SMU
SINGAPORE MANAGEMENT UNIVERSITY

Linear Regressions and Optimization

Instructor: Steven C.H. Hoi
School of Information Systems
Singapore Management University
Email: chhoi@smu.edu.sg

Outline

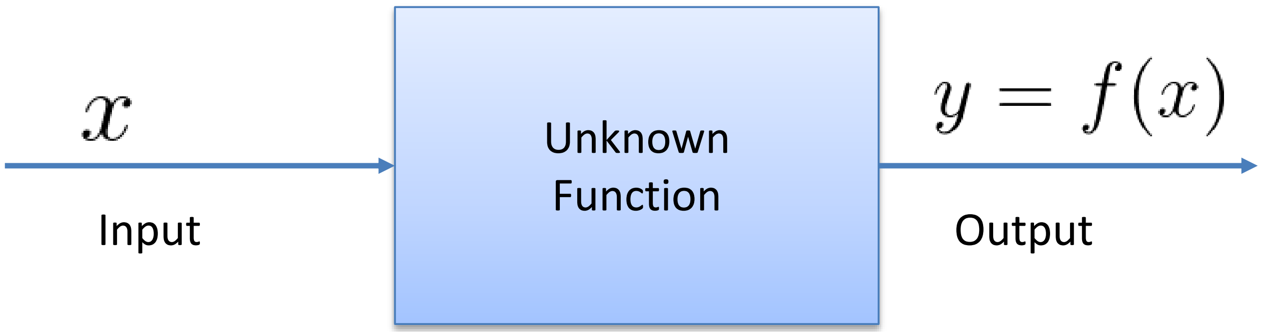
- Supervised Learning
- Linear Regression with One Variable
  - Model Representation
  - Cost Functions
  - Gradient Descent
- Linear Regression with Multiple Variables
  - Learning rate
  - Normal Equation
- Linear Basis Function Models

Acknowledgement: some slides adapted from Andrew Ng & Pedro Domingos

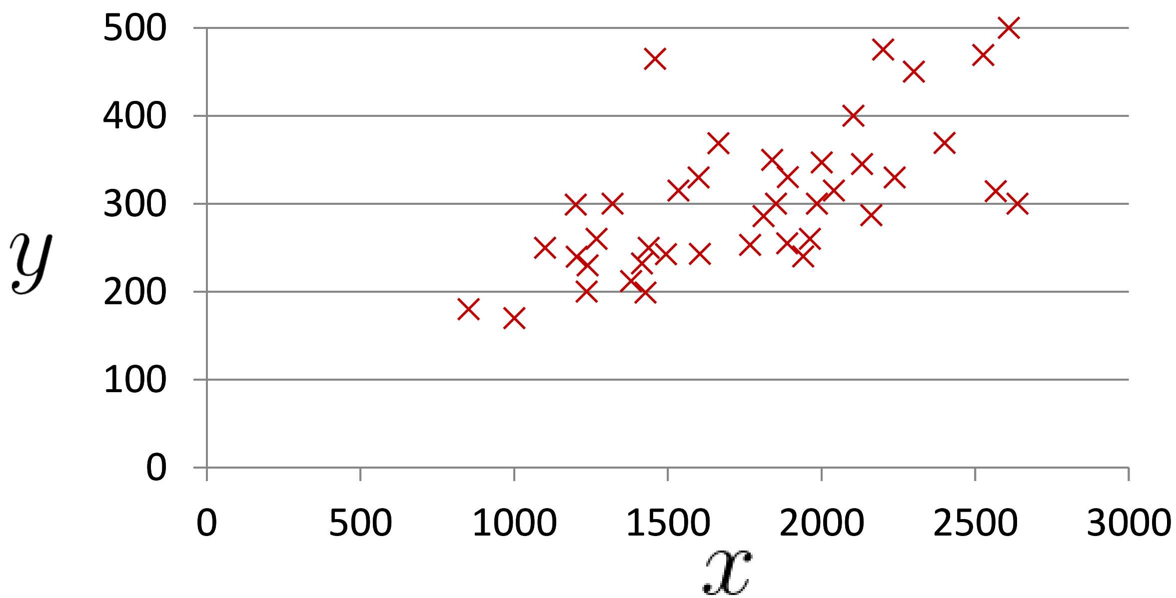
Supervised Learning

- Supervised (Inductive) Learning
- Formalization
  - Input: $\mathbf{x} \in \mathcal{X} \quad \mathbb{R}^n$
  - Output: $y \in \mathcal{Y} \begin{cases} \mathbb{R} & \text{regression} \\ \{+1, -1\} & \text{binary classification} \\ \{1, 2, \ldots, K\} & \text{multi-class classification} \end{cases}$
  - Target function: $f : \mathcal{X} \to \mathcal{Y}$ (unknown)
  - Training Data: $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$
  - Hypothesis: $h : \mathcal{X} \to \mathcal{Y} \qquad h \approx f$
  - Hypothesis space: $h \in \mathcal{H}$

A Learning Problem



$$h(x) \approx f(x)$$



Hypothesis Spaces

- Linear models

$$h(x) = ax + b \approx f(x)$$

  – **Infinite** possible hypotheses!
  – Any choices of coefficient a and b will result in a possible hypothesis

