Linear classifiers

Gosia Migut

Admin stuff

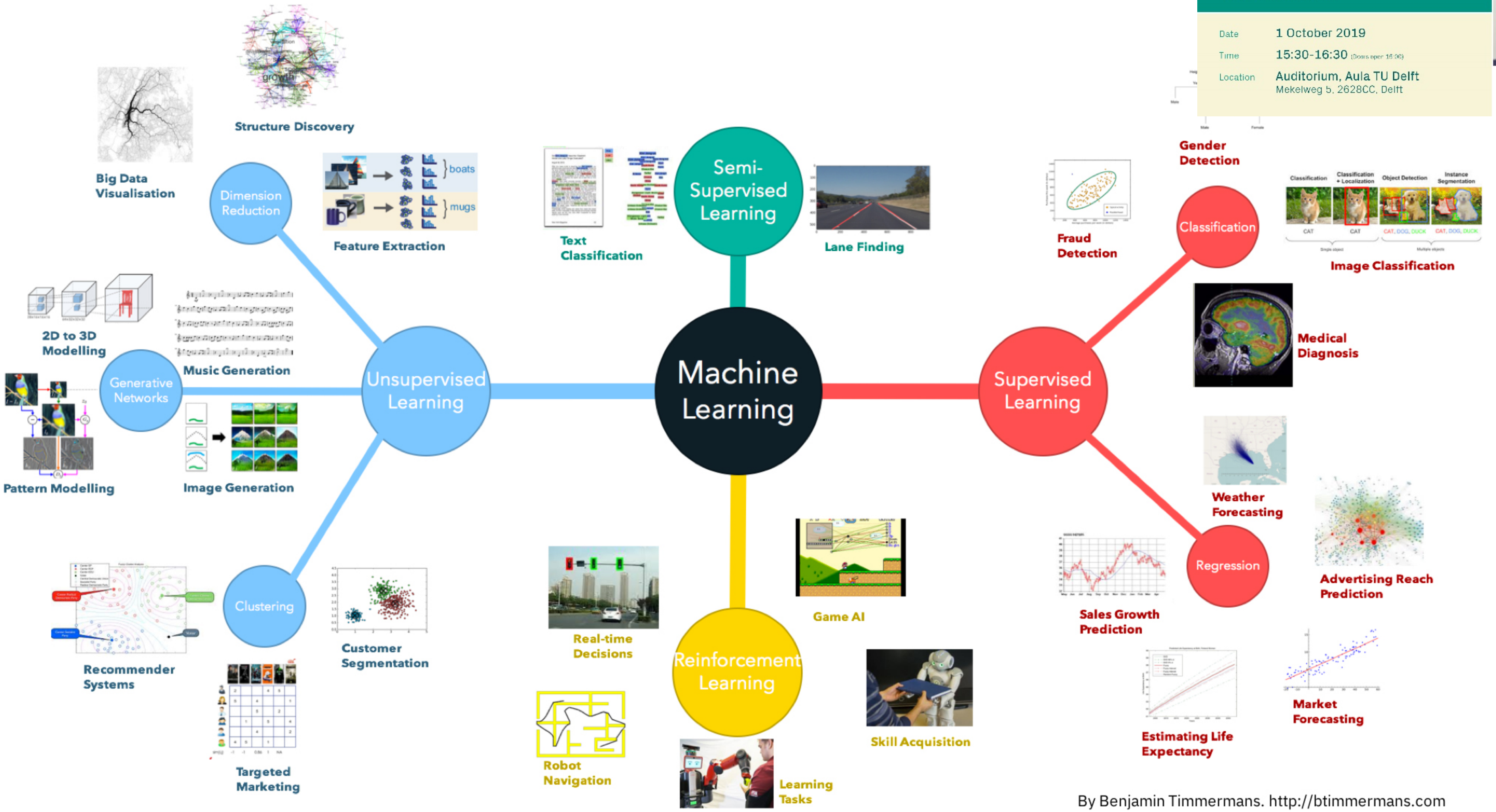
- Feedback (thank you, also student panel!)
  - I'm staying!
  - 2nd year bachelor CSE course
  - Digital practice exam in week 5
  - No labs answers, TA's available
  - More examples with the formulas

- Lab week 4 more challenging!
  - Includes material from Tuesday and Friday lecture
  - Notation corresponds with the reading material

I “owe” you

- Will be fixed this week:
  - Pseudo code Parzen width parameter optimization
  - Solution last exercise Naive Bayes

Machine Learning overview



By Benjamin Timmermans. http://btimmermans.com

Generative vs discriminative models

- A **generative** model explicitly models the joint probability distribution $p(x|y)$ and then uses Bayes rule to compute posterior probabilities $p(y|x)$
  – Parametric density estimation: eg. Nearest mean, LDA, QDA
  – Non-parametric density estimation: eg. k-nn, Parzen, Naive Bayes

- A **discriminative** model directly models $p(y|x)$ from the training examples.
  – Linear: eg. logistic regression, svm
  – Non-linear: eg. decision trees, multi-layer perceptron

Learning objectives of this lecture

- After this lecture you will be able to explain:
  - what the general idea of linear classification is
  - what $w^T x$ means
  - What a cost function is
  - the gradient descent algorithm
  - how to optimize a cost function using gradient descent
  - what the difference between gradient descent and stochastic gradient descent is.

Reading of this week

- CS229 Lecture notes by Andrew Ng (Standford University):
  - Supervised learning p.1-2
  - Part I Linear regression p. 3-4
  - 1. LMS algorithm p. 4-7
  - 3. Probabilistic interpretation p. 11-13
  - Part II Classification and logistic regression p. 16-19

  http://cs229.stanford.edu/notes/cs229-notes1.pd

- Lab of week 4 is consistent with the notation of the reading

Linear classifiers

Note on the notation

- Parameters notation
  - In the lecture we use **w**
  - In the reading and lab θ is used

Linear classifier

- Linear classifer has a **linear decision boundary**.

- Decision boundary of a linear classifier for 2 dimensions is a line:
$$w_1x_1 + w_2x_2 + w_0 = 0$$

- A hyperplane is a generalization of a straight line to > 2 dimensions.
- A hyperplane contains all the points in a d dimensional space satisfying the following equation
$$w^Tx + w_0 = 0$$

Linear classifier: terminology

$$h(x) = w^T x + w_0$$

- The slope of the hyperplane is determined by the **parameter (weight) vector** $\mathrm{w} = (w_1, \ldots, w_d)$.
- The location (intercept) is determined by **bias** w_0.
- The function of the input h(x) is a **linear combination** of the parameters w.

Linear classifier

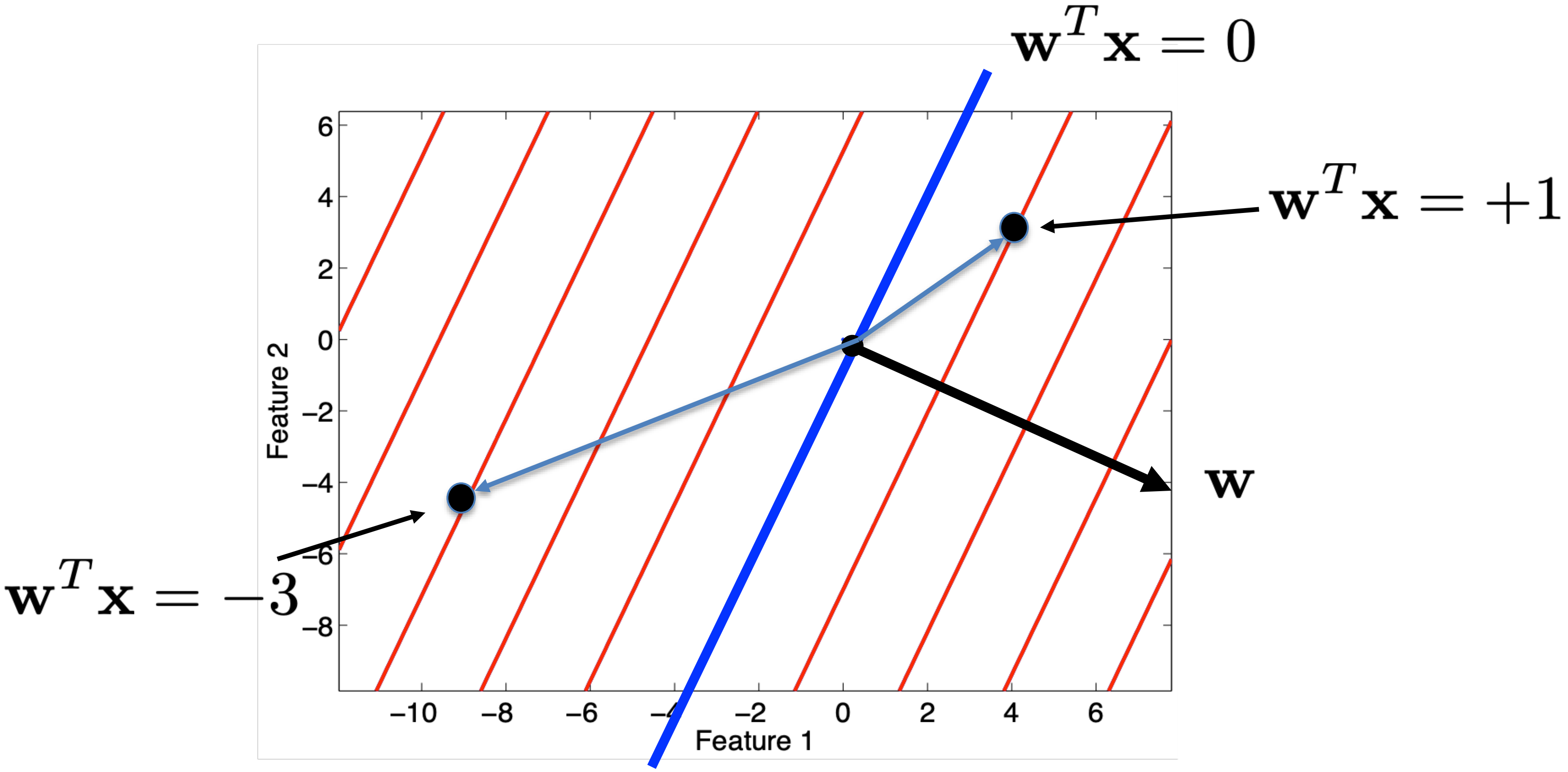
- Given the linear classifier:

$$h(x) = w^T x + w_0$$

- Classify x to $\begin{cases} y_1 \; if \; w^T x + w_0 \geq 0 \\ y_0 \; if \; w^T x + w_0 < 0 \end{cases}$

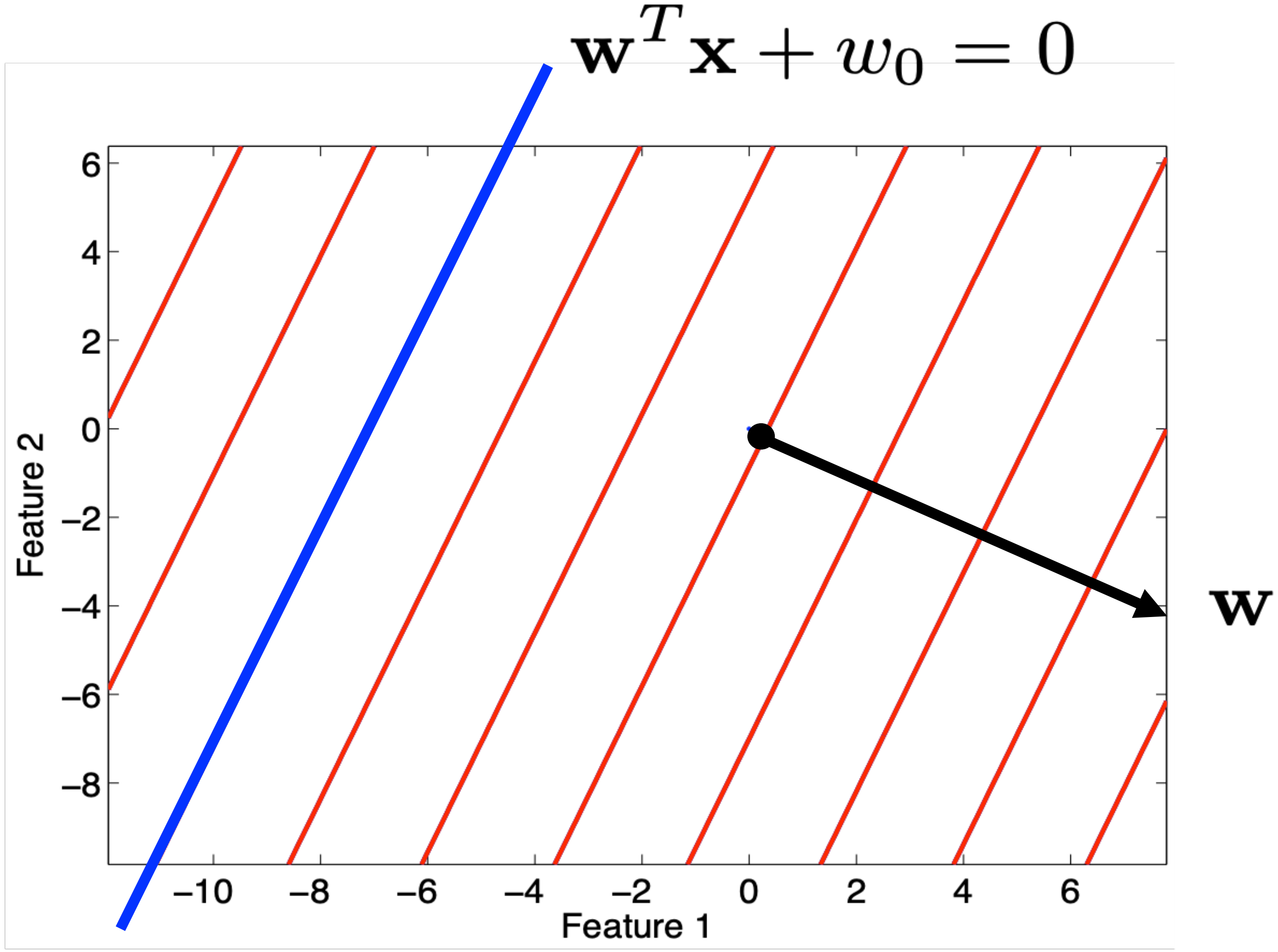
What does $w^T x$ mean?

