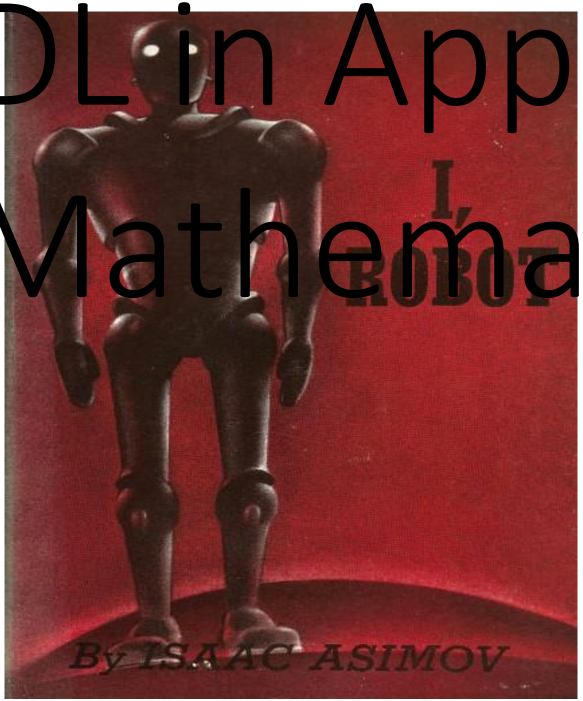
DL in Applied Mathematics



Deep Learning

Linear models

Lecture 2: Calculating Network Error with

Loss.

Cross-entropy loss.

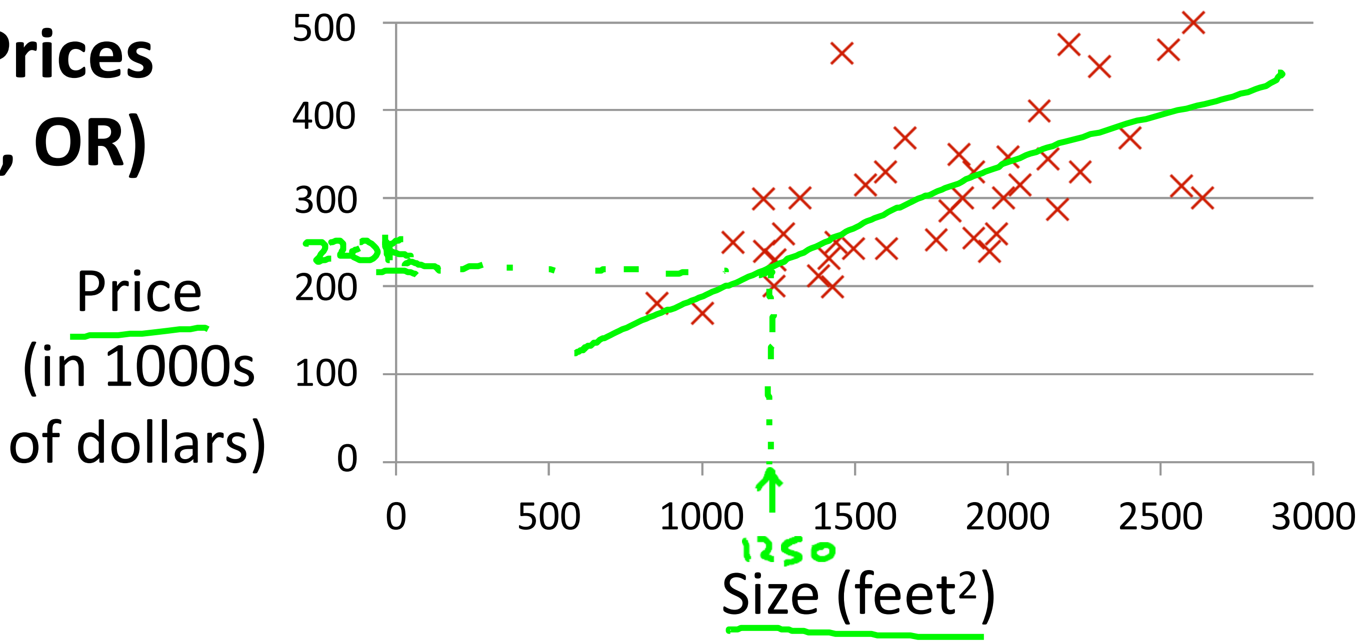
Definitions

- The **loss function** is a measure of how well a single data point was predicted by the model. It quantifies the difference between the predicted value and the actual value for that data point.

- It's typically used within the context of a specific training example. For instance, **in a regression problem**, a common loss function is the mean squared error (MSE), where the loss for each individual data point is the square of the difference between the predicted value and the actual value.

- **In classification problems**, other types of loss functions like cross-entropy loss are used.

Housing Prices (Portland, OR)



<u>Supervised Learning</u>

Given the “right answer” for each example in the data.

<u>Regression Problem</u>

Predict real-valued output

Classification: Discrete-valued output

Training set of housing prices (Portland, OR)

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

$m = 47$

Notation:

- **m** = Number of training examples
- **x**'s = "input" variable / features
- **y**'s = "output" variable / "target" variable