- Polynomial models

$$h(x) = ax^2 + bx + c \approx f(x)$$

- Any nonlinear models

$$h(x) = g(x) \approx f(x)$$

Two Views of Learning

- **Learning is the removal of our remaining uncertainty.**
  - If we are know that x and y are linearly dependent, then we could use the training data to infer the linear function
- **Learning requires guessing a good, small hypothesis class.**
  - We could start with a very small / simple class, and enlarge it until it contains a hypothesis that fits the data
- **But we could be wrong**
  - Our prior knowledge might be wrong
  - Our guess of the hypothesis class could be wrong
    - The smaller the hypothesis class, the more likely we are wrong

Two Strategies for Machine Learning

- **Develop Languages for Expressing Prior Knowledge**
  - Rule grammars and stochastic models
- **Develop Flexible Hypothesis Spaces**
  - Nested collections of hypotheses, rules, linear models, decision trees, neural networks, etc
- **For either case, the key is to**
  - Developing efficient algorithms for finding a Hypothesis that best approximates the target function for fitting the data

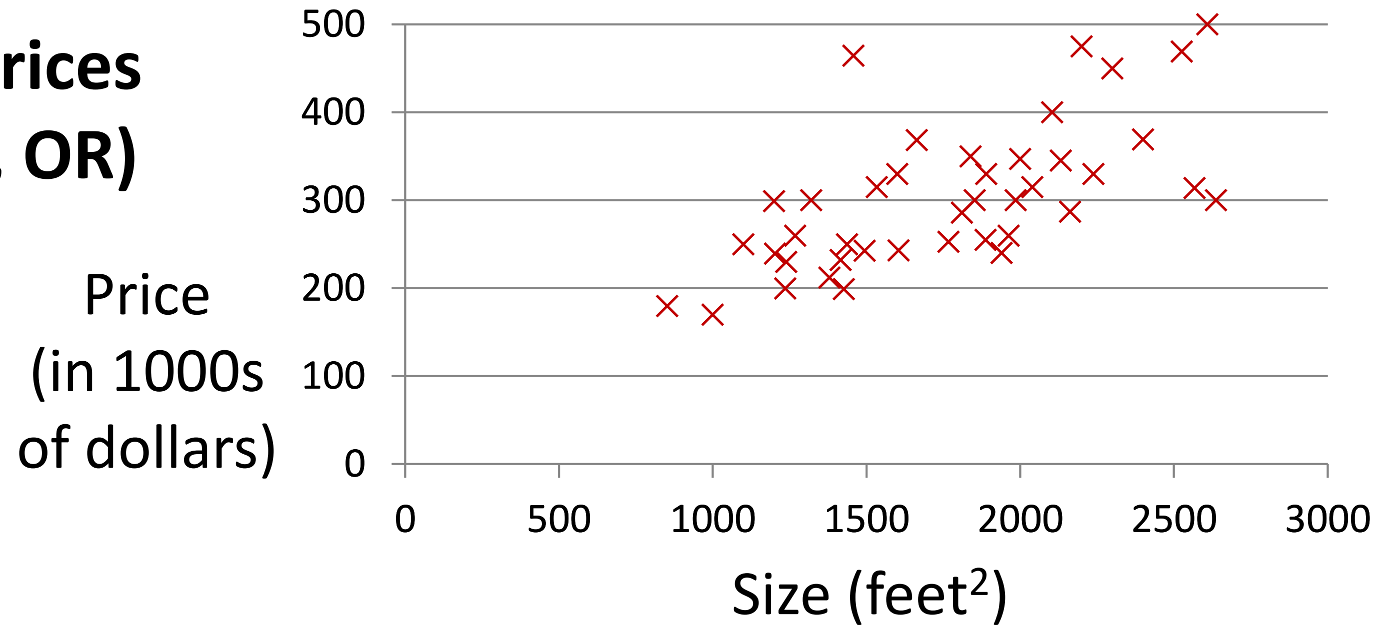
Key Issues in Machine Learning

- What are good hypothesis spaces?
  - Which spaces have been useful in practical applications and why?
- What algorithms can work with these spaces?
  - Are there general design principles for machine learning algorithms?
- How can we find the best hypothesis in an efficient way?
  - How to find the optimal solution efficiently ("optimization" question)
- How can we optimize accuracy on future data?
  - Known as the "overfitting" problem (i.e., "generalization" theory)
- How can we have confidence in the results?
  - How much training data is required to find accurate hypothesis? ("statistical" question)
- Are some learning problems computationally intractable? ("computational" question)
- How can we formulate application problems as machine learning problems? ("engineering" question)

Linear Regression with One Variable

Regression with One Variable

Housing Prices (Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Regression with One Variable

Training set of housing prices (Portland, OR)

