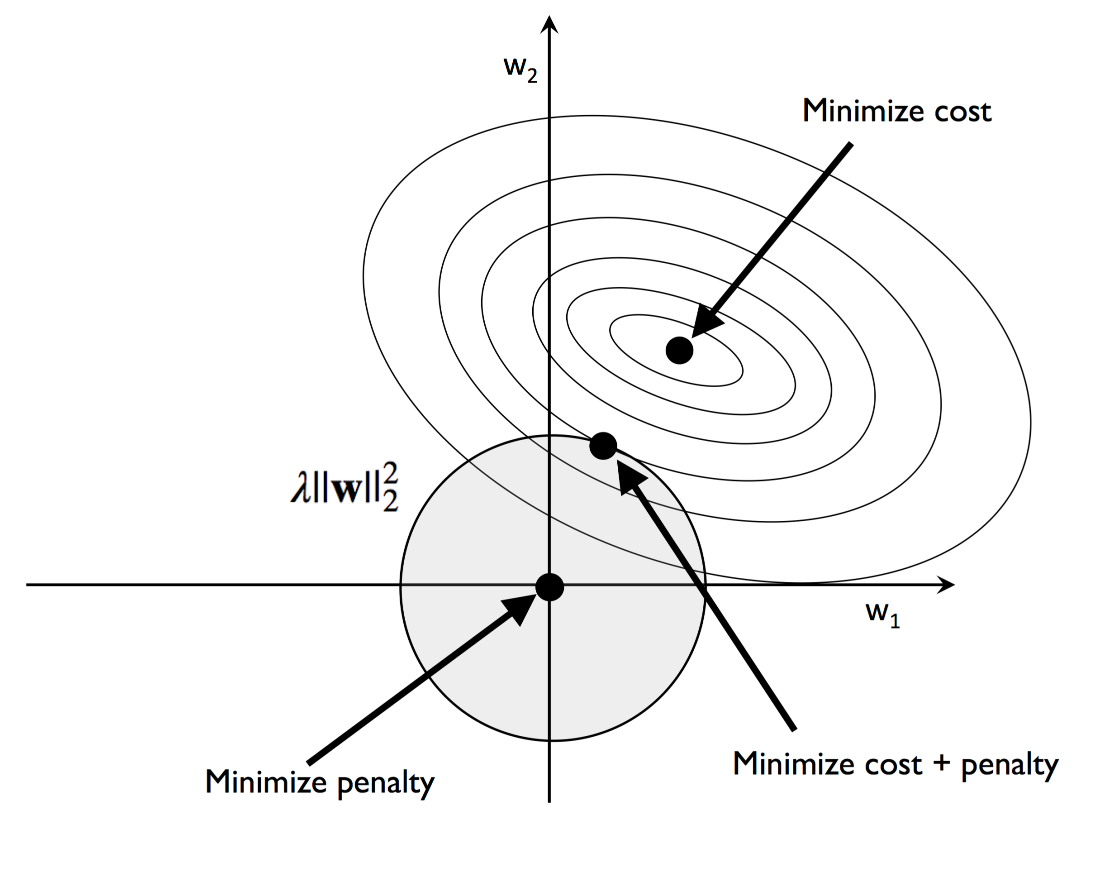
Can be ZERO!

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{\boldsymbol{\theta}}\left(x^i\right) - y^i\right)^2 + \lambda\sum_{j=1}^{n}\left|\boldsymbol{\theta}_j\right|$$