- Assume I have a 2-dimensional w:



Slide credit: David Tax

What does $w^T x + w_0$ mean?



$w_0 = ?$

Slide credit: David Tax

Incorporate the bias term

- Quite often you see

$$h(x) = w^T x = 0$$

- Instead of

$$h(x) = w^T x + w_0 = 0$$

$$h(x) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + w_0$$

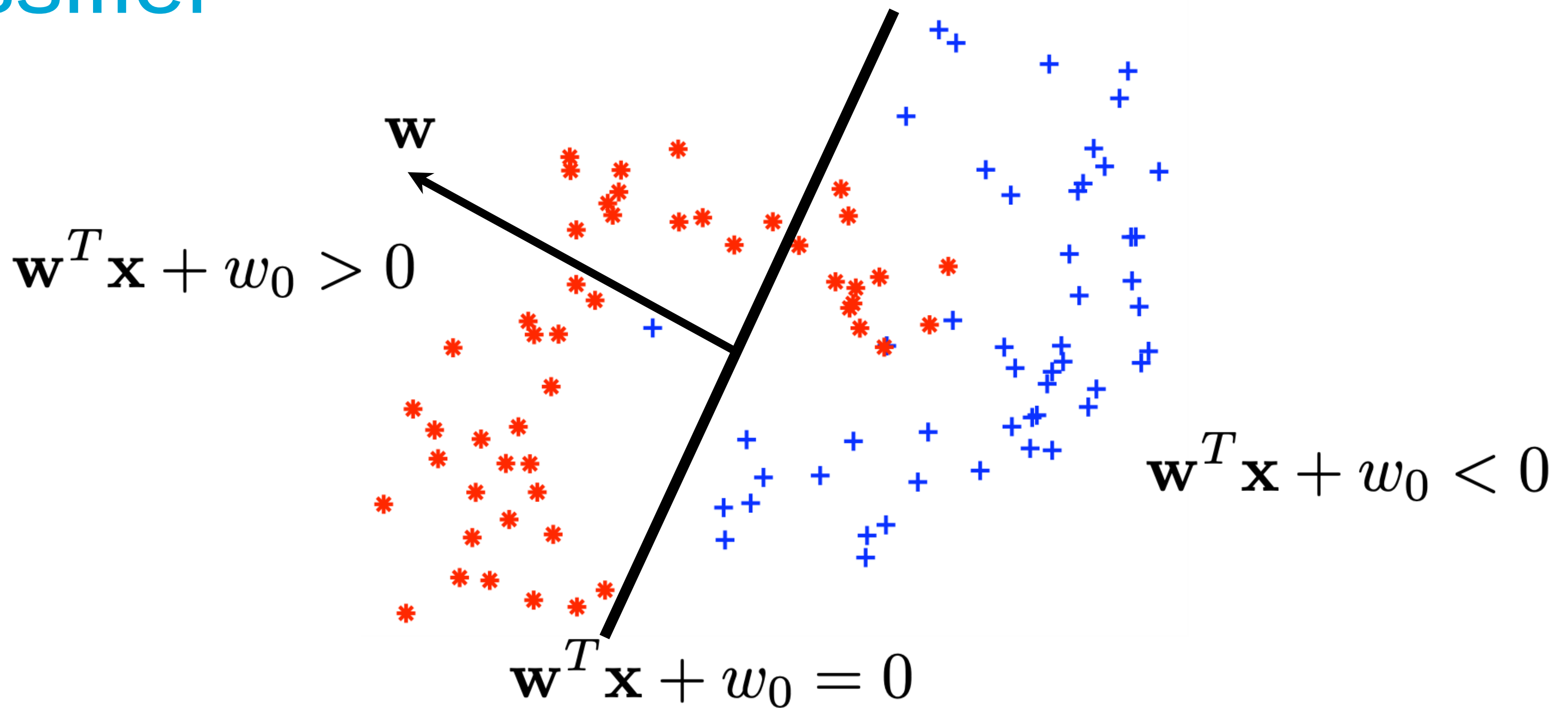
- No problem if we redefine the feature vector:

$$\tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix} => x_0 = 1$$

- $h(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d = \sum_{i=0}^{d} x_i w_i = w^T x$

Slide credit: David Tax

Linear classifier



- Classifier is a linear function of the features
- The classification depends if the weighted sum of the features is above or below 0

Linear classifier

- The goal of the learning process is to come up with a “good” weight vector w

- The learning process will use examples to guide the search of a “good” w

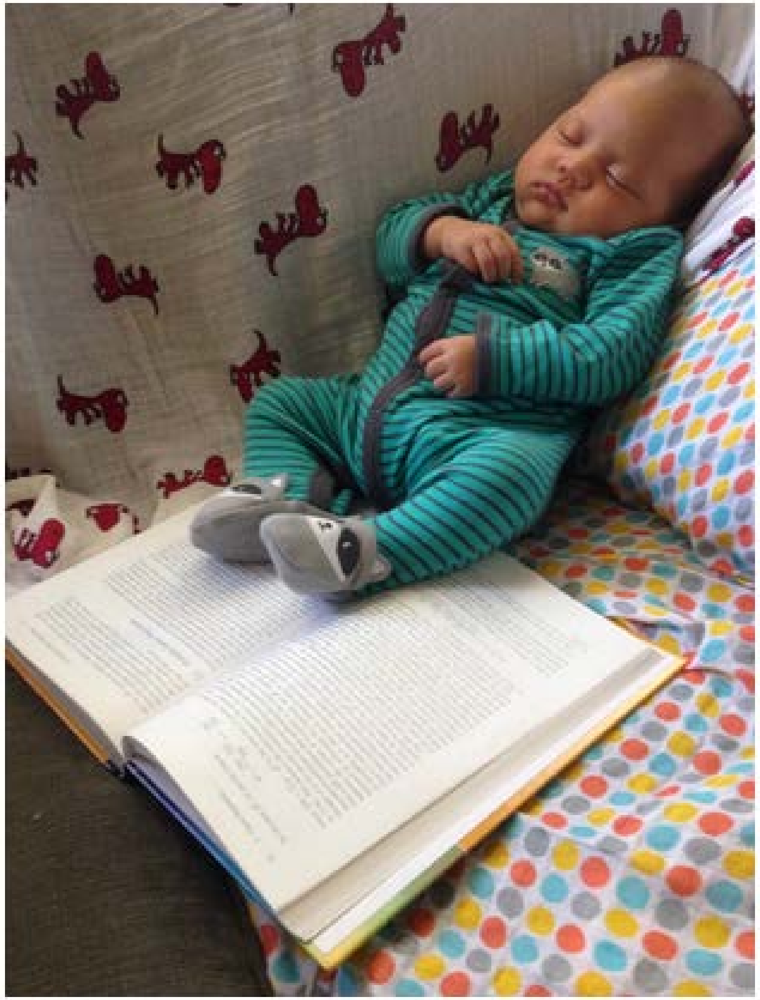
- Different notions of “goodness” exist, which yield different learning algorithms

Define a “goodness”/error measure

- Cost function
  - Measure of performance => single real number
  - Should be optimized
  - Yields different learning algortihms
  - Eg. Log-likelihood (Naive Bayes)

Cost function

Consider a regression problem…



Linear Regression?

I covered that last year.

Wake me up when we get to
Support Vector Machines!

Noah Mackey

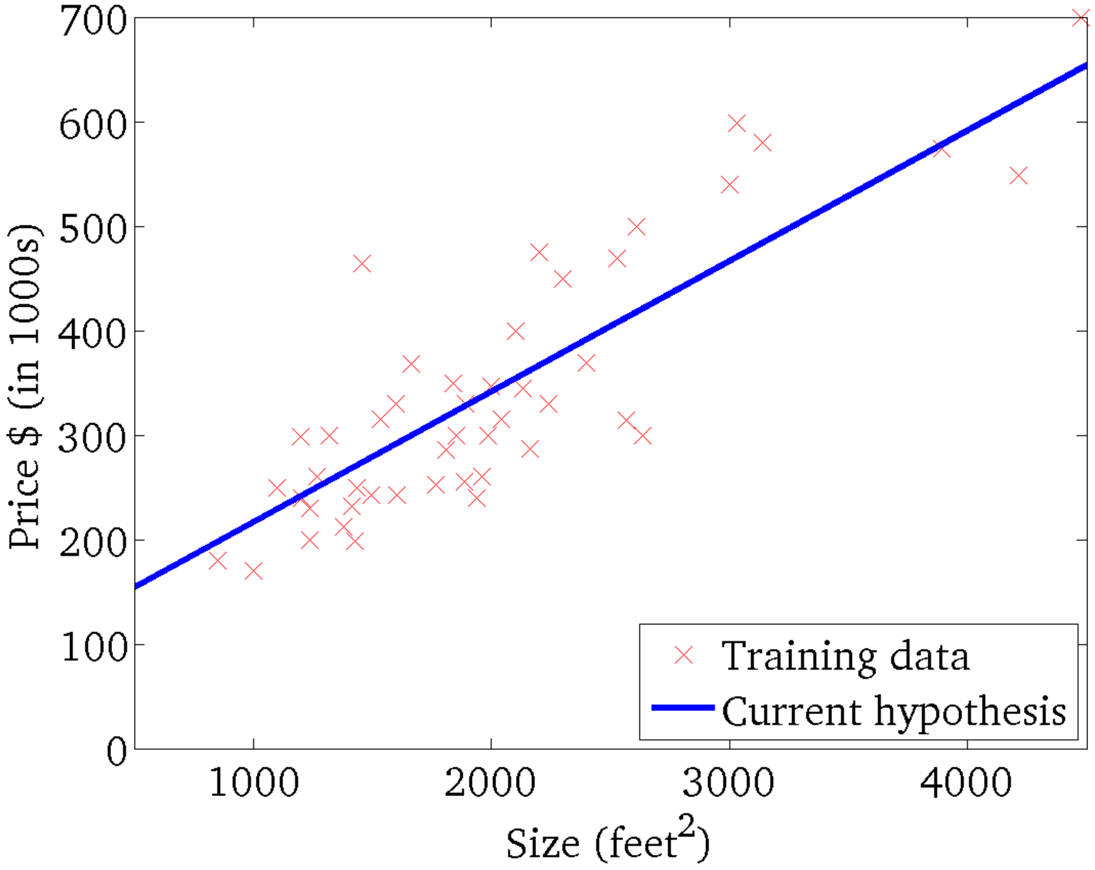
TU Delft

Univariate linear regression

- Training data: observations paired with outcomes (real number)
- Observations have features (predictors, typically also real numbers)
- The model is a regression line $y = w_1 x + w_o$ which best fits the observations:
  - w_1 is the slope
  - w_o is the intercept
  - This model has two parameters (or weigths)
  - One feature = x
  - Example:
    - x = size of property
    - y= price of property

TUDelft

Linear regression

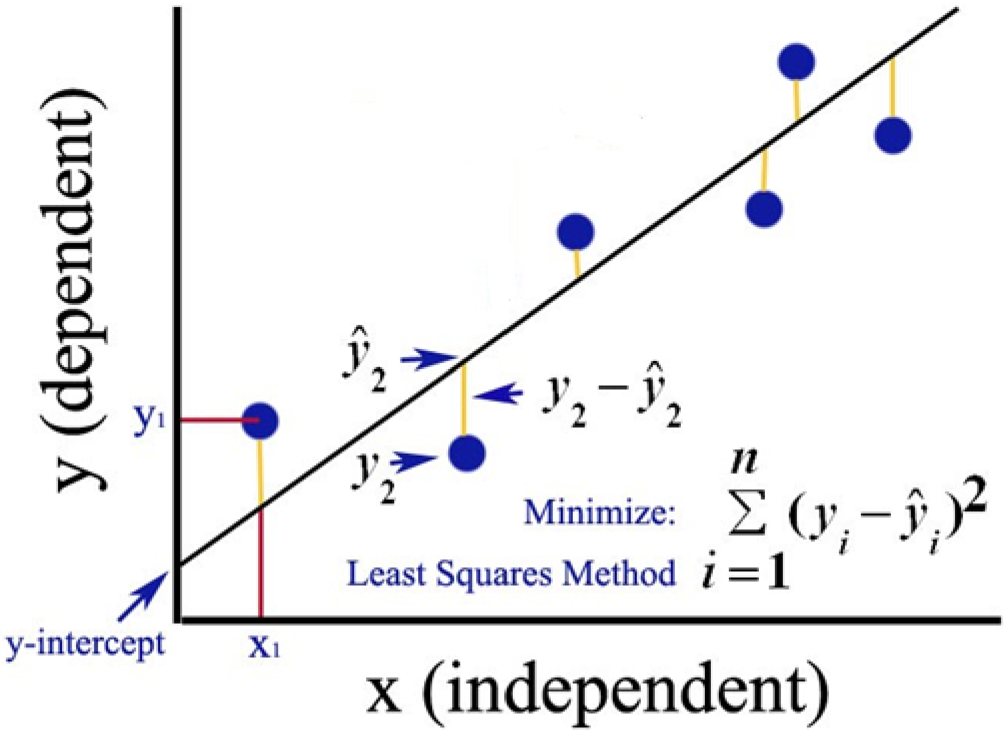


Multivariate linear regression

- More generally $y = \ w_0 + \sum_{i=0}^{d} w_i x_i$, where
  - y is outcome
  - w_0 is intercept
  - $x_1, \dots, x_d$ is feature vector and
  - $w_1, \dots, w_d$ parameter/weight vector
- Get rid of bias: $\sum_{i=0}^{d} x_i w_i = w^T x$