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

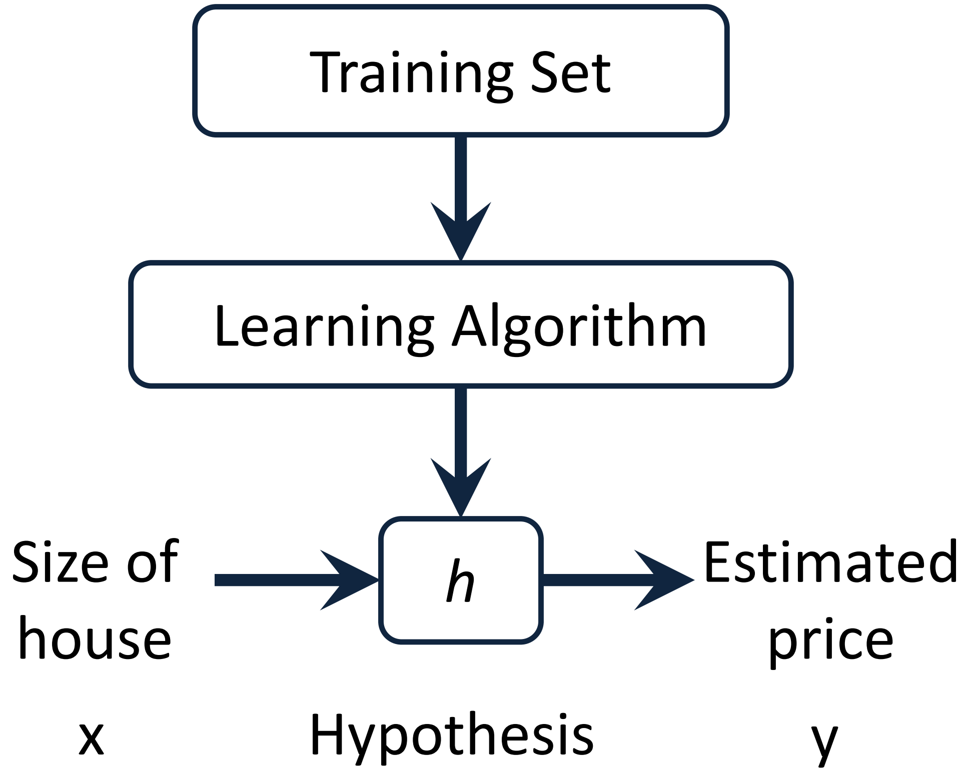
Notation:

m = Number of training examples

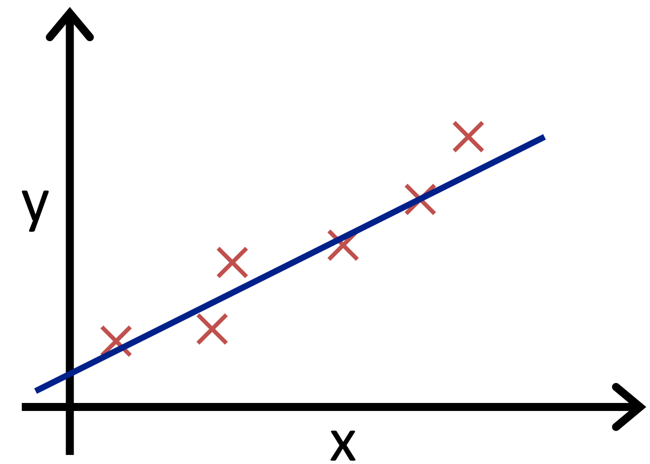
x’s = “input” variable / features

y’s = “output” variable / “target” variable

Model Representation



How do we represent *h* ?



$$h(x) = w_0 + w_1 x$$

Linear regression with one variable.
"**Univariate Linear Regression**"

How to choose parameters w_0, w_1 ?

Formulation: Cost Function

Hypothesis:

$$h(x) = w_0 + w_1 x$$

Parameters:

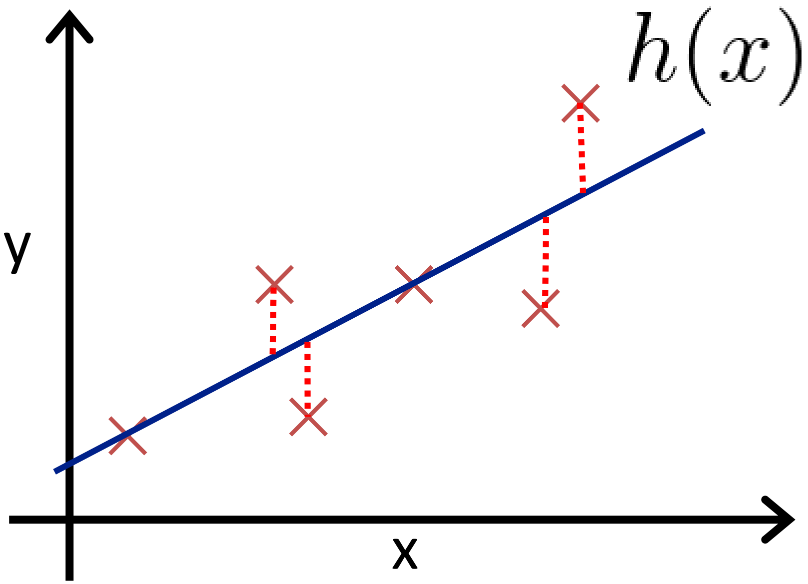
$$w_0, w_1$$

Cost Function: mean squared error (MSE)

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

Goal:

$$\min_{w_0, w_1} J(w_0, w_1)$$



Formulation: Cost Function

Simplified

Hypothesis:

$$h(x) = w_0 + w_1 x$$

$$h(x) = w_1 x$$

Parameters:

$$w_0, w_1$$

$$w_1$$

Cost Function:

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

$$J(w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

Goal:

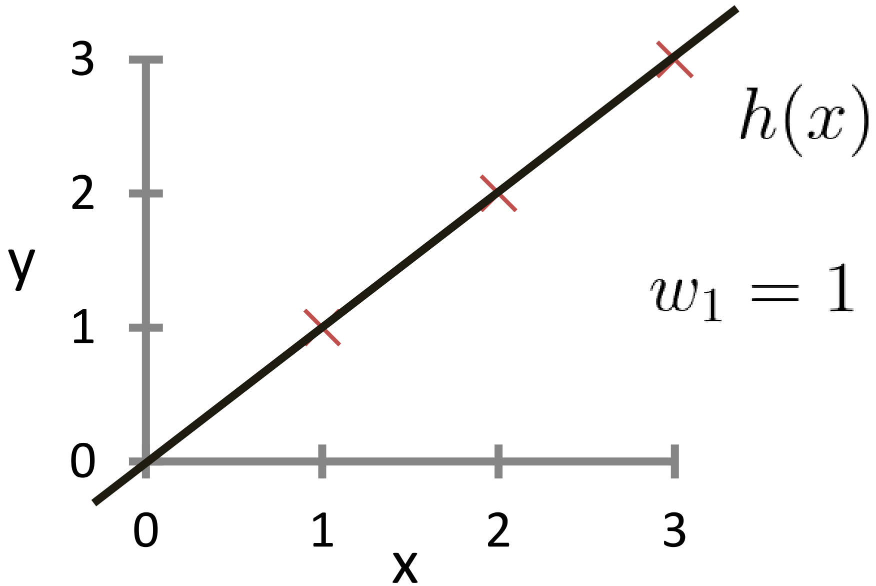
$$\min_{w_0, w_1} J(w_0, w_1)$$

$$\min_{w_1} J(w_1)$$

Cost Function: Example

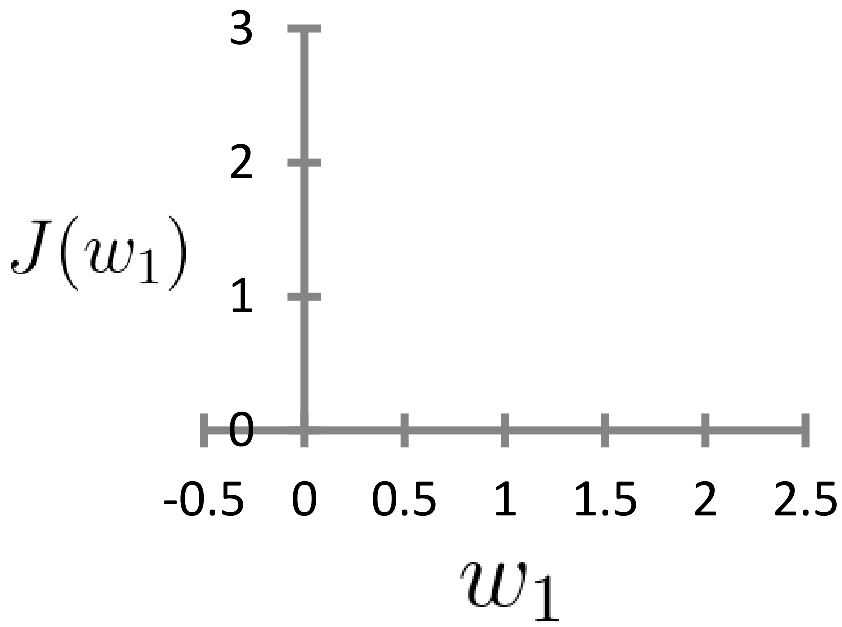
$h(x)$

For fix w_1 this is a function of x



$J(w_1)$

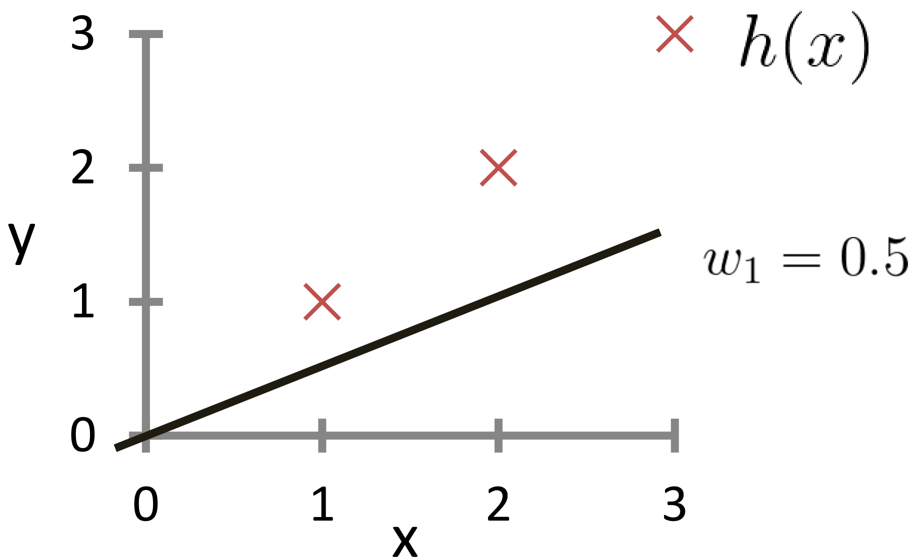
function of the parameter w_1



Cost Function: Example

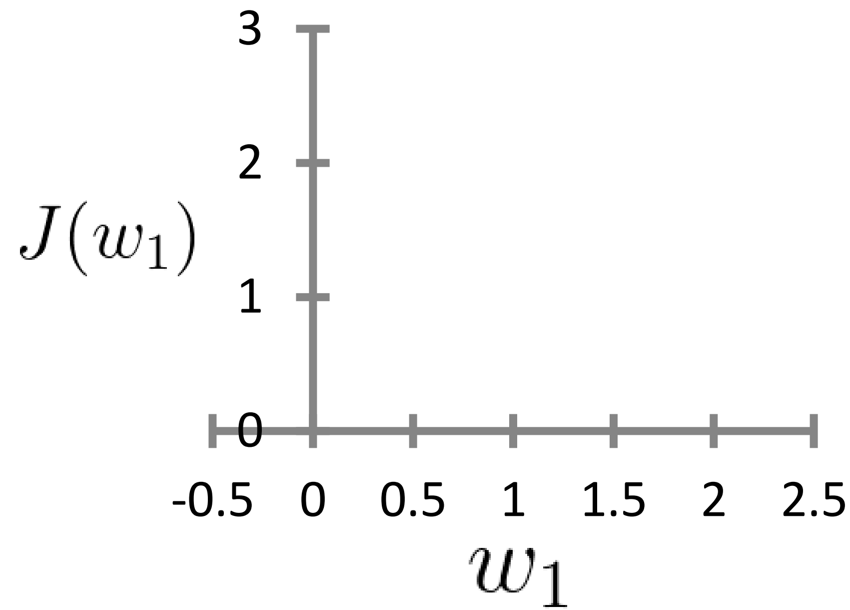
$h(x)$

For fix w_1 this is a function of x