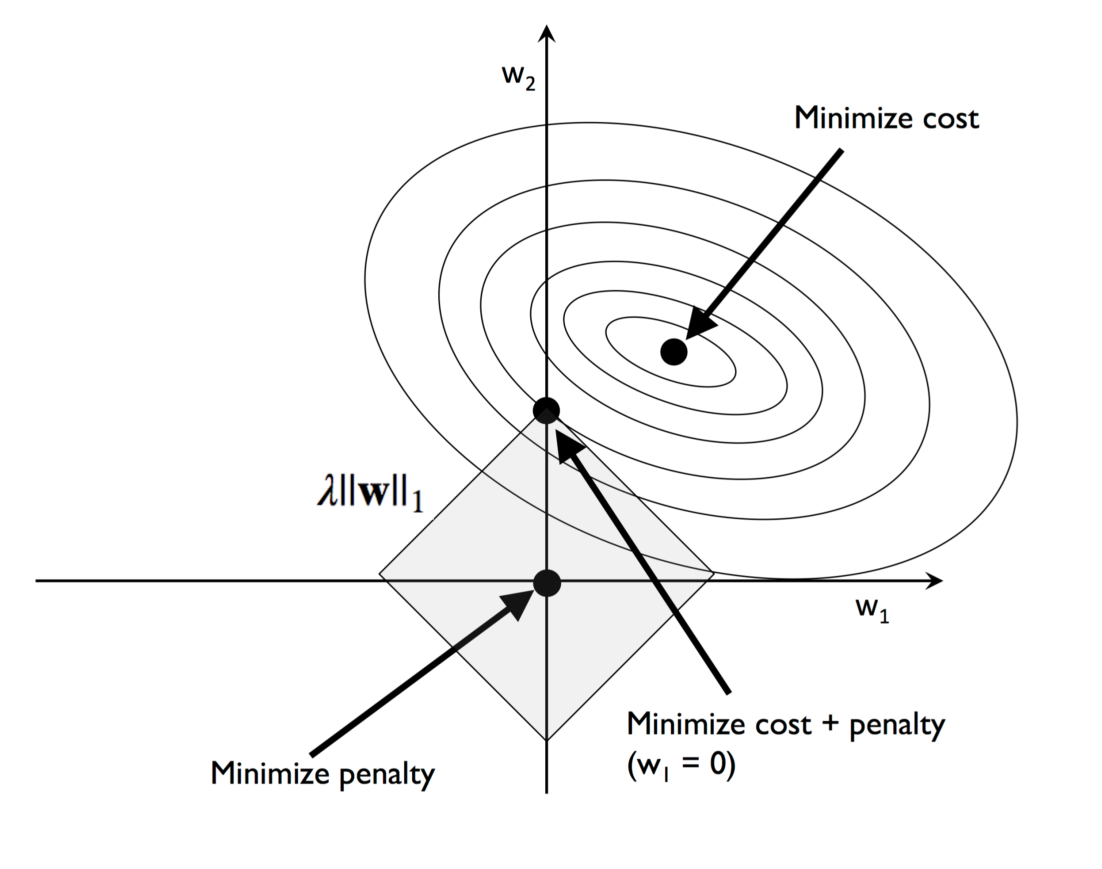
- Very similar to L2 regularization.
- However, instead of a quadratic penalty term as in L2, we penalize the model by the absolute weight coefficients
- This is a subtle, but important change. Some of the coefficients may be shrunk exactly to zero.
- Perform both “Feature selection” and “Regularization”