Cost function for linear regression

- Minimize sum squared error over N training examples



Cost function intuition

- Hypothesis: $h(w) = w_1 x_1 + w_0$

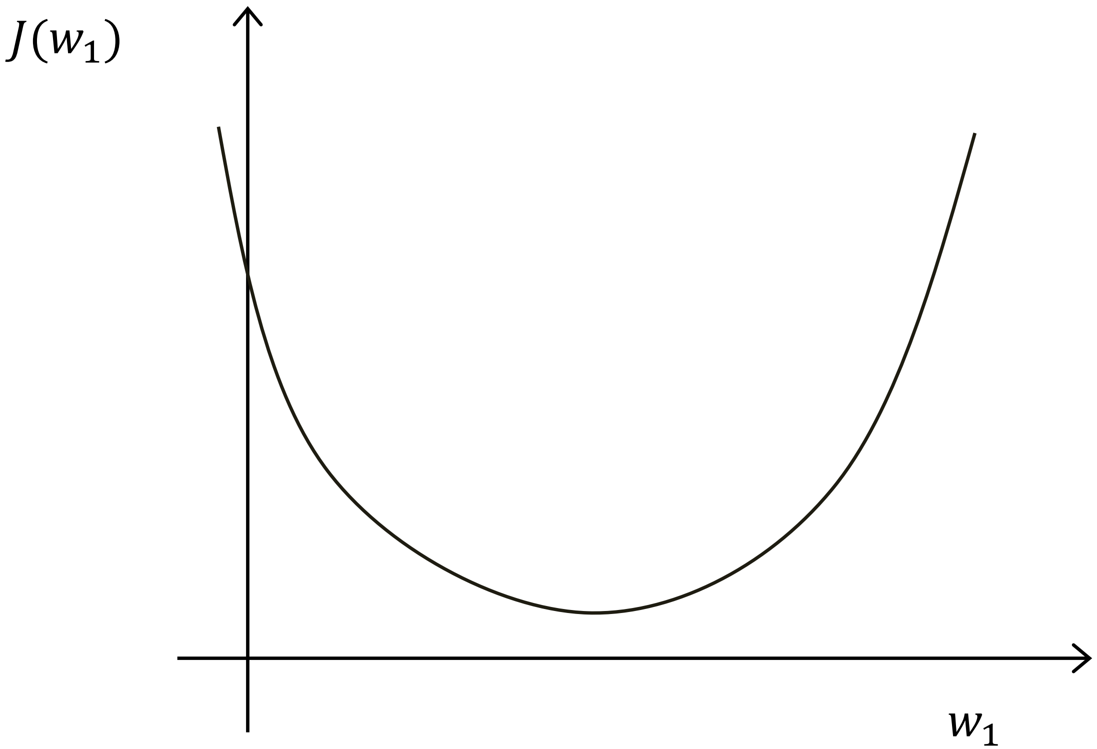
- Parameters: w_0 and w_1

- Cost function: $J(w_0, w_1) = \frac{1}{2n} \sum_{i=1} (h(x)^{(i)} - y^{(i)})^2$

- Goal: $\underset{w_0, w_1}{\text{minimize}}\, J(w_0, w_1)$

Cost function intuition

- Cost function $J(w_1)$ against w_1



Cost function optimization

- Solution: Set the derivative to 0, and solve:

$$\frac{\partial J(w)}{\partial w} = 0$$

(typically hard/impossible to do)

- Solution: Follow the derivatives (gradient) until you hit a (local) minimum.
  - What is gradient descent?
  - What is stochastic gradient descent?

Gradient descent

Gradient descent algorithm

- Goal: $\underset{w_0,w_1}{\text{minimize}}\, J(w_0, w_1)$

- Outline:
  - Start with some w_0, w_1 (eg. $w_0 = 5, w_1 = 0.167$)
  - Keep changing w_0, w_1 to reduce $J(w_0, w_1)$

- Repeat untill convergence {

$$\boxed{w_j := w_j - \alpha \frac{\partial J(w_0, w_1)}{\partial w_j}}$$

What does it all mean?

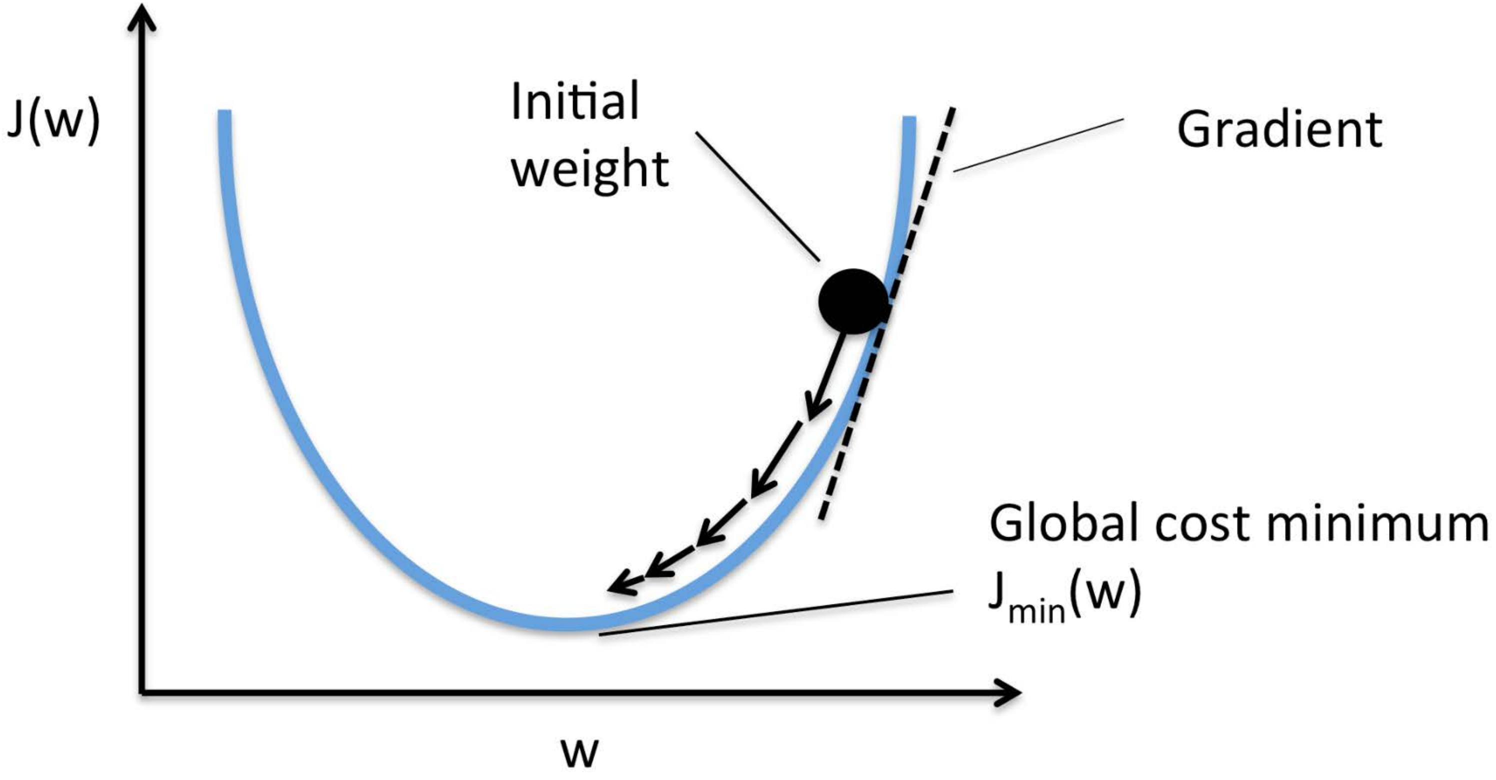
}

Gradient vector

$$w_j := w_j - \alpha \frac{\partial J(w_0, w_1)}{\partial w_j}$$

- Gradient vector has as coordinates the partial derivatives of a function

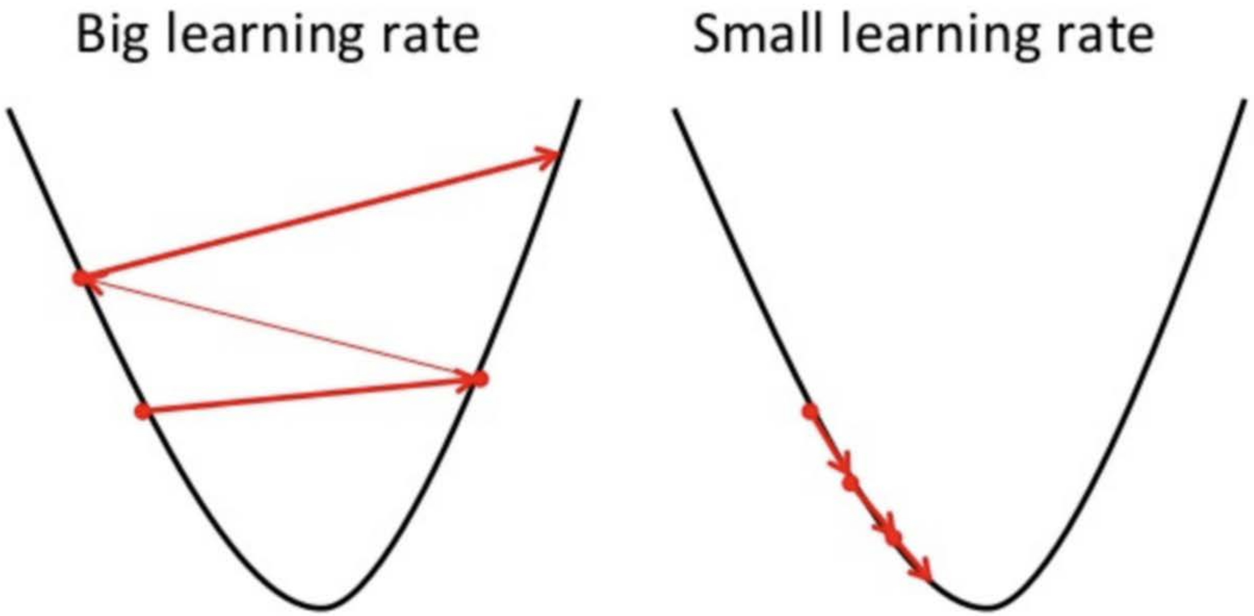
- α is learning rate = speed of descent



Learning rate

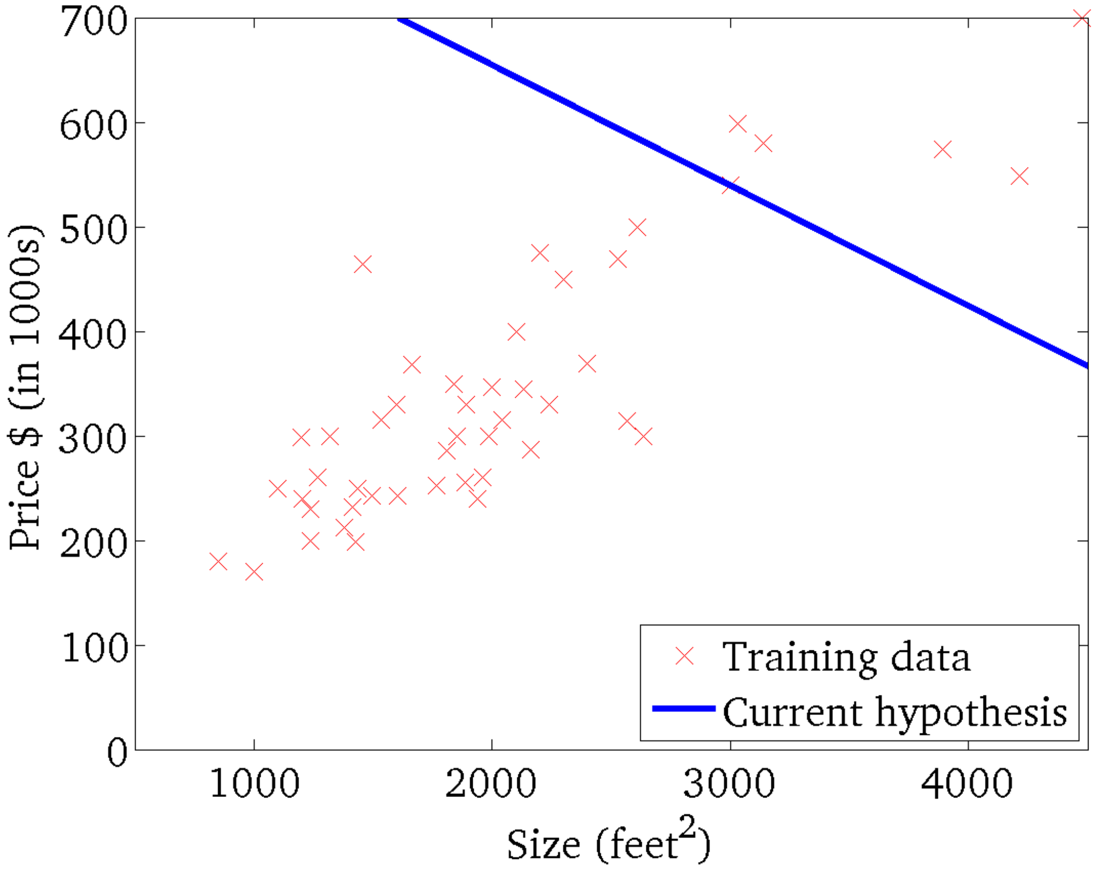
$$w_j := w_j - \alpha \frac{\partial J(w)}{\partial w_j}$$