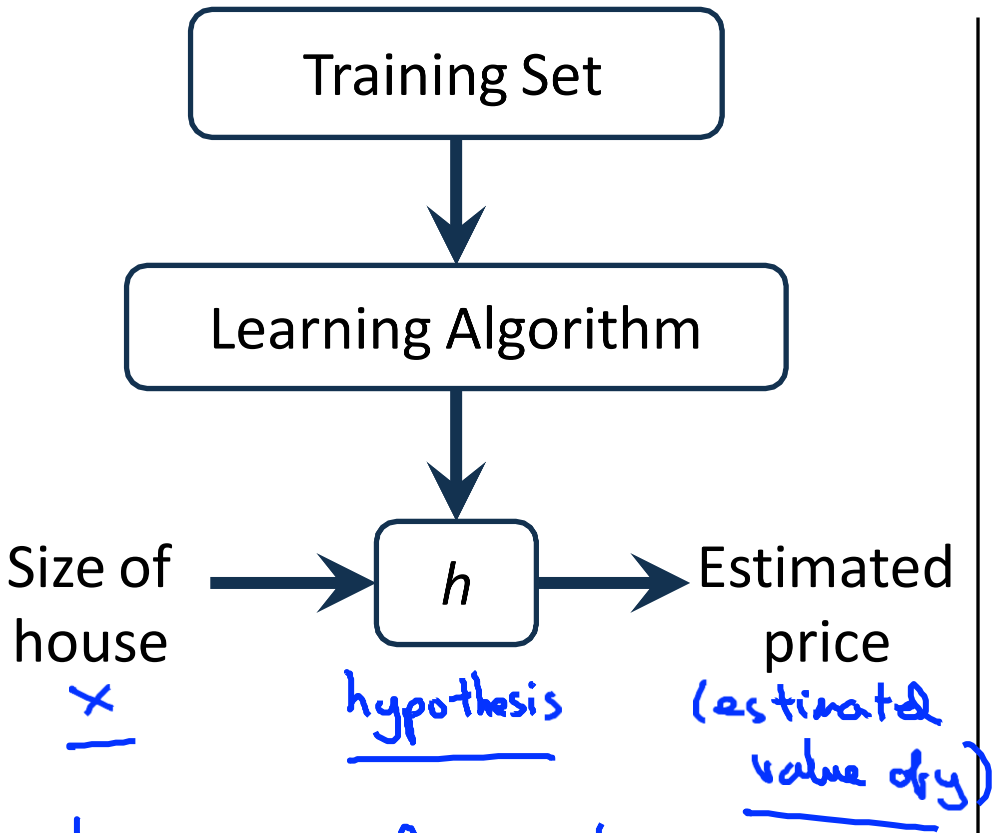
(x, y) – one training example

$(x^{(i)}, y^{(i)})$ – i^{th} training example

$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

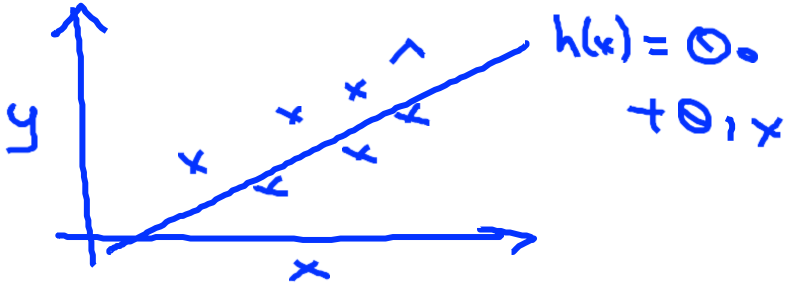


h maps from x's to y's.

How do we represent *h* ?

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

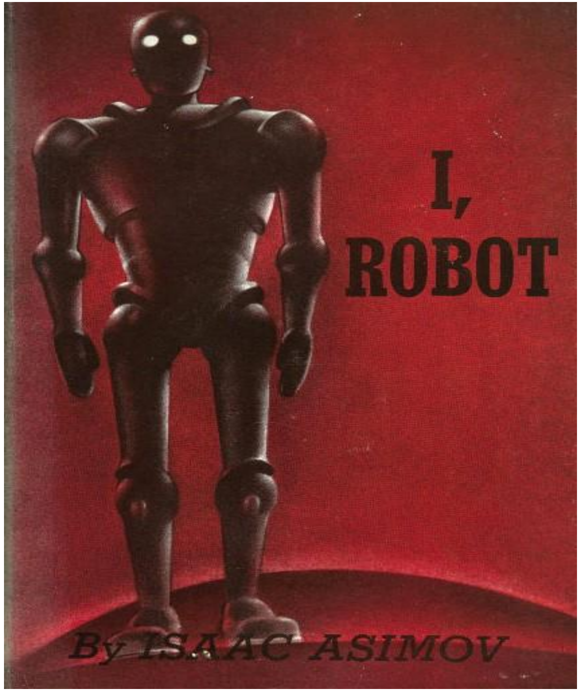
Shorthand: $h(x)$



Linear regression with one variable. (x)

Univariate linear regression.

└ one variable



Deep Learning

Linear regression with one variable

Cost function

Definations

Difference of Loss function from Cost Function

The terms "loss function" and "cost function" are often used interchangeably in the context of machine learning and statistics, but they can have slightly different meanings depending on the context:

The cost function, sometimes referred to as the "objective function" or "error function", is a broader concept. It's the average of the loss functions across all data points in the training dataset.

Essentially, the cost function represents the total error of the model over the entire training dataset. It's what the training algorithm (like gradient descent) seeks to minimize.

The cost function is more about the performance of the model as a whole, rather than individual predictions.

In practice, the distinction between these two terms is often blurred, and they are sometimes used interchangeably, especially in the context of optimization during training. In this process, the goal is to adjust the parameters of the model to minimize the overall cost, which involves reducing the loss for each individual training example.

To summarize, while the loss function pertains to errors in individual predictions, the cost function aggregates these errors across the entire dataset to guide the overall training of the model.

Training Set

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

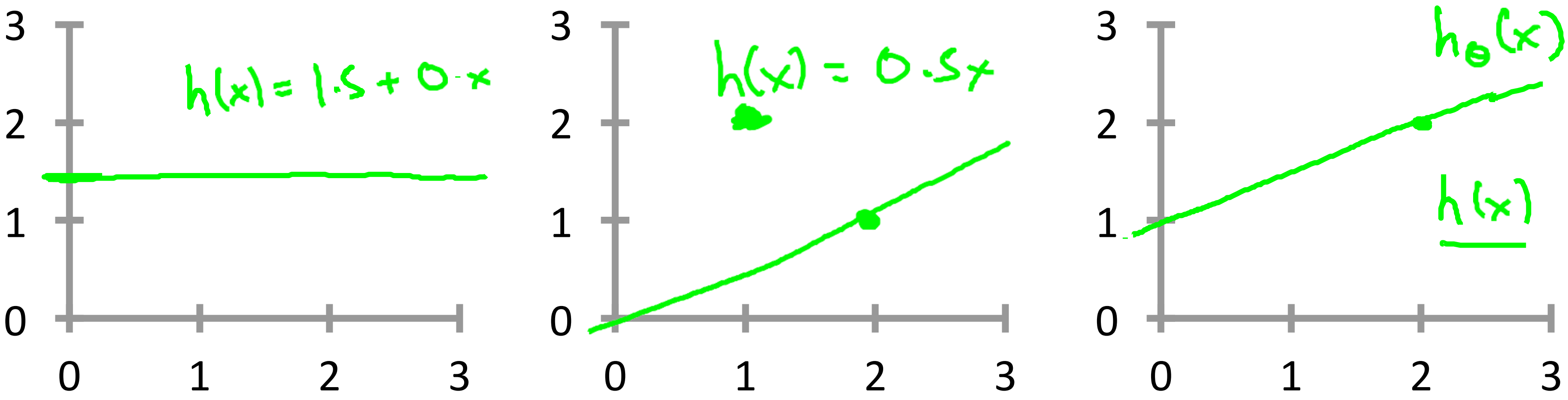
$m = 47$

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

θ_i's: Parameters

How to choose θ_i's ?