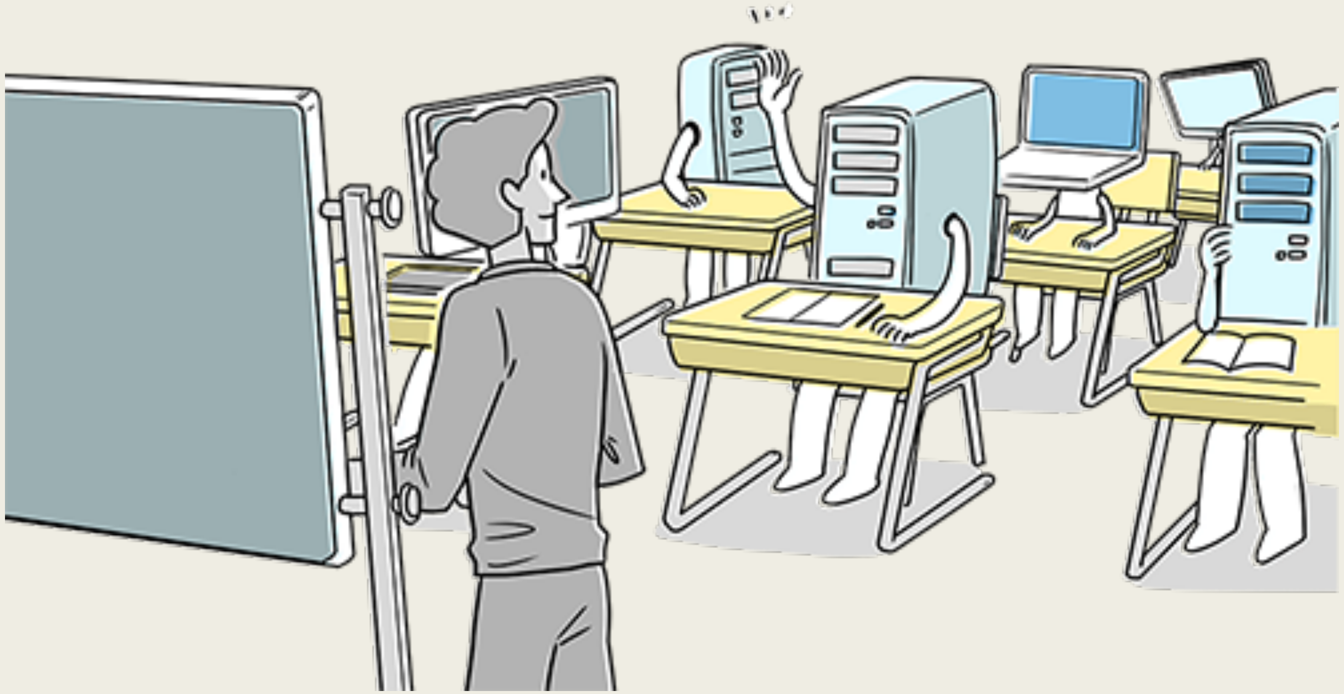
Ridge (L2) vs Lasso (L1)

Ridge



Lasso





CLASSIFICATION

Classification

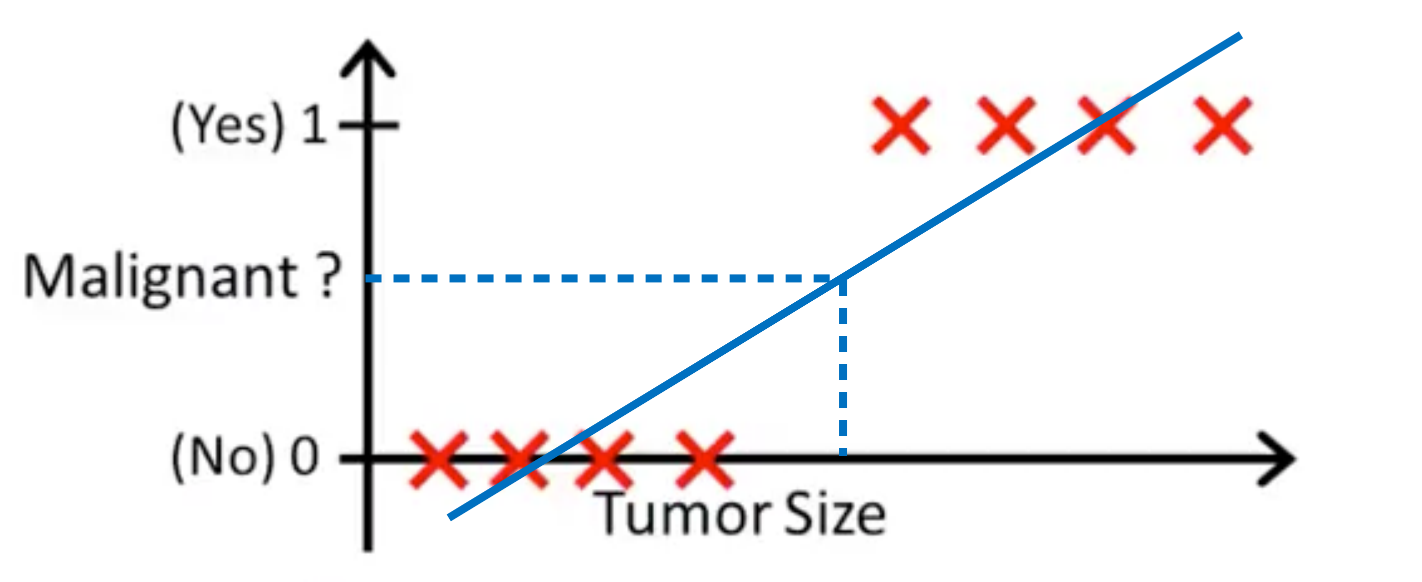
- Email: Spam/ Not Spam ?
- Online Transactions: Fraudulent (Yes/No) ?
- Tumor: Malignant/ Benign?