- If α is too small gradient descent can be slow
- If α is too large, gradient descent can overshoot the minimum (it may fail to converge)



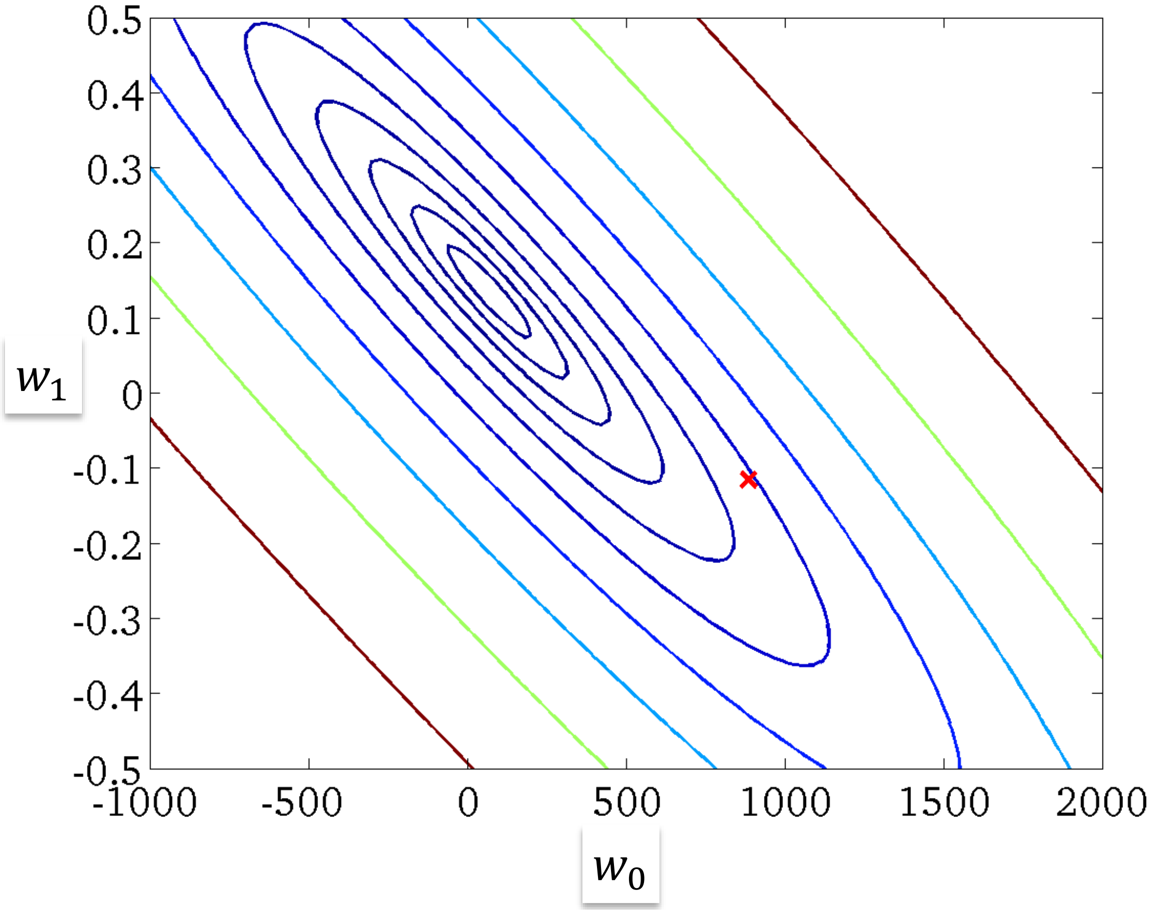
h(x)

(for fixed w_0 , w_1 this is a function of x)



$J(w_0, w_1)$

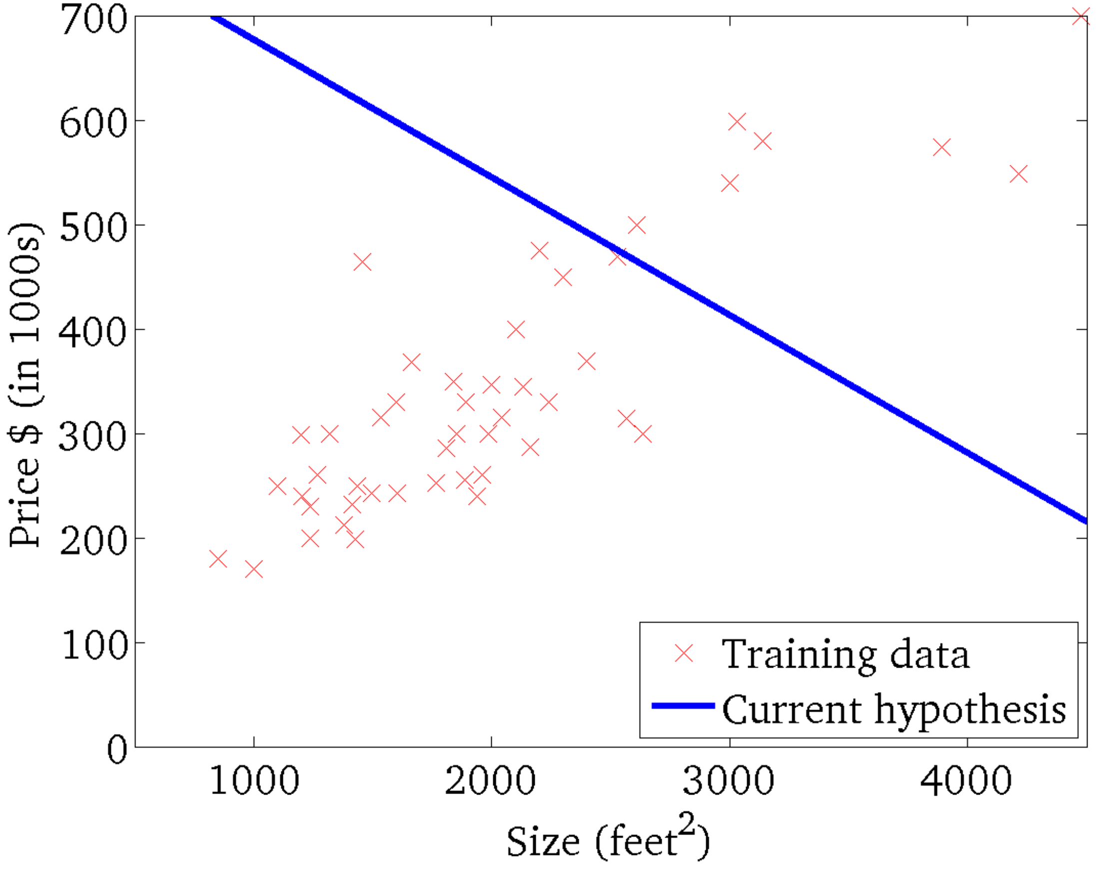
(function of the parameters w_0, w_1)



Slide credit: Andrew Ng

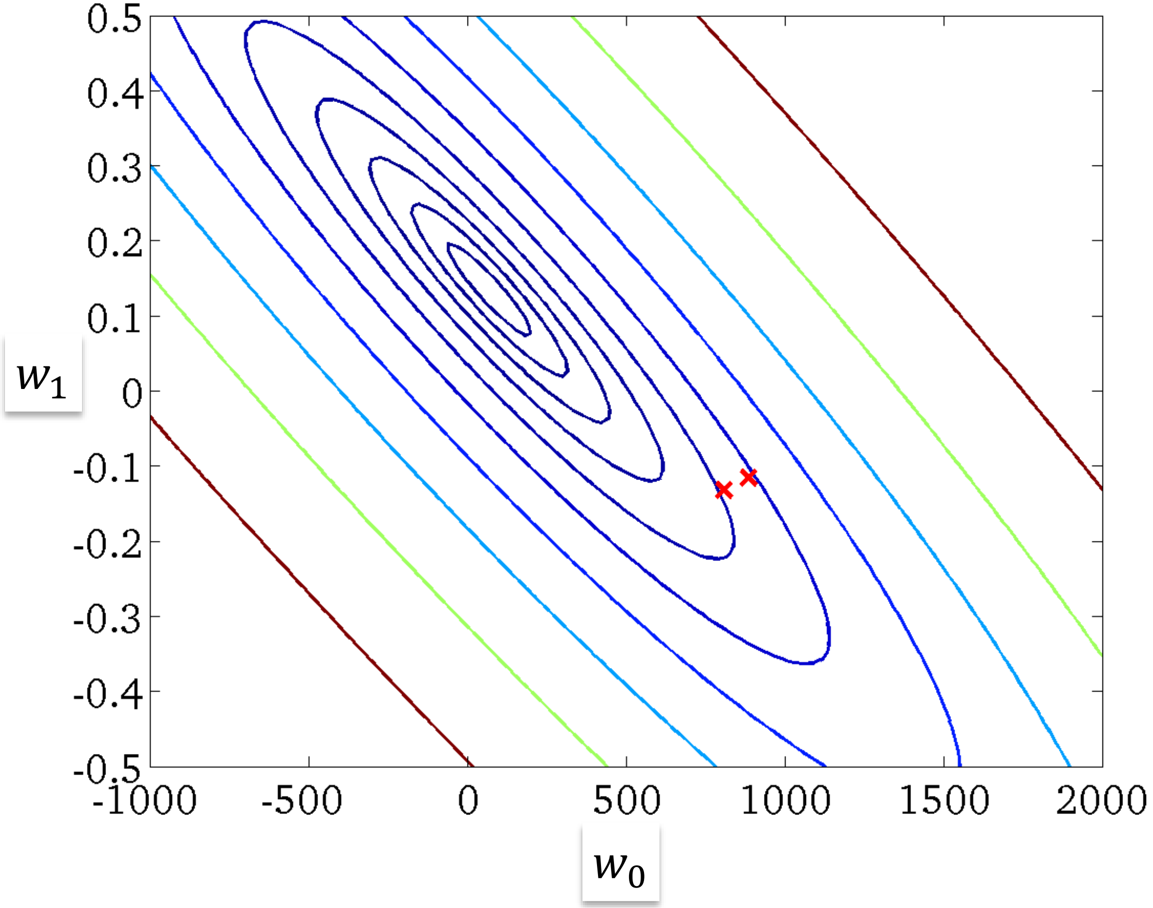
h(x)

(for fixed w_0 , w_1 this is a function of x)



$J(w_0, w_1)$

(function of the parameters w_0, w_1)