$J(w_1)$

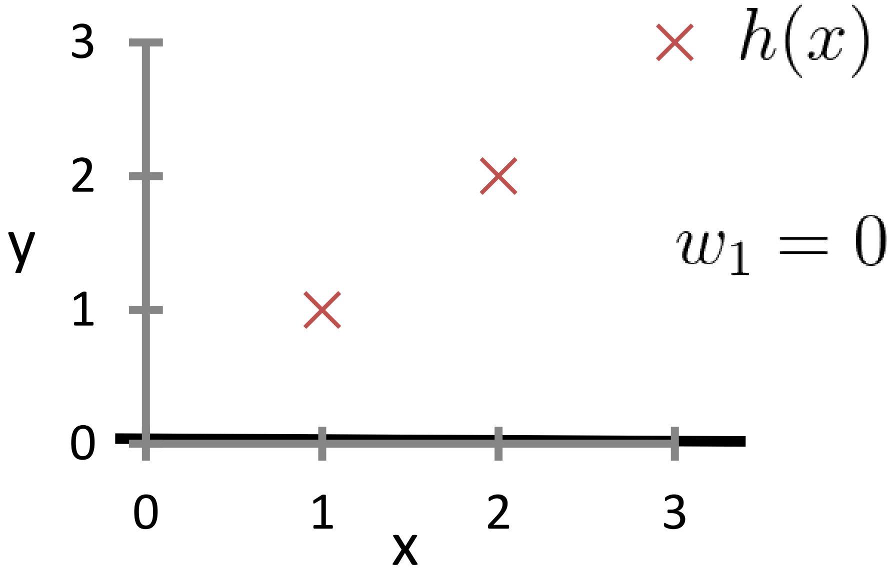
function of the parameter w_1



Cost Function: Example

$h(x)$

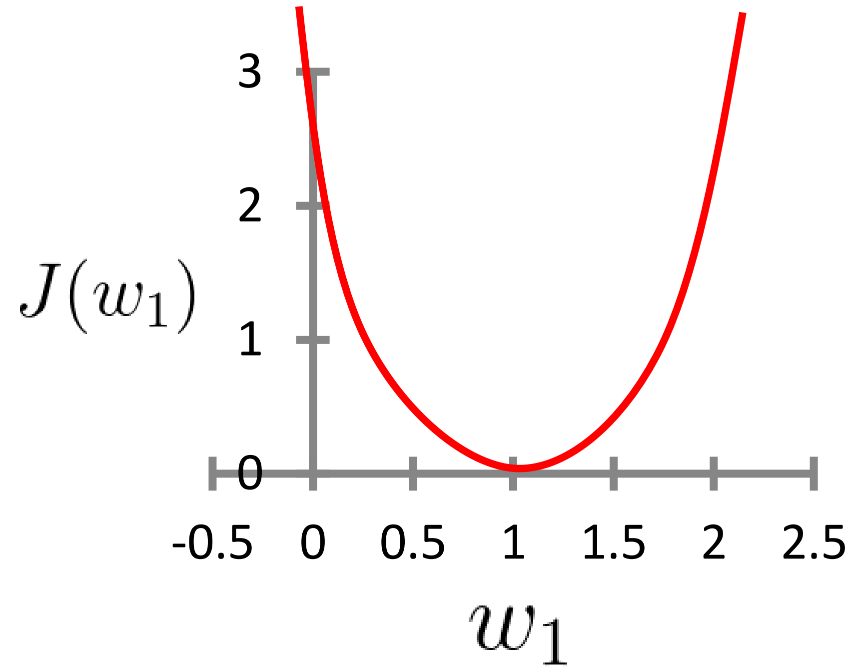
For fix w_1 this is a function of x



$w_1 = 0$

$J(w_1)$

function of the parameter w_1



Cost Function

Hypothesis: $h(x) = w_0 + w_1 x$

Parameters: w_0, w_1

Cost Function: $J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x_i) - y_i\right)^2$