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

0: “Negative Class” (e.g., benign tumor)

1: “Positive Class” (e.g., malignant tumor)

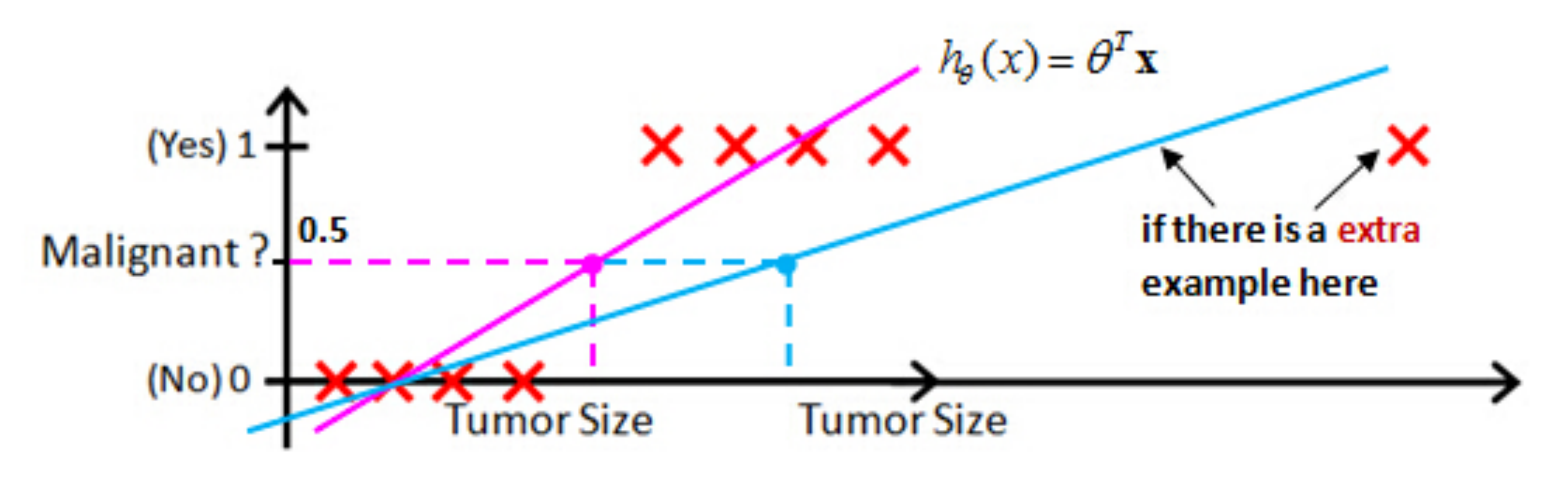
How to develop a classification algorithm?



Can we use linear regression ? $h(x) = \boldsymbol{\theta_0} + \boldsymbol{\theta_1} x_1$

If $h(x) \geq 0.5$, predict "y = 1"
If $h(x) < 0.5$, predict "y = 0"

How to develop a classification algorithm?