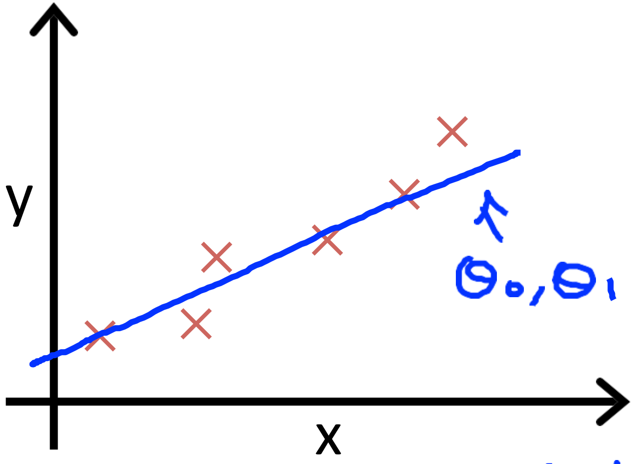
$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$

$\theta_0 = 0$
$\theta_1 = 0.5$

$\theta_0 = 1$
$\theta_1 = 0.5$



$(x^{(i)}, y^{(i)})$

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

x, y

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

m: # training examples

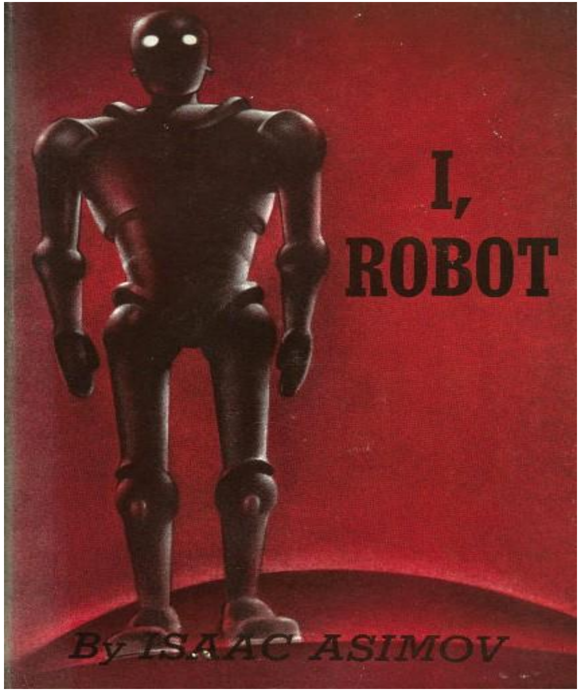
$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

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

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

Cost function

Squared error function



Deep Learning

Linear regression with one variable

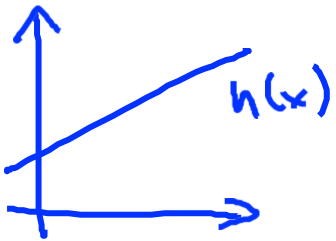
Cost function intuition I

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

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

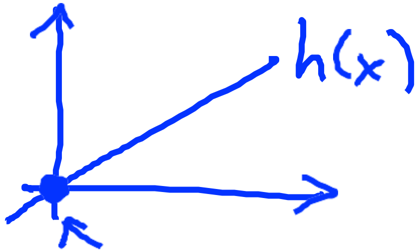
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Simplified

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

$\theta_0 = 0$

$$\theta_1$$



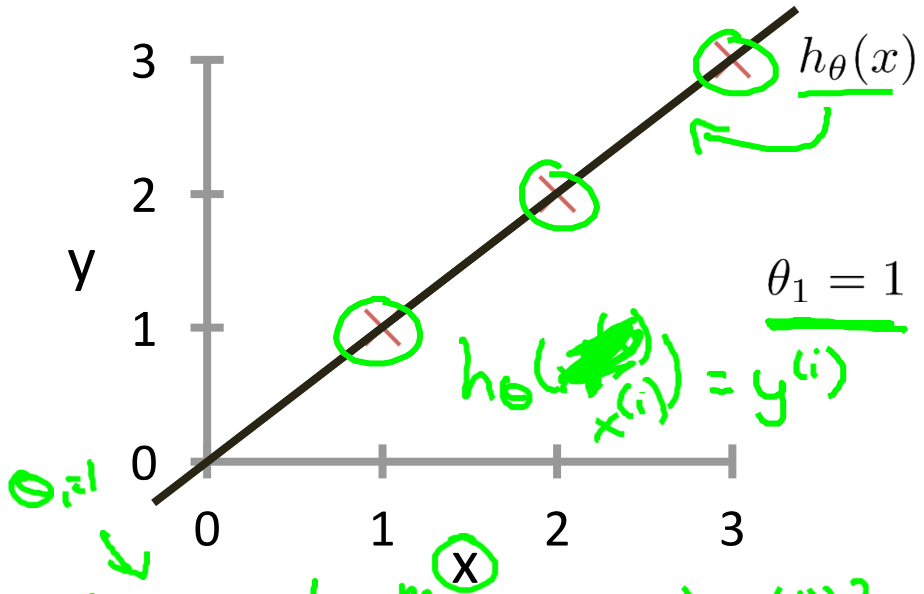
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$\theta, x^{(i)}$

$$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$$

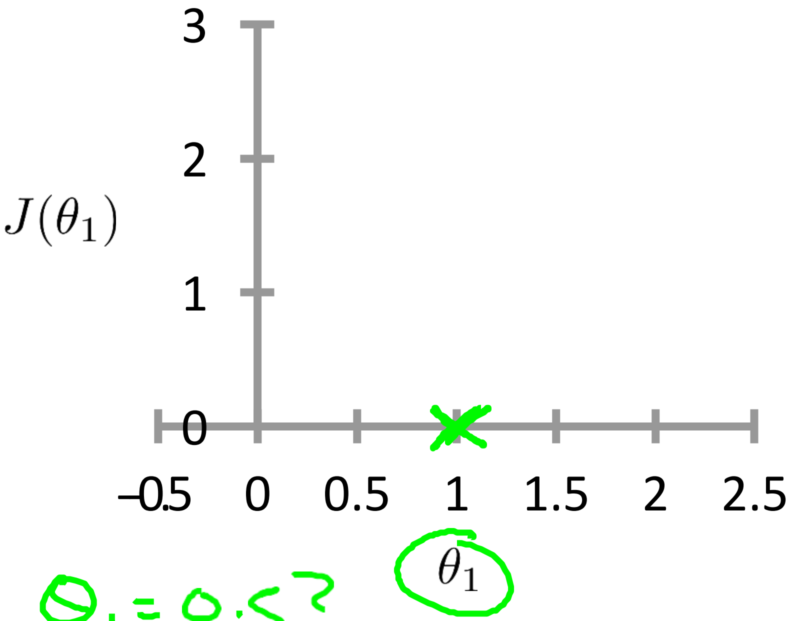
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