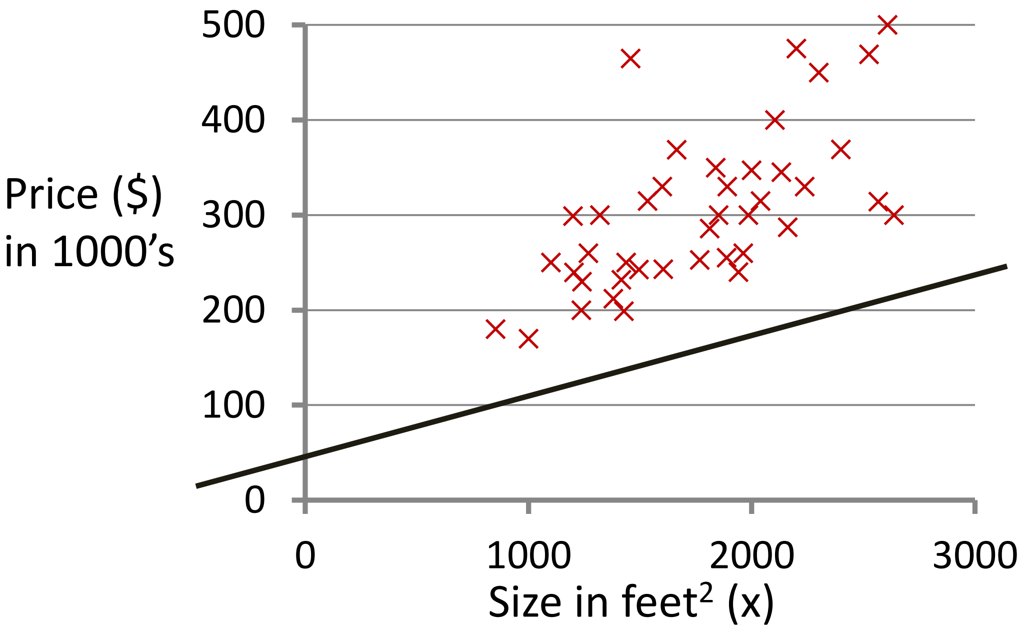
Goal:

$$\min_{w_0, w_1} J(w_0, w_1)$$

Cost Function

$$h(x)$$

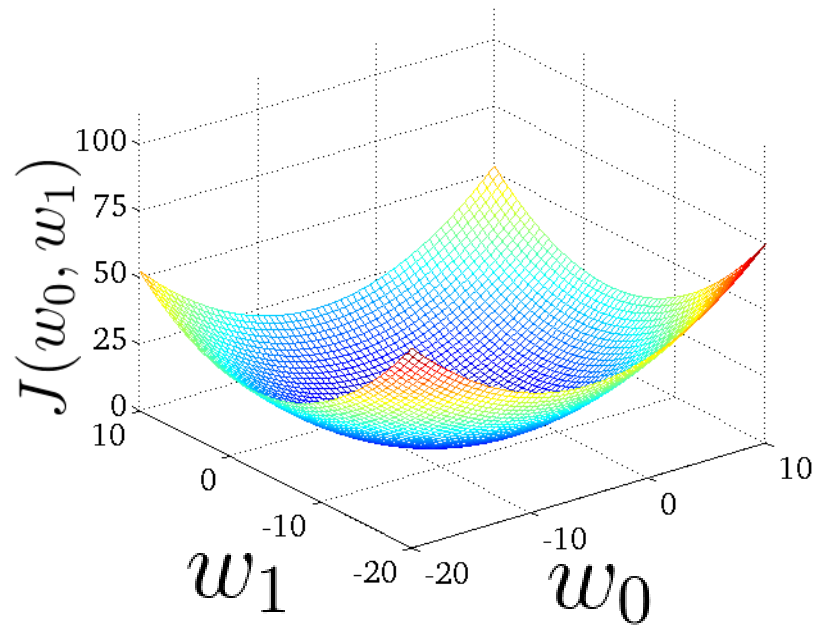
(for fixed w_0, w_1, this is a function of x)



$$h(x) = 50 + 0.06x$$

$$J(w_0, w_1)$$

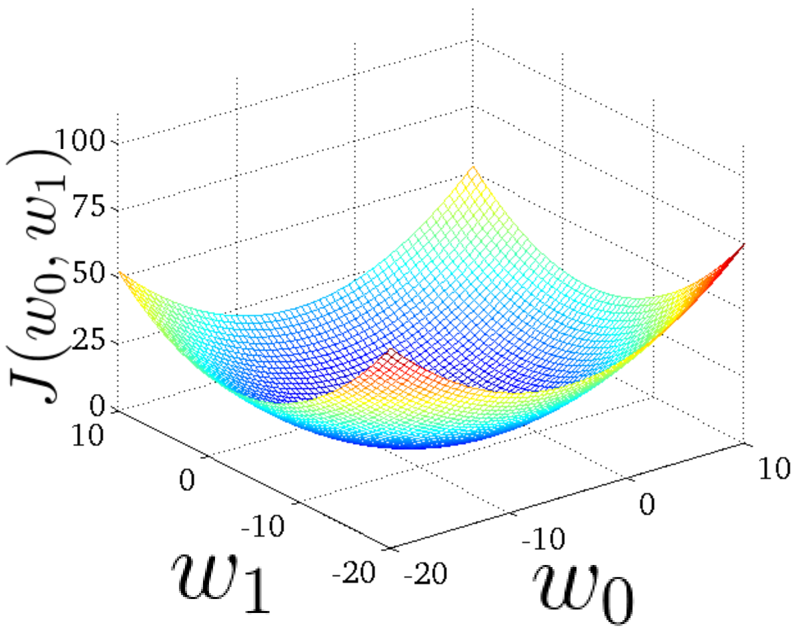
(function of the parameters w_0, w_1)