Linear regression to a classification problem often isn't a great idea

Classification**:** y = 1 or 0

But for linear regression: $\mathrm{h}(x)$ can be > 1 or < 0

We want $0 \leq \mathrm{h}(x) \leq 1$

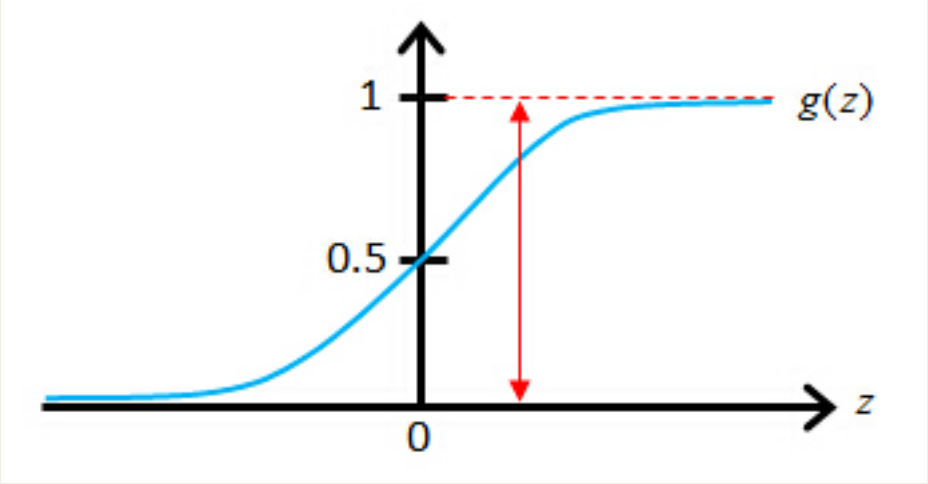
Logistic Regression Model

We want: $\mathbf{0} \leq \mathbf{h}(\boldsymbol{x}) \leq \mathbf{1}$

$$\mathbf{h}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{\theta}_0 + \boldsymbol{\theta}_1 \mathbf{x}_1)$$

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

$$\mathbf{h}(\boldsymbol{x}) = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-(\boldsymbol{\theta}_0 + \boldsymbol{\theta}_1 \mathbf{x}_1)}}$$



The sigmoid/logistic function g(z)

Interpretation of Hypothesis Output

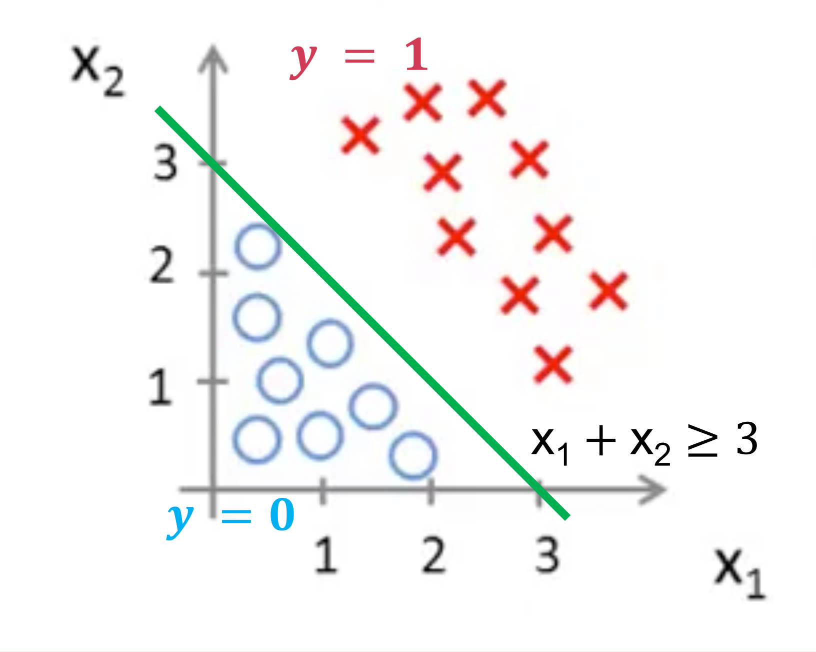
$$\mathbf{h}(\boldsymbol{x}) = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-(\boldsymbol{\theta}_{\mathbf{0}} + \boldsymbol{\theta}_{\mathbf{1}} \mathbf{x}_{\mathbf{1}})}}$$