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

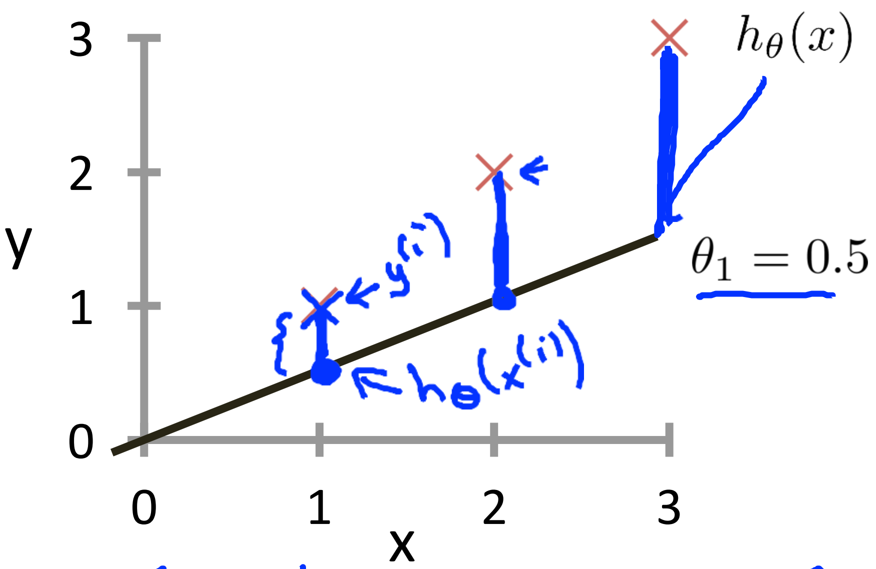
$$= \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 x^{(i)} - y^{(i)}\right)^2 = \frac{1}{2m}\left(0^2 + 0^2 + 0^2\right) = 0^2$$

$\theta_1 = 0.5$?

$J(1) = 0$

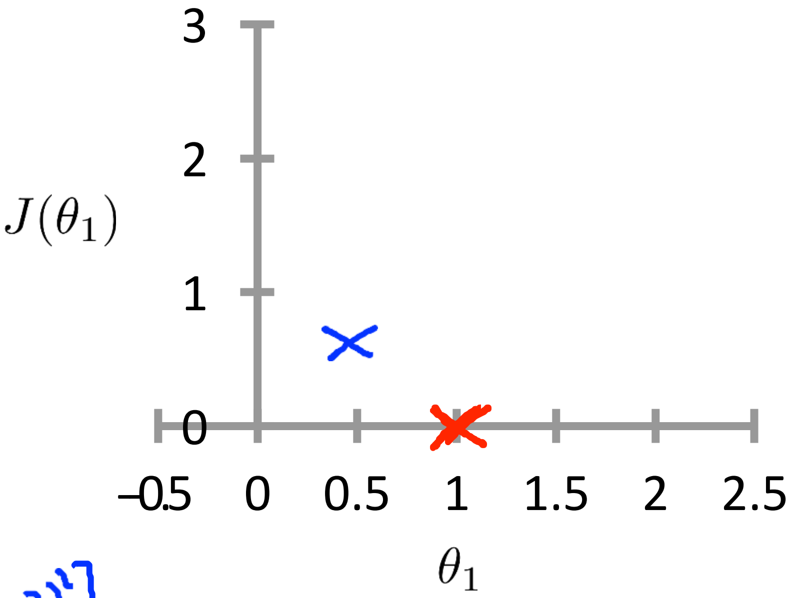
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

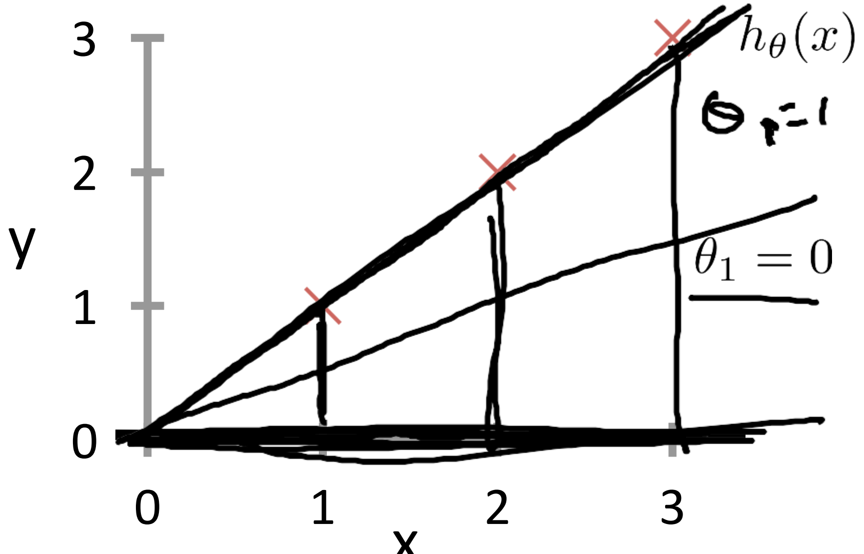
$$= \frac{1}{2\times 3}(3.5) = \frac{3.5}{6} \approx 0.58$$

$\theta_1 = 0$?

$J(0) = $?

$h_\theta(x)$

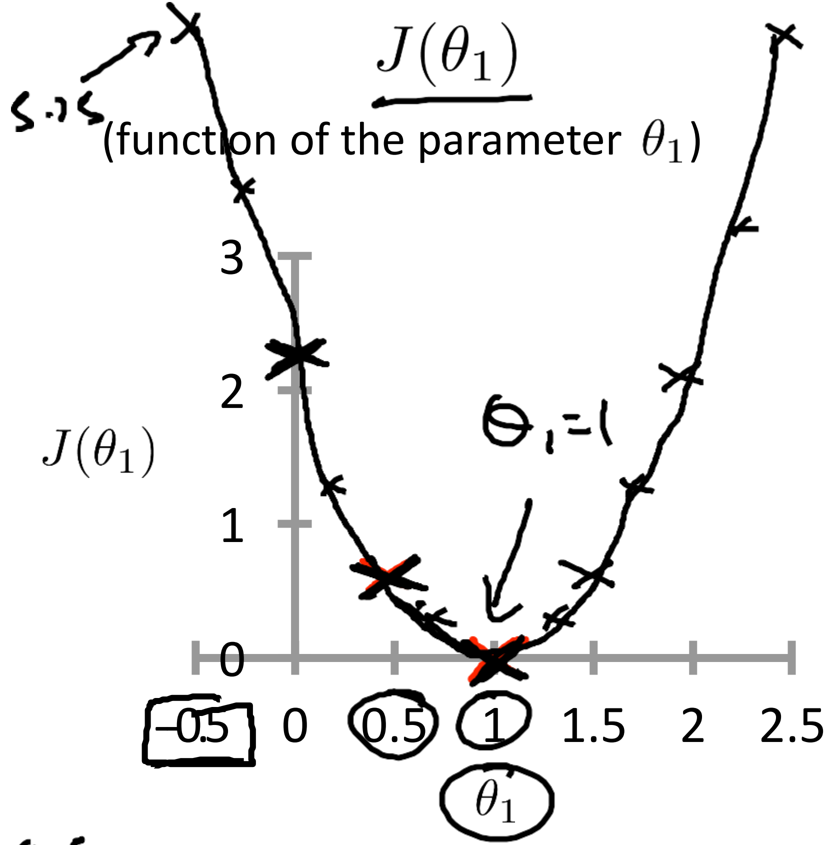
(for fixed θ_1, this is a function of x)