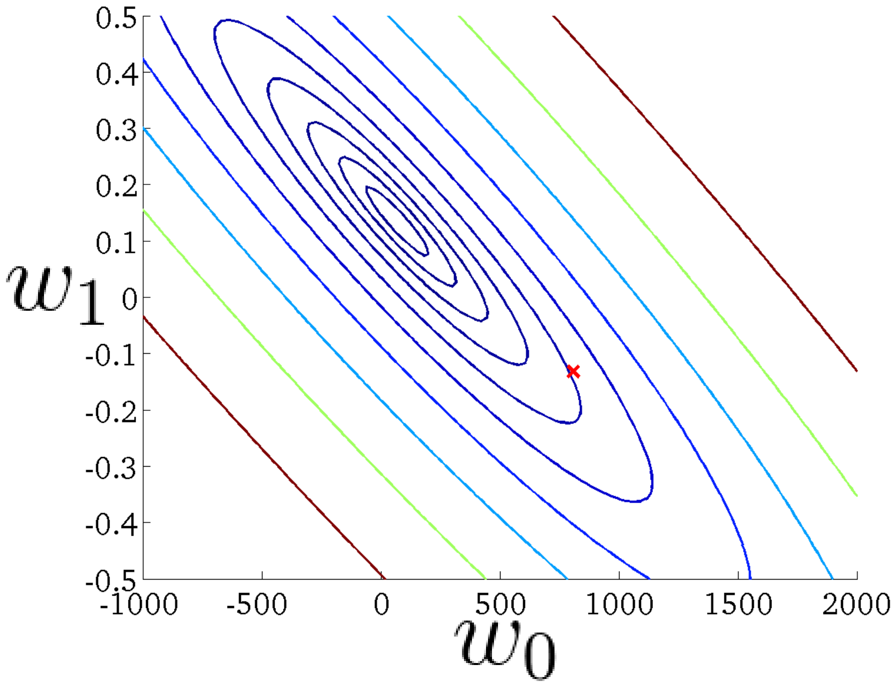


Cost Function

- Contour plots

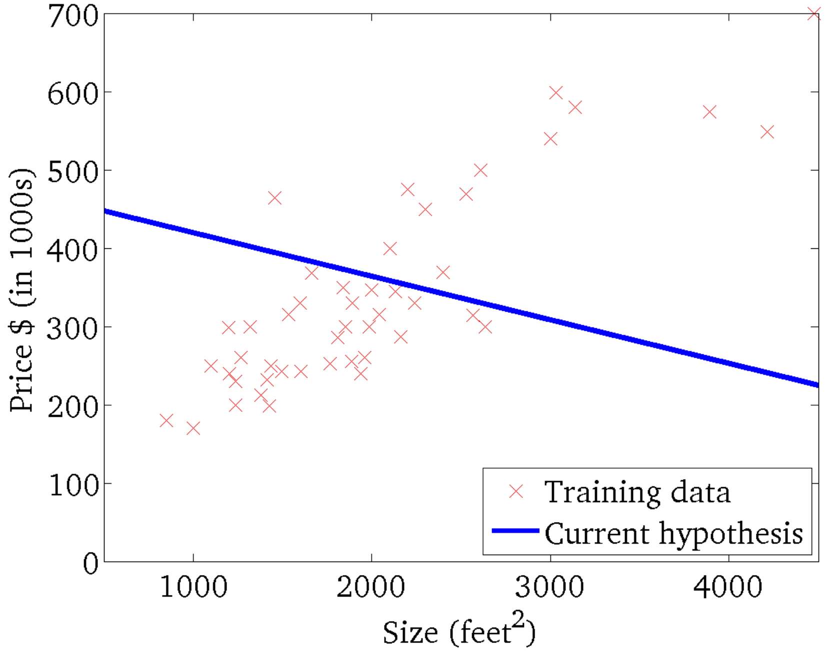
$$J(w_0, w_1)$$





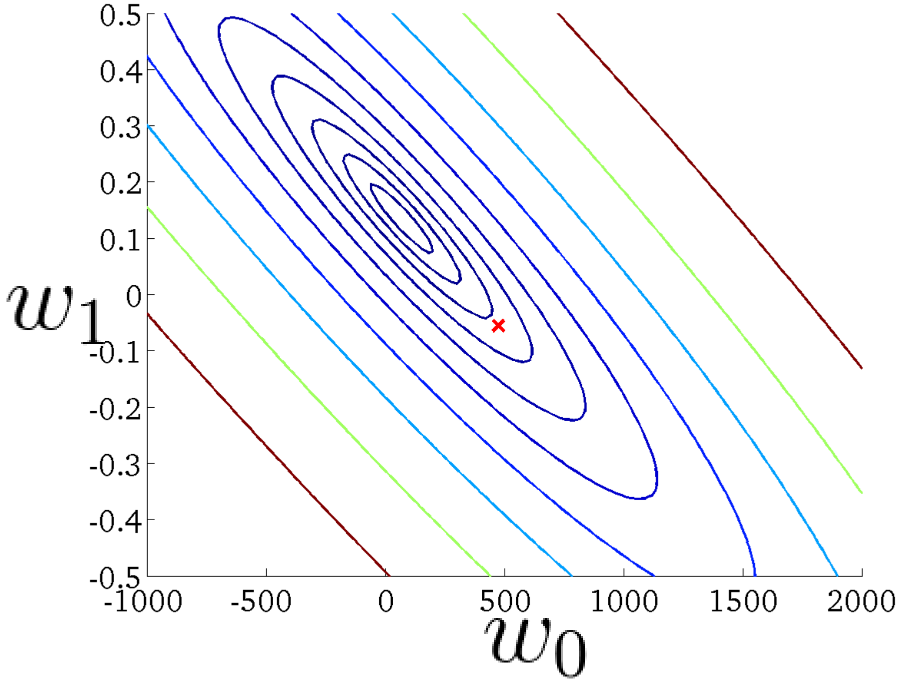
$$h(x)$$

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