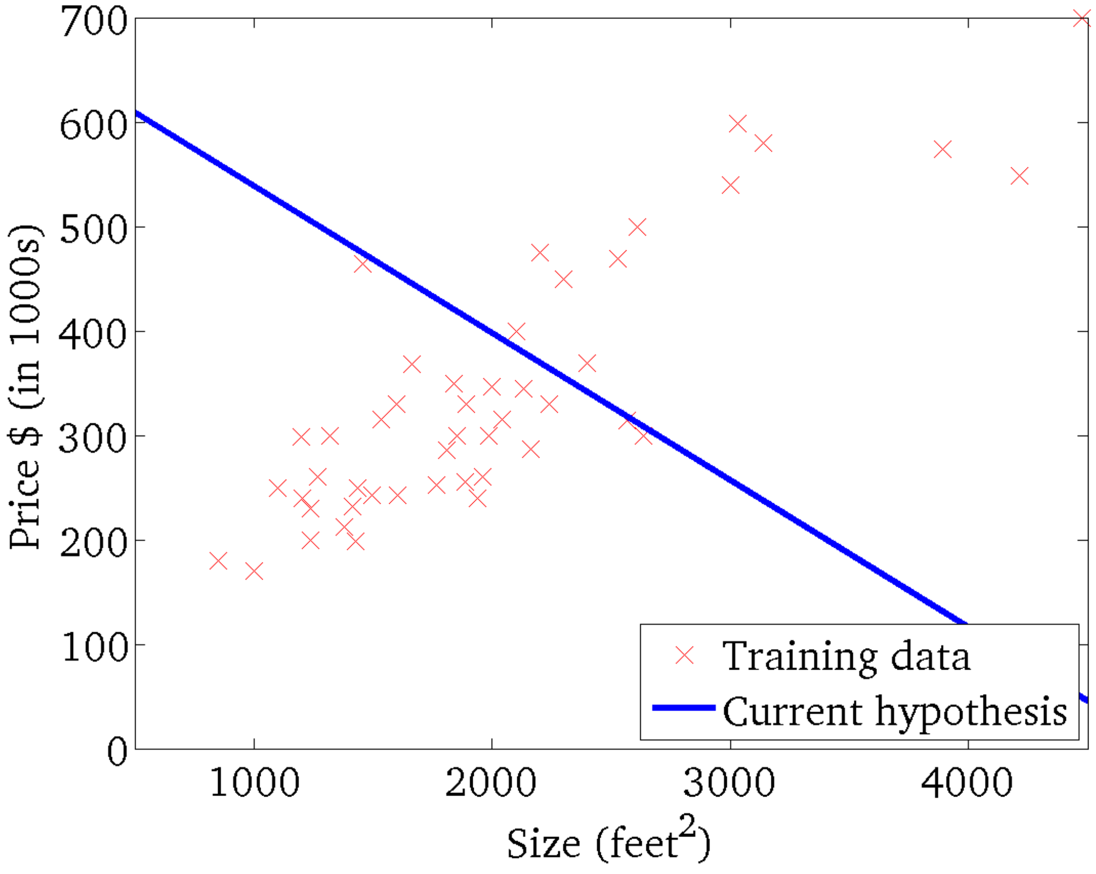


TU Delft

Slide credit: Andrew Ng

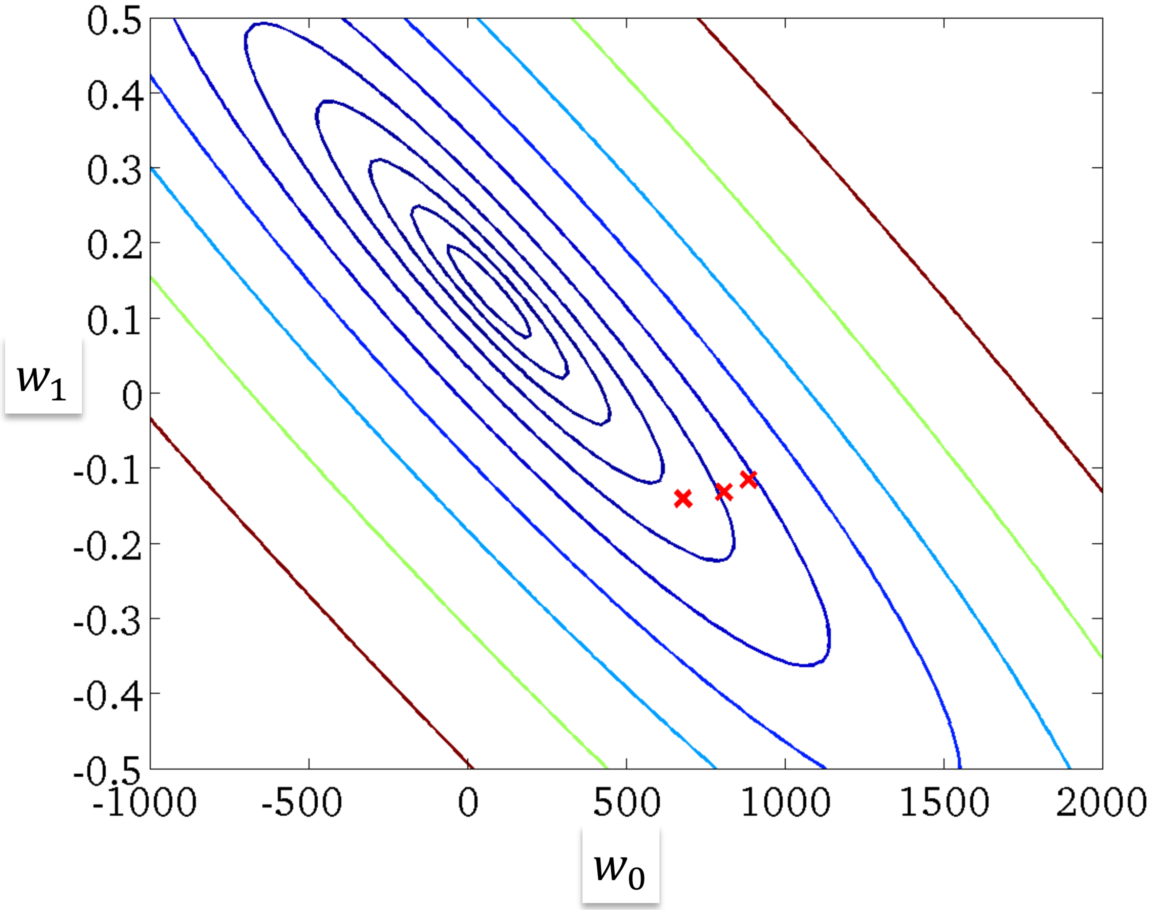
h(x)

(for fixed w_0 , w_1 this is a function of x)



$J(w_0, w_1)$

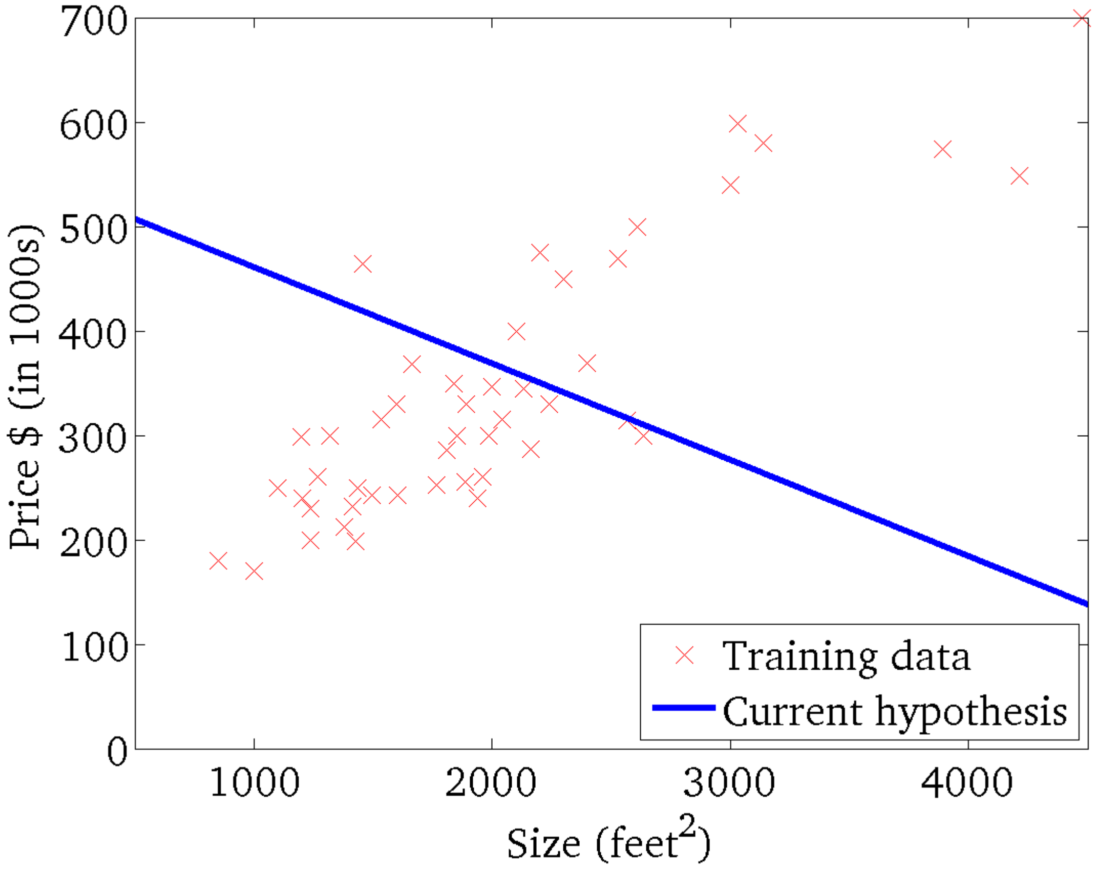
(function of the parameters w_0, w_1)



Slide credit: Andrew Ng

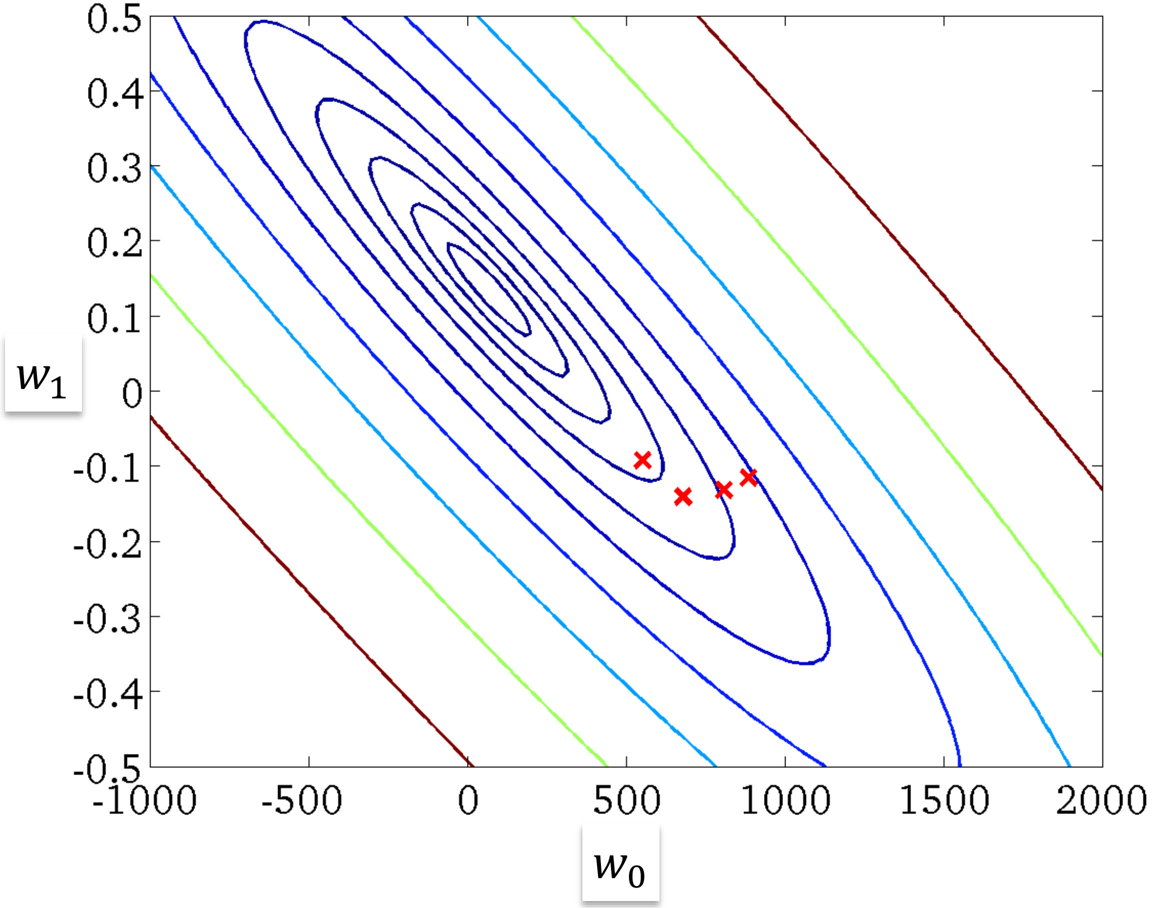
h(x)

(for fixed w_0 , w_1 this is a function of x)



$J(w_0, w_1)$

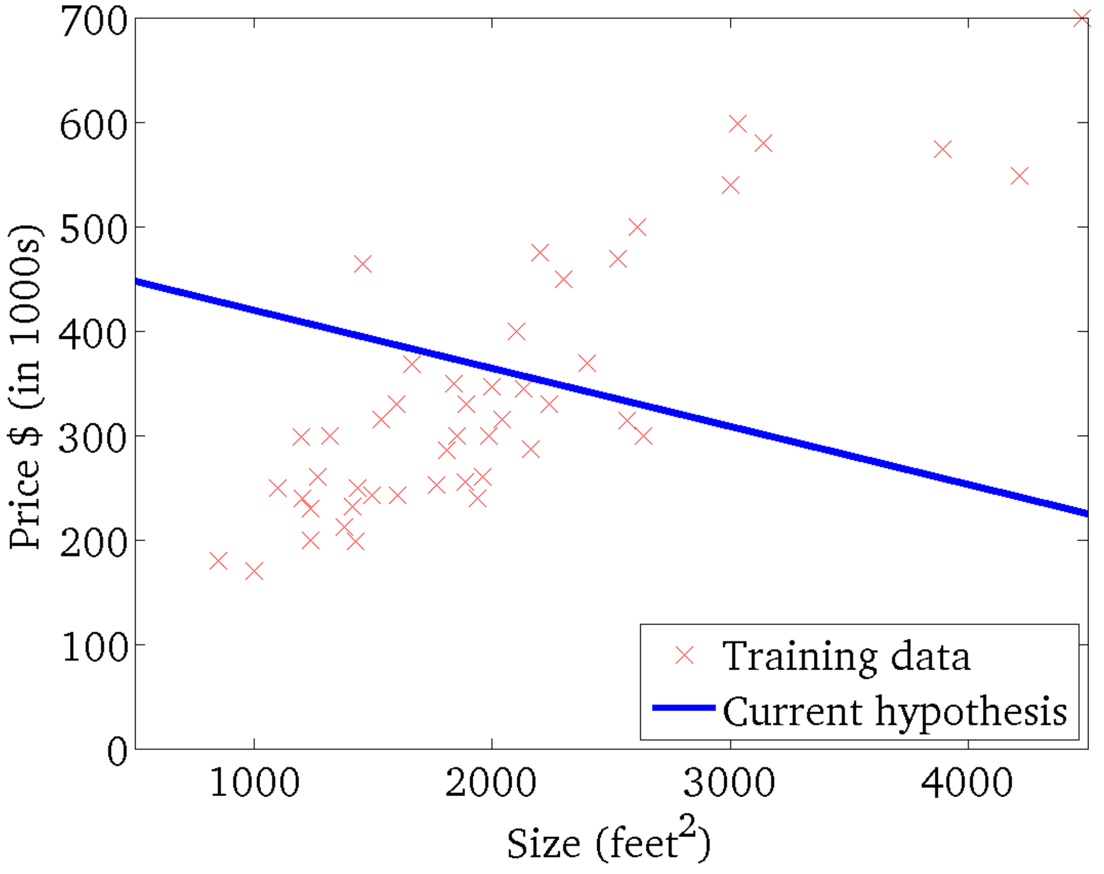
(function of the parameters w_0, w_1)