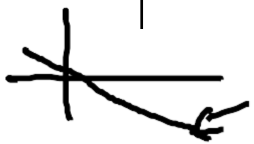
$$J(0) = \frac{1}{2m}(1^2 + 2^2 + 3^2) = \frac{1}{6} \cdot 14 \approx 2.3$$

$J(\theta_1)$

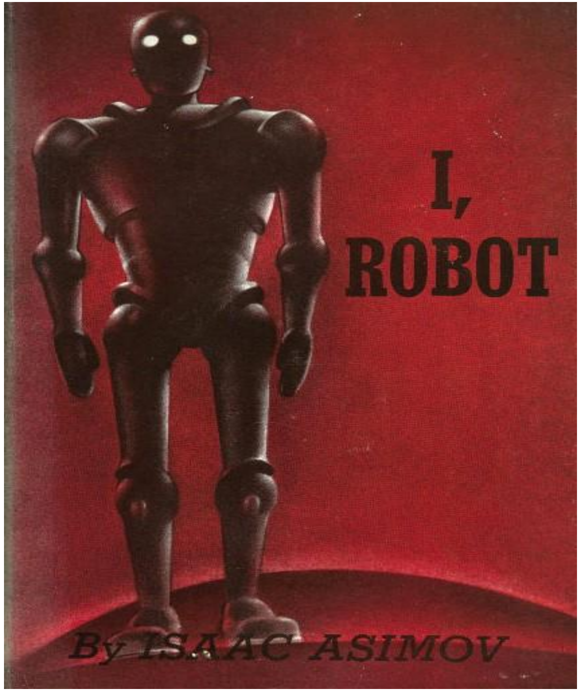
(function of the parameter θ_1)



h(x) = −0.5



$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$



Machine Learning

Linear regression with one variable

Cost function intuition II

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

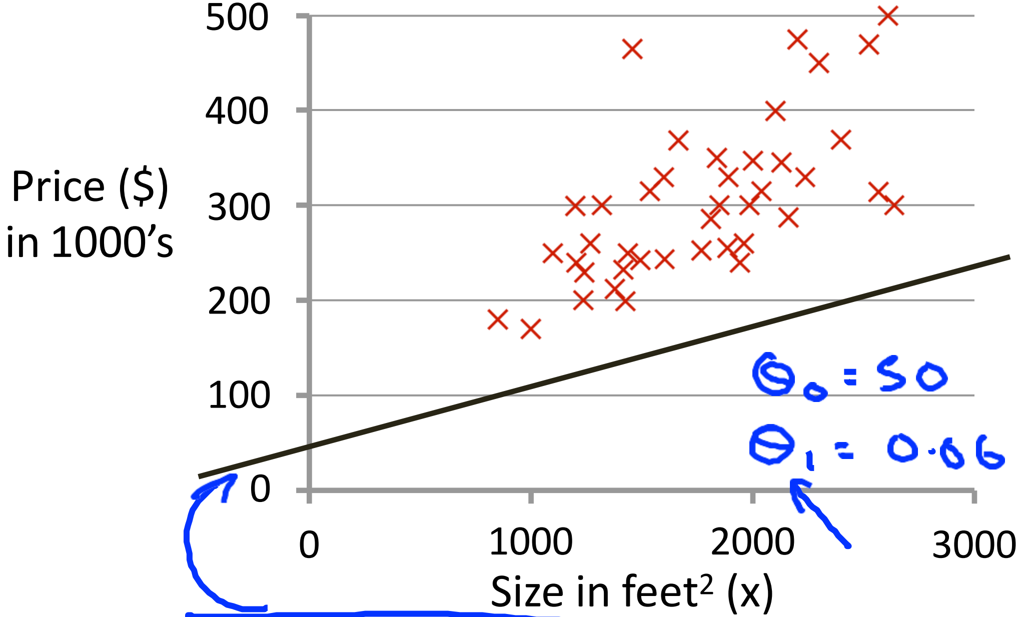
Parameters: θ_0, θ_1

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

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

$h_\theta(x)$

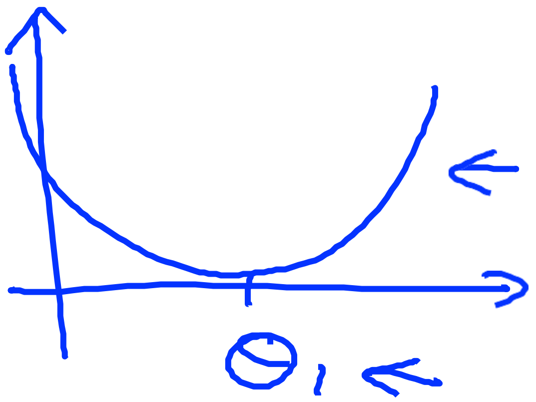
(for fixed θ_0, θ_1, this is a function of x)



$$h_\theta(x) = 50 + 0.06x$$

$J(\theta_0, \theta_1)$

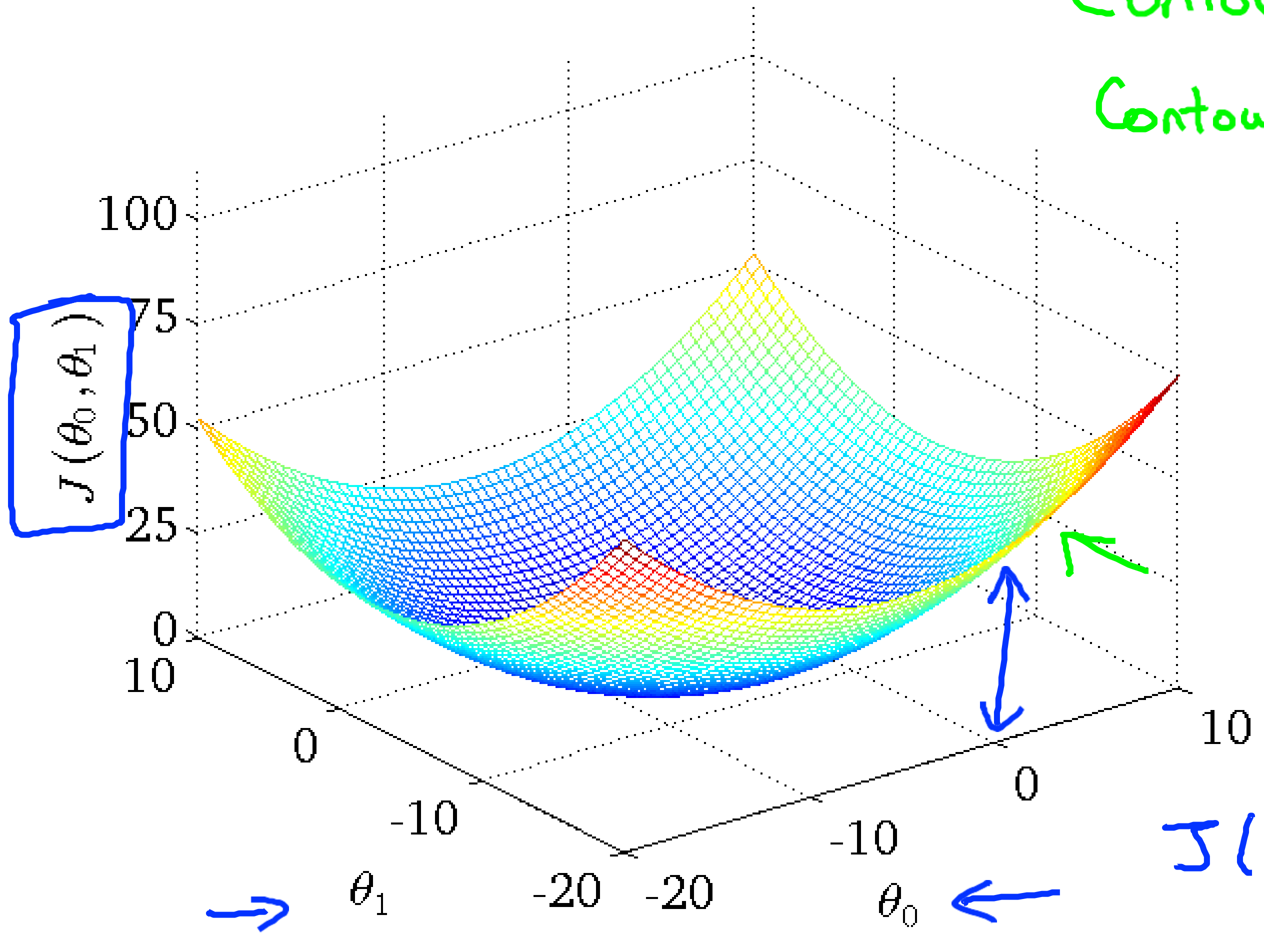
(function of the parameters θ_0, θ_1)