$h(x)$ = estimated probability that y = 1 on input *x*

$h(x) = \mathrm{P}(\mathrm{y} = 1 | \mathrm{x})$

e.g. P(email is spam | number of word “discount” in the email)

Decision Boundary



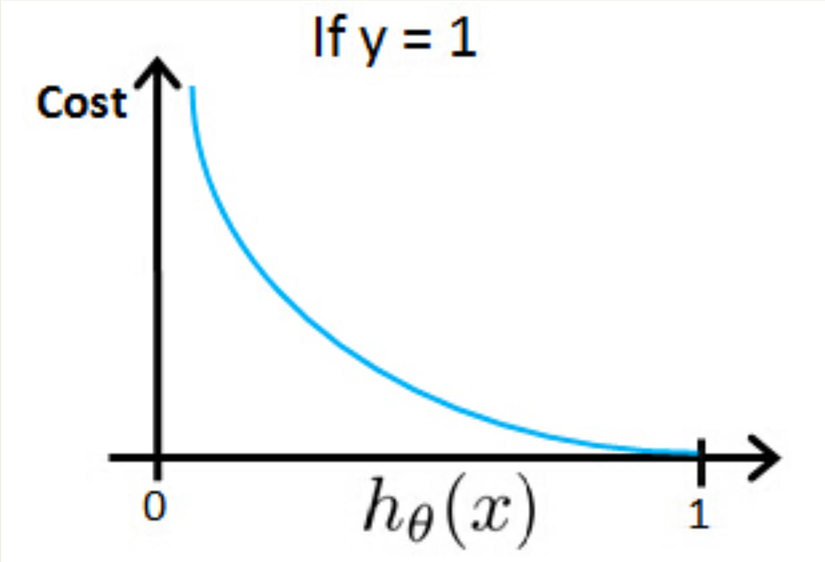
$$h(x) = \frac{1}{1 + e^{-(-3 + 1x_1 + 1x_2)}}$$

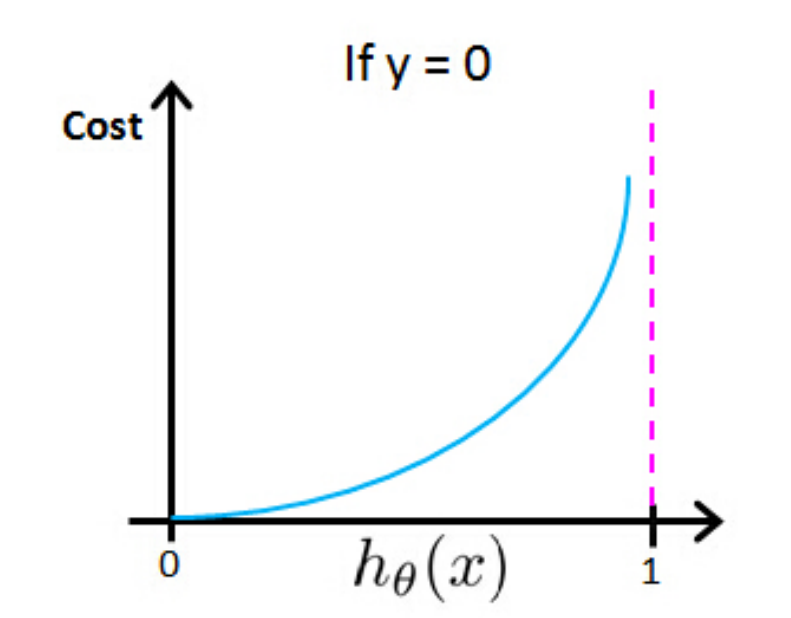
Predict $y = 1$ if $h(x) \geq 0.5$

equivilent to $-3 + x_1 + x_2 \geq 0$

Logistic Regression Cost function

$$J(\theta) = \mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





Parameter Learning with Gradient Descent

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

Want $\min_\theta J(\theta)$:

Algorithm looks identical to linear regression!!

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

}