Slide credit: Andrew Ng

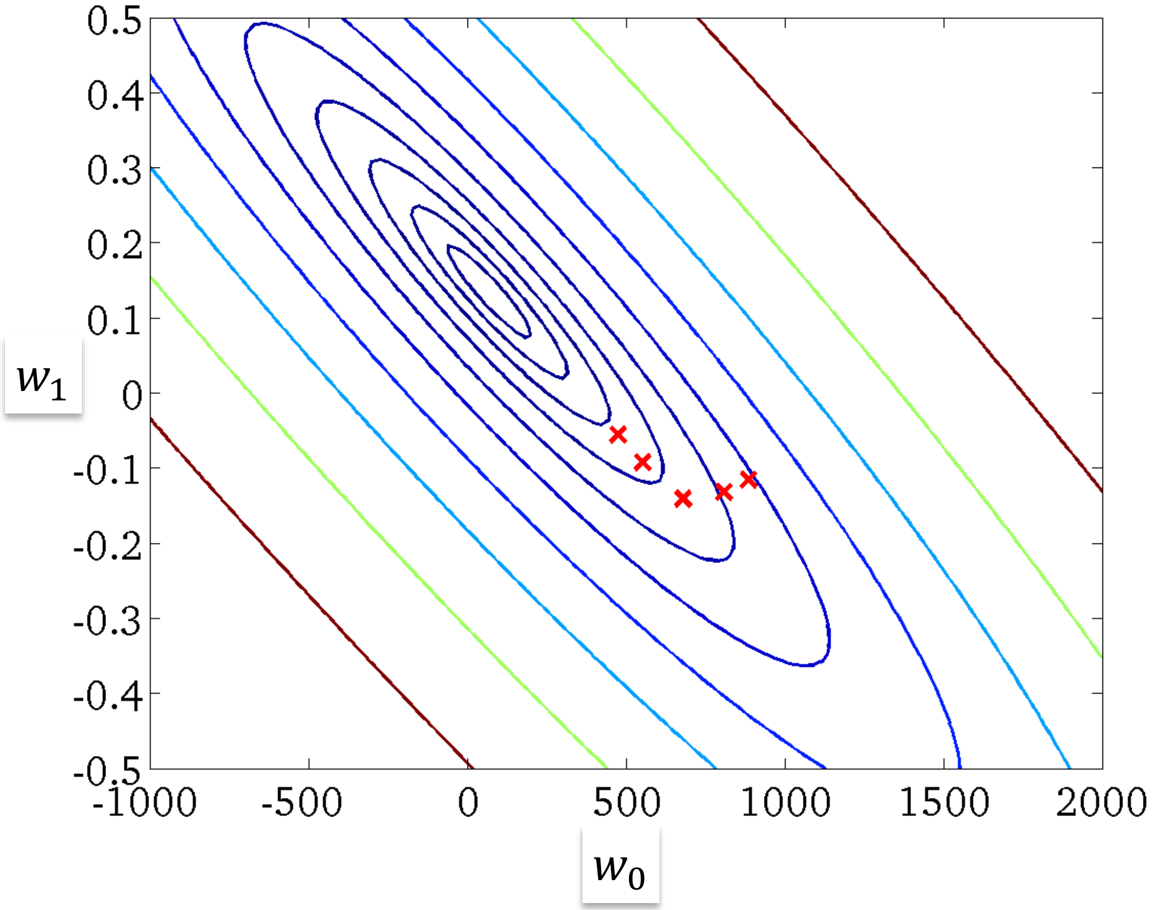
h(x)

(for fixed w_0 , w_1 this is a function of x)



$J(w_0, w_1)$

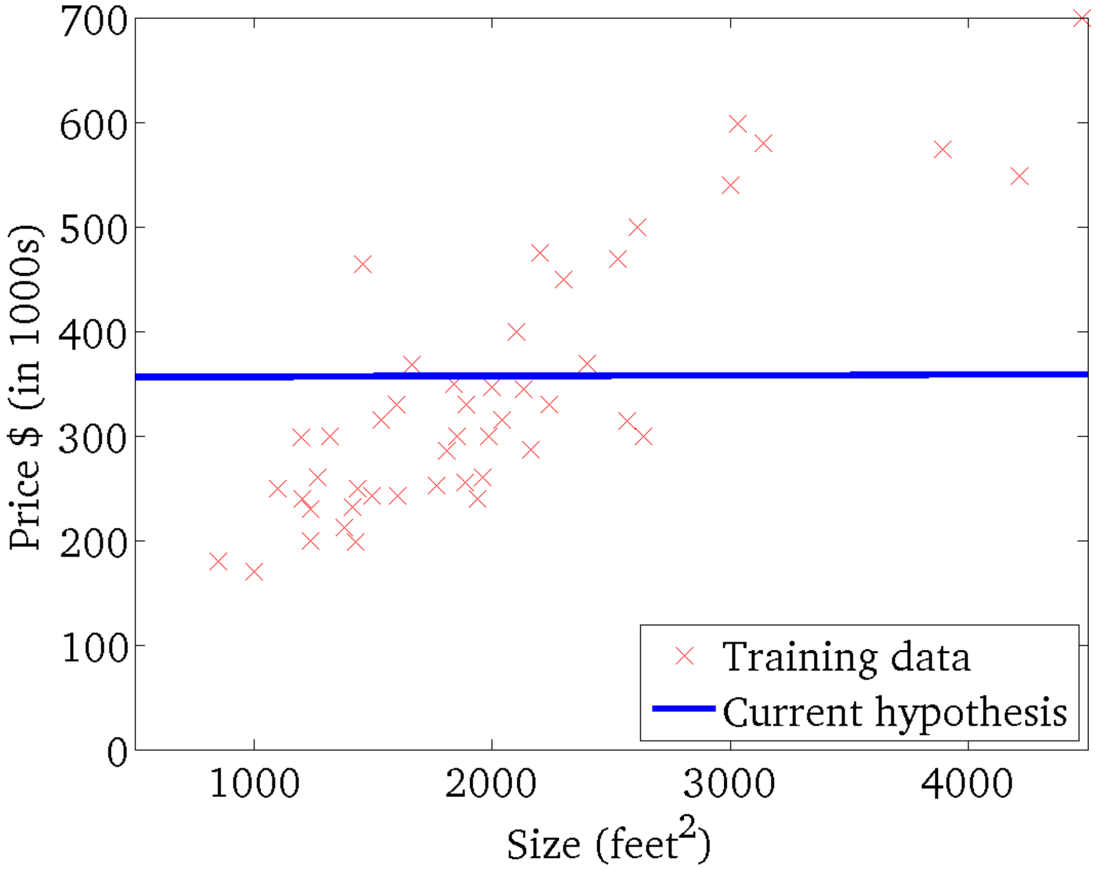
(function of the parameters w_0, w_1)



Slide credit: Andrew Ng

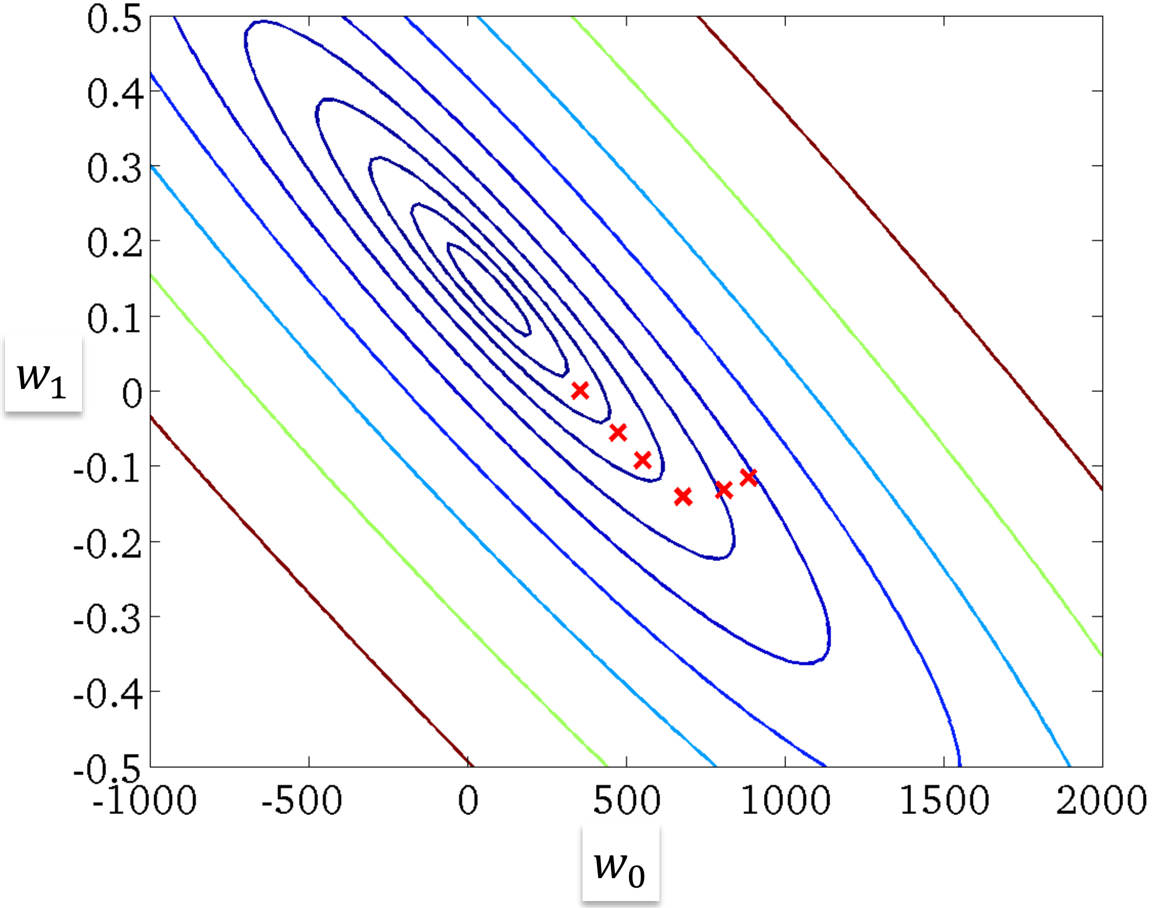
h(x)

(for fixed w_0, w_1 this is a function of x)



$J(w_0, w_1)$

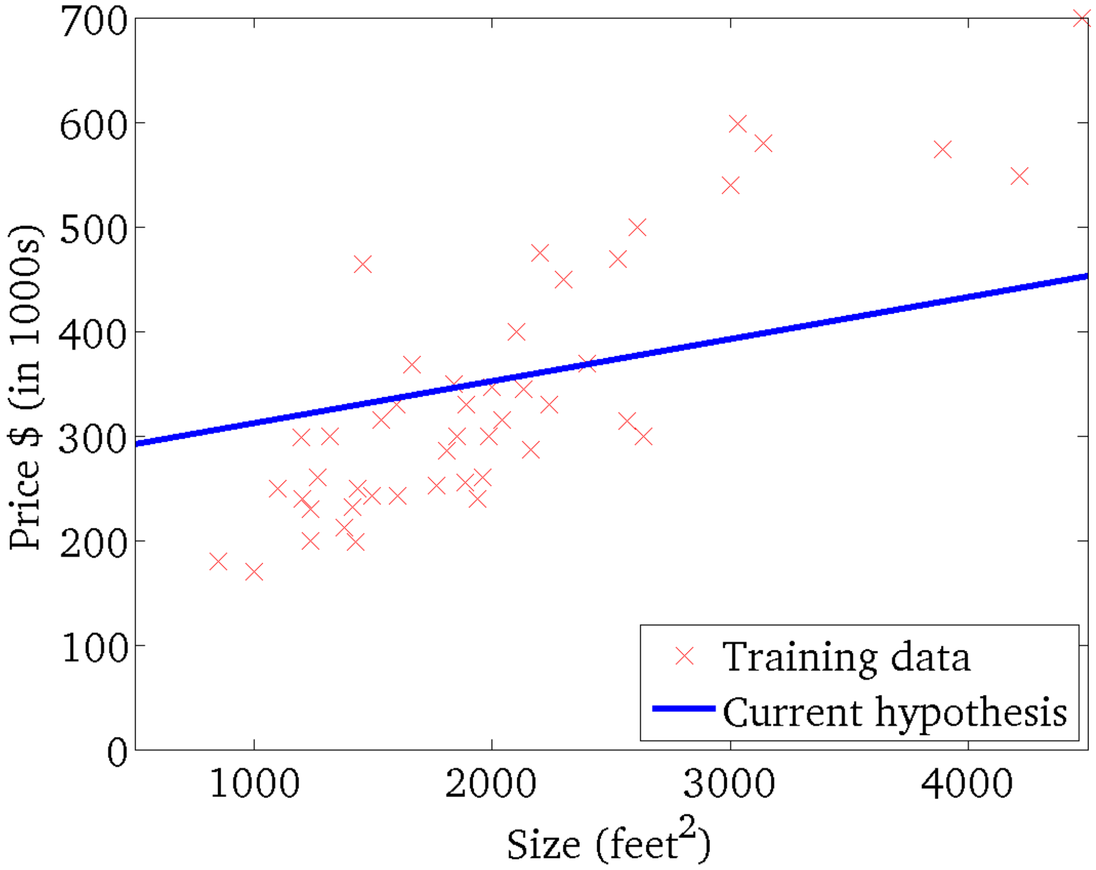
(function of the parameters w_0, w_1)