$$\theta_0, \theta_1$$

Contour plots

Contour figures.



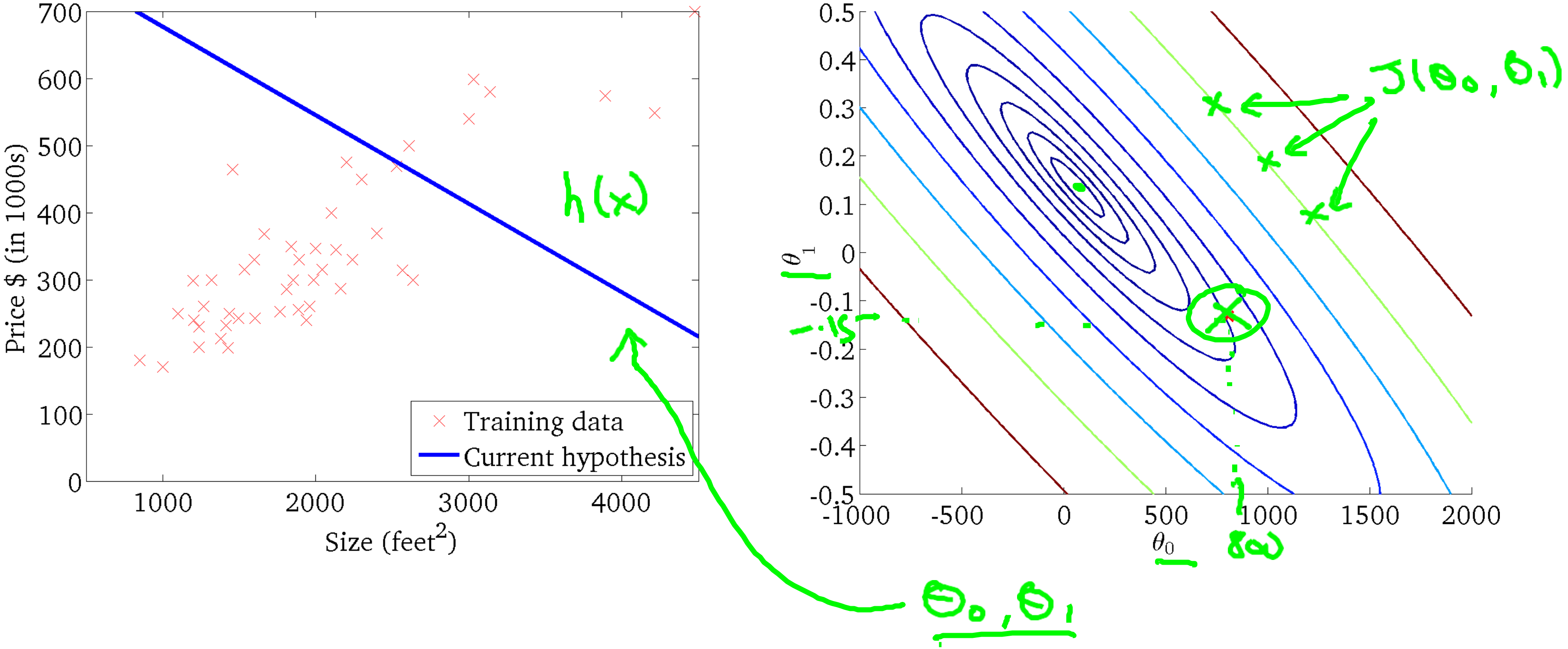
$J(\theta_0, \theta_1)$

$h_\theta(x)$

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

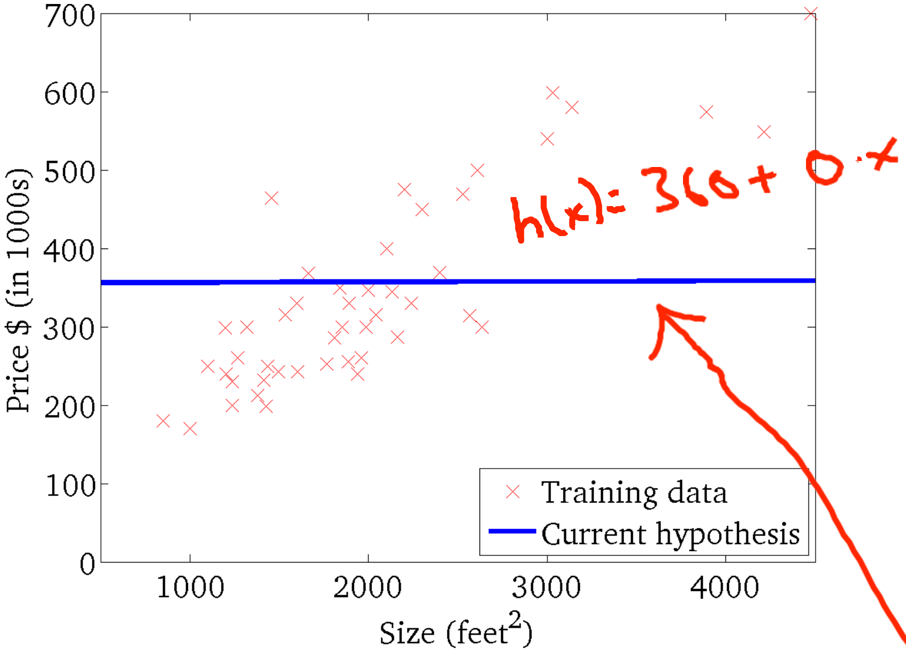
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