Slide credit: Andrew Ng

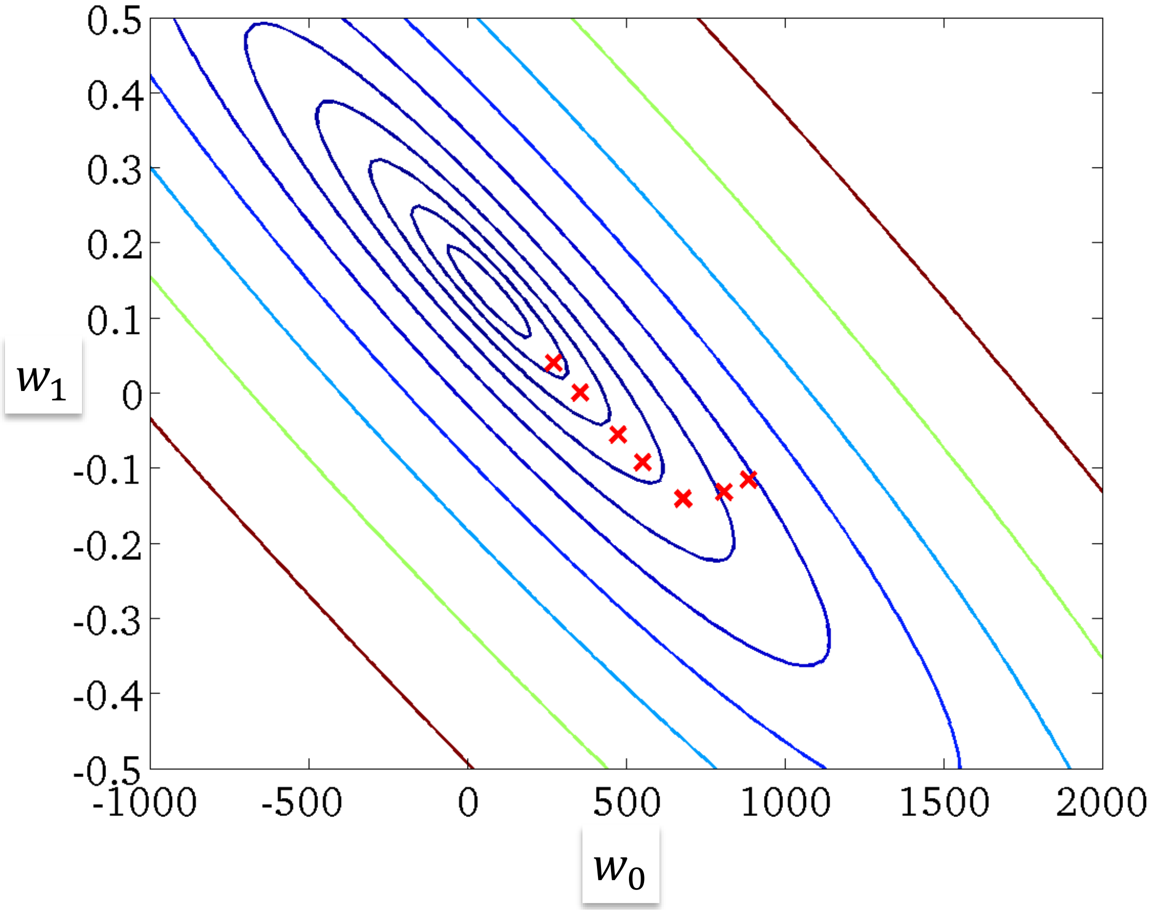
h(x)

(for fixed w_0 , w_1 this is a function of x)



$J(w_0, w_1)$

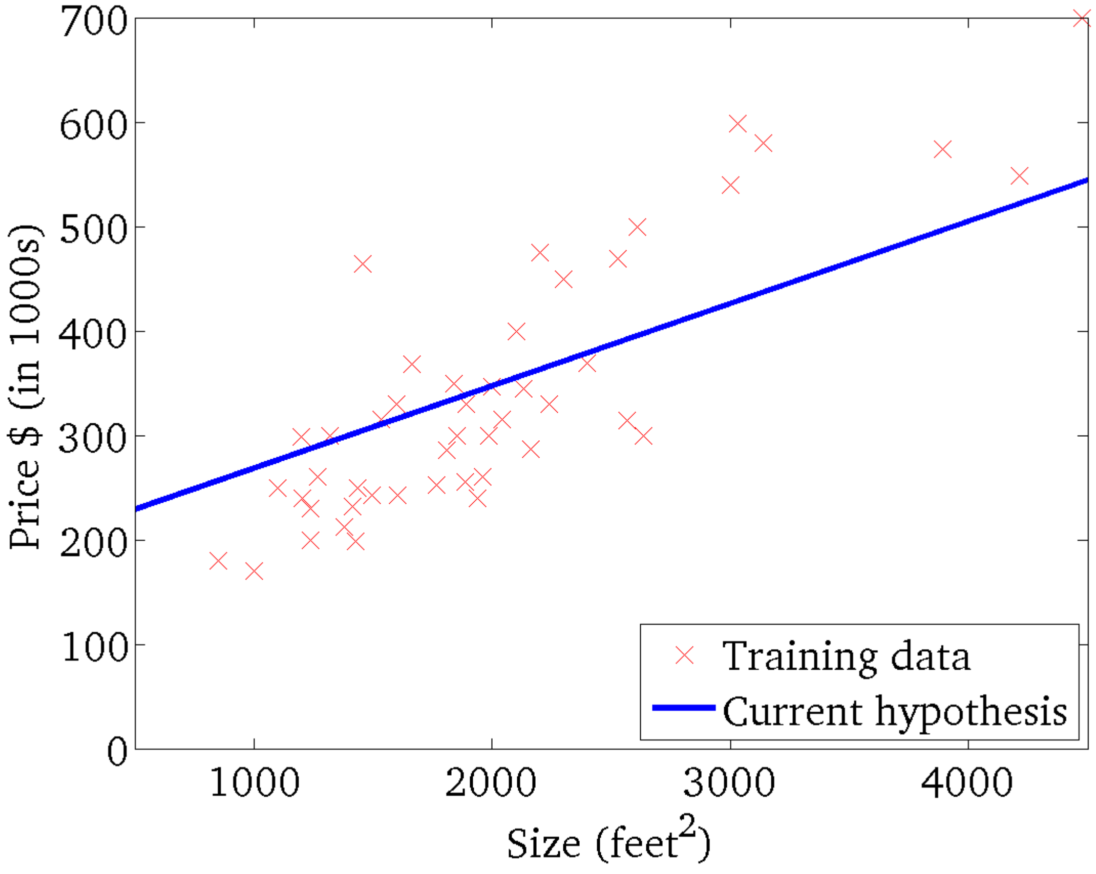
(function of the parameters w_0, w_1)



Slide credit: Andrew Ng

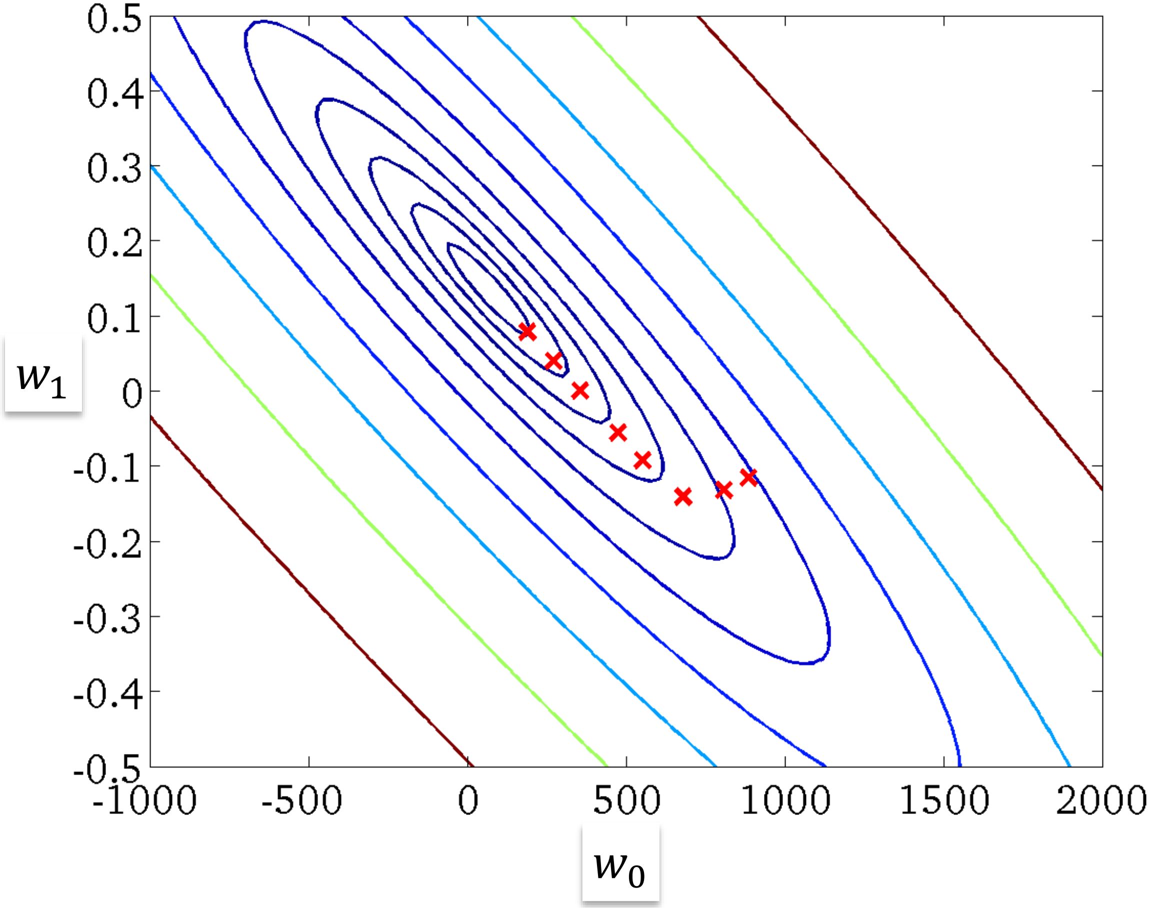
h(x)

(for fixed w_0 , w_1 this is a function of x)



$J(w_0, w_1)$

(function of the parameters w_0, w_1)



TUDelft

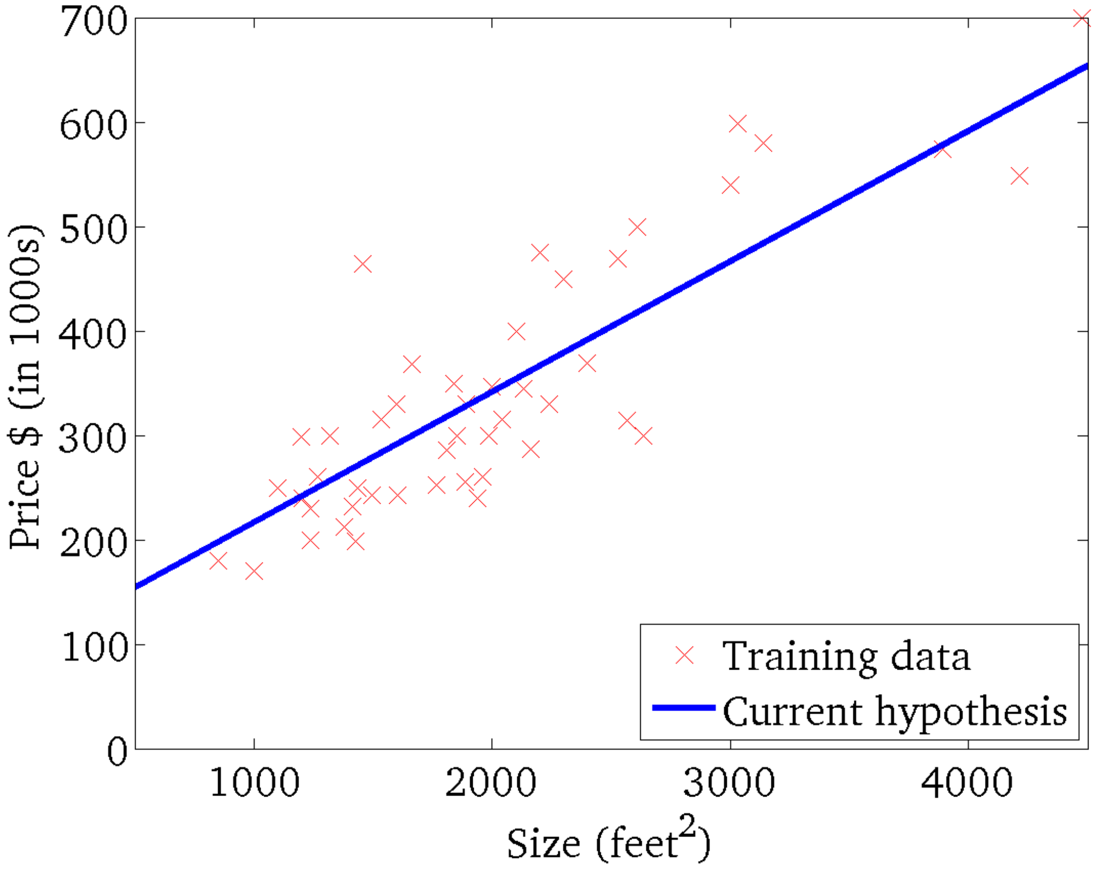
Slide credit: Andrew Ng

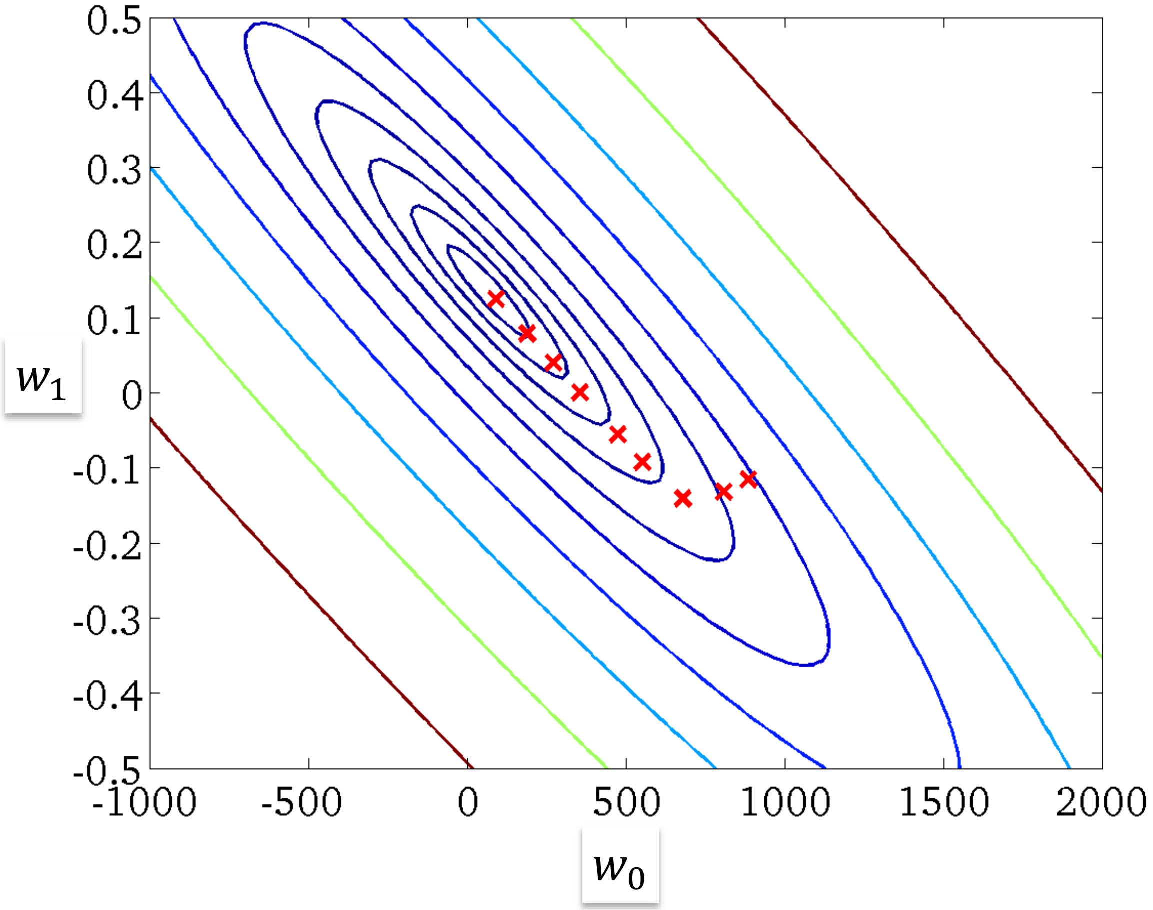
h(x)

(for fixed w_0 , w_1 this is a function of x)

$J(w_0, w_1)$

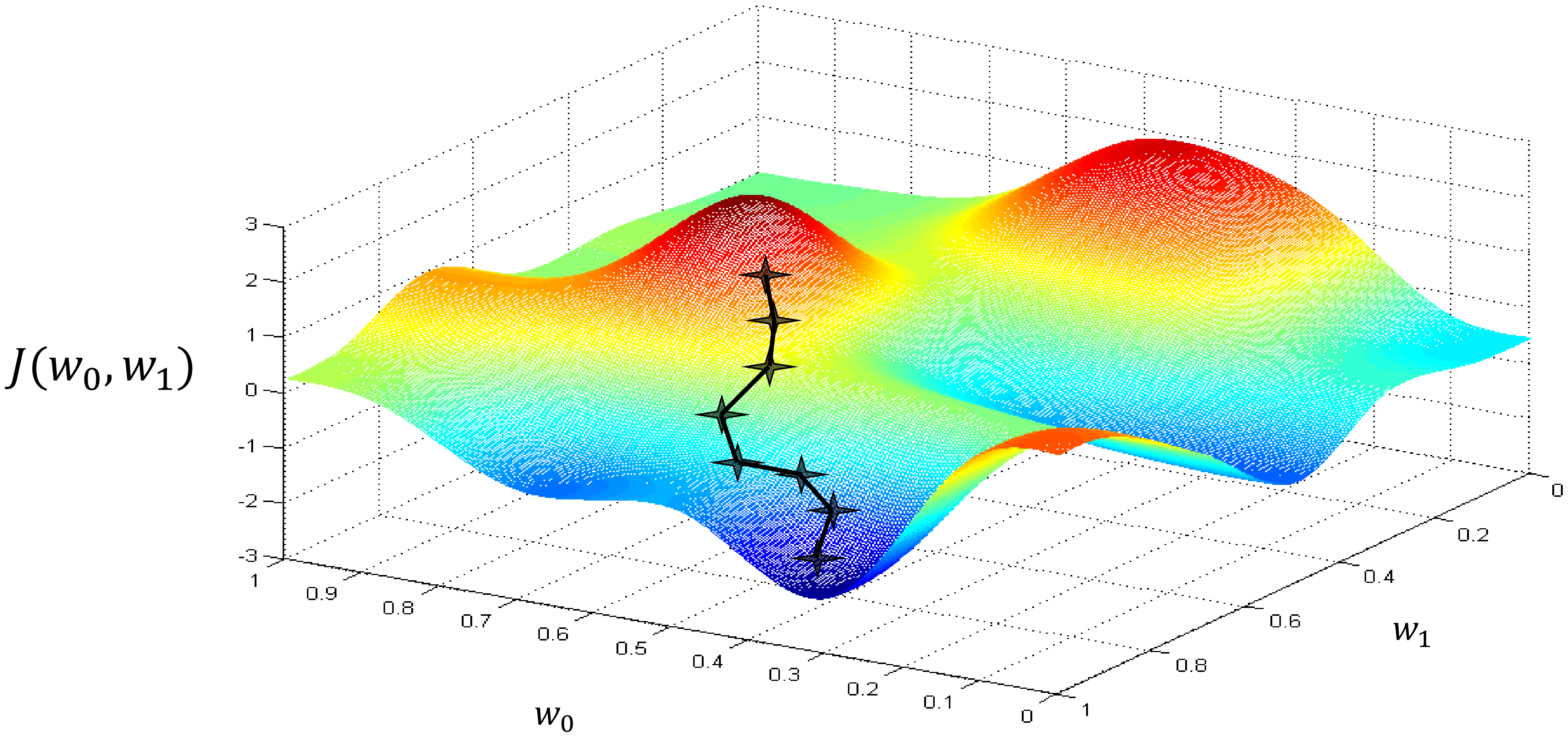
(function of the parameters w_0, w_1)





Slide credit: Andrew Ng

Gradient descent



Gradient descent for univariate linear regression

- Hypothesis: $h(w) = w_1 x_1 + w_0$
- Parameters: w_0 and w_1
- Cost function: $J(w_0, w_1) = \frac{1}{2n} \sum_{i=1} (h(x)^{(i)} - y^{(i)})^2$
- Goal: $\underset{w_0, w_1}{\text{minimize}}\, J(w_0, w_1)$

Gradient descent for univariate linear regression

- $w_0 := w_0 - \alpha \frac{\partial(\frac{1}{2n}\sum_{i=1}((w_1 x_1 + w_0)^{(i)} - y^{(i)})^2)}{\partial w_0}$
- $w_0 := w_0 - \alpha \frac{1}{n}\sum_{i=1}((w_1 x_1 + w_0)^{(i)} - y^{(i)})$

- $w_1 := w_1 - \alpha \frac{\partial(\frac{1}{2n}\sum_{i=1}((w_1 x_1 + w_0)^{(i)} - y^{(i)})^2)}{\partial w_1}$
- $w_1 := w_1 - \alpha \frac{1}{n}\sum_{i=1}((w_1 x_1 + w_0)^{(i)} - y^{(i)})\, x_1^{(i)}$

TU Delft

General idea of gradient descent

- A gradient is a slope of a function
- That is, a set of partial derivatives, one for each dimension (parameter)

$$w_j := w_j - \alpha \frac{\partial J(w)}{\partial w_j}$$

- By following the gradient of a function we can descend to the minimum
- α is a learning rate and controls the speed of descent

Stochastic gradient descent

- We could compute the gradient of cost function for the full dataset before each update
- Instead
  - Compute the gradient of the cost function for a single example
  - Update the weight
  - Move on to the next example

TU Delft

Logistic regression

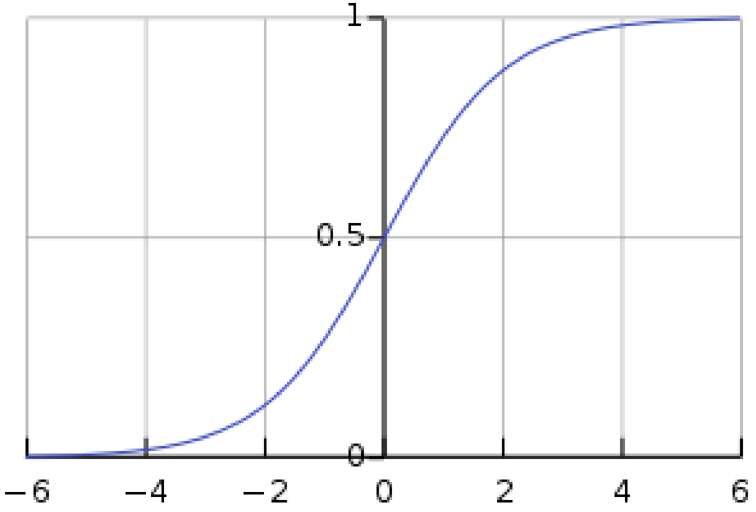
Logistic regression: a taste

- So that you can start with the lab
- More details on Friday

Logistic regression

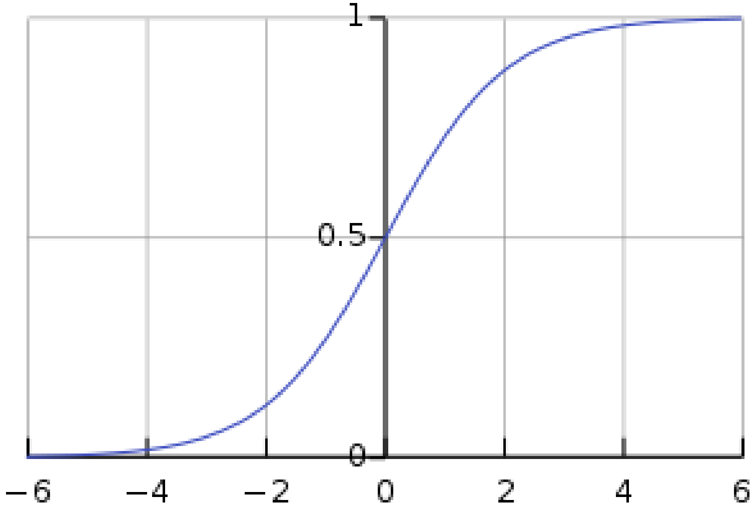
- Let's change the form of linear hypotheses
  $h(x) = w^T x$ to satisfy $0 \leq h(x) \leq 1$

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



- Let's plug $w^T x$ into the logistic function
- $z = w^T x$
- $h(x) = g(w^T x)$

Logistic regression properties



- $\mathrm{h}(x) = \frac{1}{1+e^{(-w^T x)}}$

- $\mathrm{h}(x)$ will give us the probability that our output is 1

- $g(z) \rightarrow 1\ as\ z \rightarrow \infty$
- $g(z) \rightarrow 0\ as\ z \rightarrow -\infty$
- Why are these properties convinient to model a probability?

On Friday

- More on Logistic regression and it’s cost function
- Example how to calculate 1 step of gradient descent for logistic regression
- Support Vector Machine and it’s cost function
- Multi-class classification
- Bias vs. variance