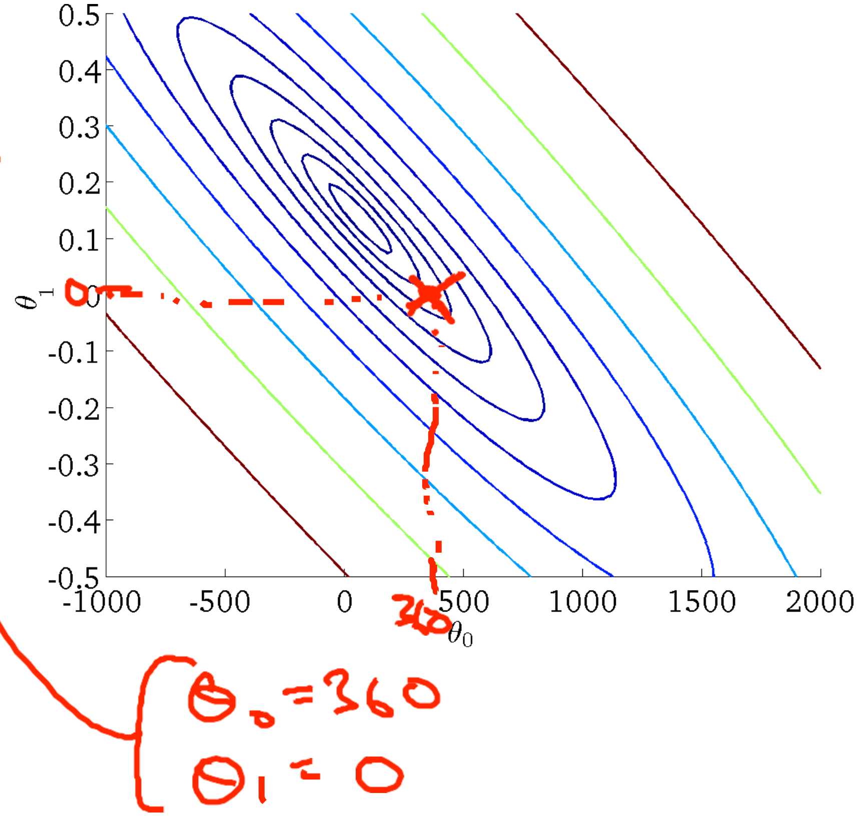
$h_\theta(x)$

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



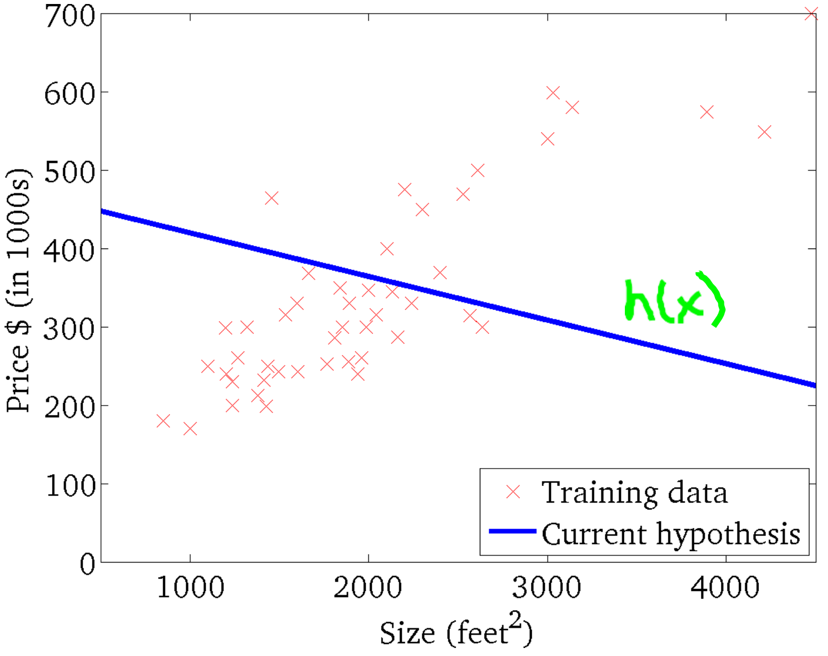
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



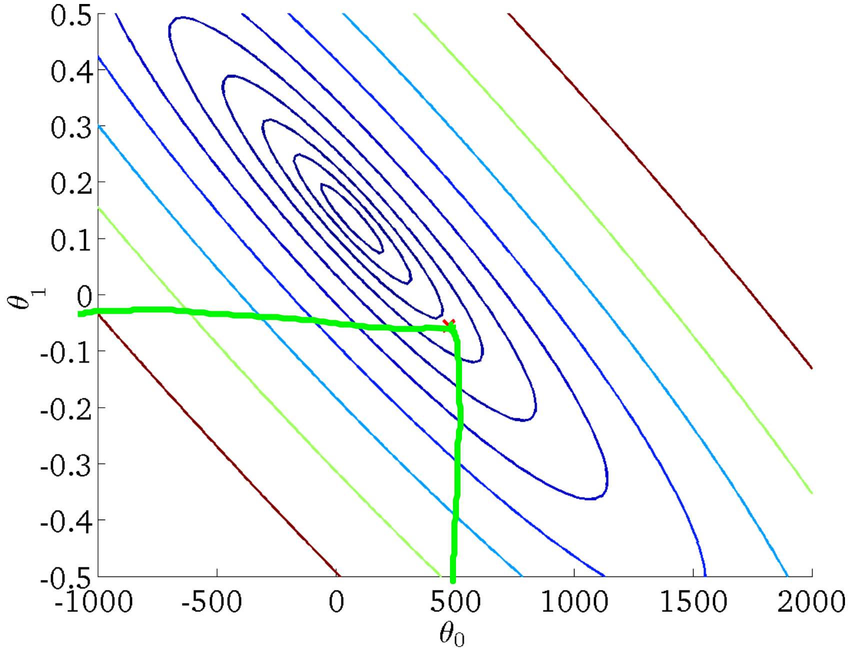
$h_\theta(x)$

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



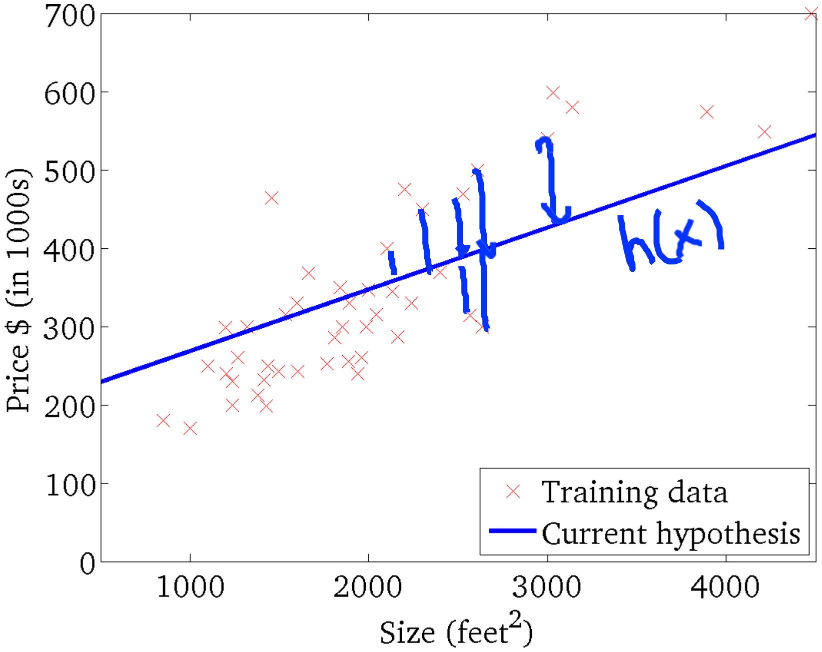
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



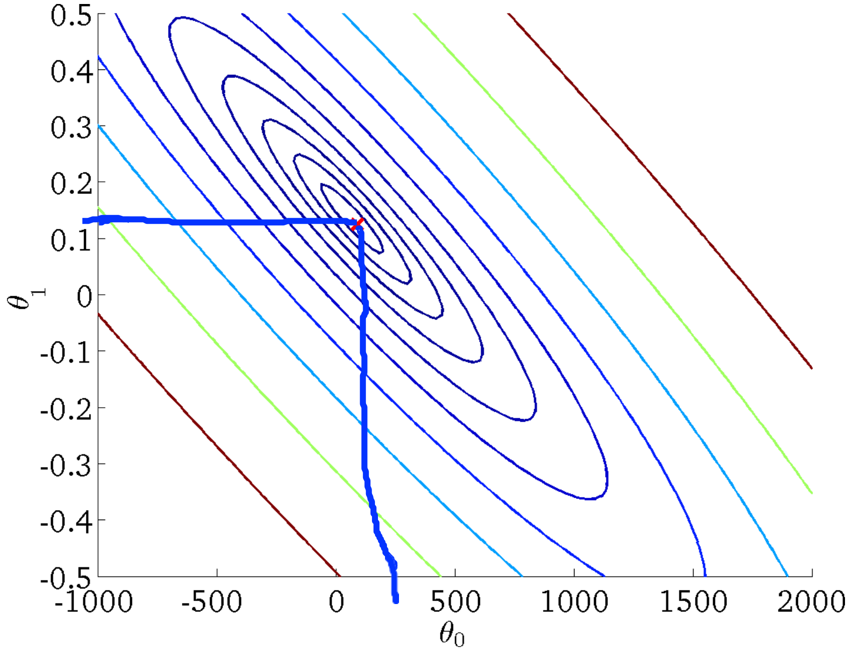
$h_\theta(x)$

(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)



Definations

Cross-entropy loss, also known as log loss, is a widely used loss function for classification problems, especially in settings where we are making predictions for categorical outcomes.

It measures the performance of a classification model whose output is a probability value between 0 and 1.

Cross-entropy loss increases as the predicted probability diverges from the actual label, making it an effective cost function for evaluating how well the model predicts the actual class.

Mathematical Definition

Given a true label *y* and a predicted probability *p*, for a binary classification problem, the cross-entropy loss can be defined as:

$$L(y,p)=-(y\log(p)+(1-y)\log(1-p))$$

. If y=1, the loss is $-\log(p)$, which becomes large as p approaches 0.
. If y=0, the loss is $-\log(1-p)$, which becomes large as p approaches 1.

For multi-class classification problems, where there are more than two classes, the cross-entropy loss is generalized as follows:

$$L=-\sum_{c=1}^{M} y_{0,c}\log\left(p_{0,c}\right)$$

Here, *M* is the number of classes, *yo,c* is a binary indicator of whether class *c* is the correct classification for observation *o*, and *po,c* is the predicted probability that observation *o* is of class *c*.

Properties and Use

- **Sensitivity to misclassifications:** Cross-entropy loss penalizes incorrect predictions heavily, especially those that are confidently wrong.
- **Application to probabilities:** It is well-suited for models like logistic regression or neural networks that output probabilities.
- **Encourages accurate probabilities:** Not only does it penalize wrong classifications, but it also penalizes correct classifications that are made with low confidence.
- **Gradient Descent Optimization:** It works well with optimization algorithms like gradient descent, as its derivative with respect to the model parameters can be efficiently calculated, facilitating model training.

Cross-entropy loss is a cornerstone of modern machine learning for classification tasks, enabling models to learn by iteratively minimizing the difference between predicted probabilities and actual class labels.

Example of Cross-Entropy Loss Calculation

Let's say we have a binary classification problem where we want to predict whether a given image contains a cat or not. Here are the true labels and the predicted probabilities for a batch of four images:

True label: Cat (1), Predicted probability of being a cat: 0.9

True label: No Cat (0), Predicted probability of being a cat: 0.2

True label: Cat (1), Predicted probability of being a cat: 0.3

True label: No Cat (0), Predicted probability of being a cat: 0.8

The cross-entropy loss for each instance would be calculated as follows:

For the first image: $-(1\cdot\log(0.9)+(1-1)\cdot\log(1-0.9))-(1\cdot\log(0.9)+(1-1)\cdot\log(1-0.9))$

For the second image: $-(0\cdot\log(0.2)+(1-0)\cdot\log(1-0.2))-(0\cdot\log(0.2)+(1-0)\cdot\log(1-0.2))$

For the third image: $-(1\cdot\log(0.3)+(1-1)\cdot\log(1-0.3))-(1\cdot\log(0.3)+(1-1)\cdot\log(1-0.3))$

For the fourth image: $-(0\cdot\log(0.8)+(1-0)\cdot\log(1-0.8))-(0\cdot\log(0.8)+(1-0)\cdot\log(1-0.8))$

Then, to get the overall cross-entropy loss for this batch, you would calculate the average of the individual losses.

The cross-entropy loss for each of the four instances in the batch is as follows:

1. For the first image (True label: Cat): The loss is approximately 0.1054

2. For the second image (True label: No Cat): The loss is approximately 0.2231

3. For the third image (True label: Cat): The loss is approximately 1.2040

4. For the fourth image (True label: No Cat): The loss is approximately 1.6094

The average cross-entropy loss for this batch of instances is approximately 0.7855.

This value represents the average penalty for the differences between the predicted probabilities and the actual labels, with a lower score indicating better performance of the model on this batch.