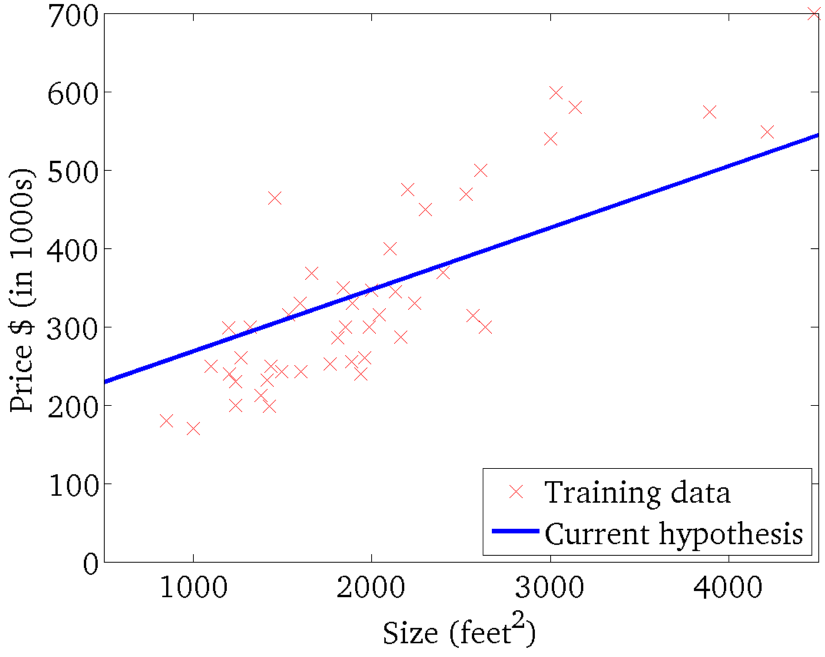
$$J(w_0, w_1)$$

(function of the parameters w_0, w_1)



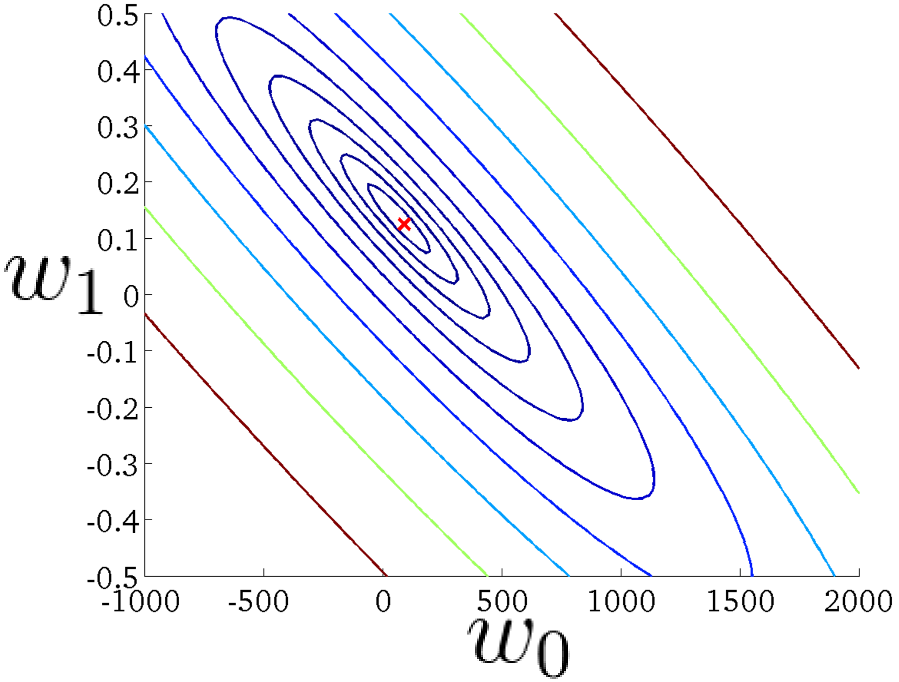
$h(x)$

(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)



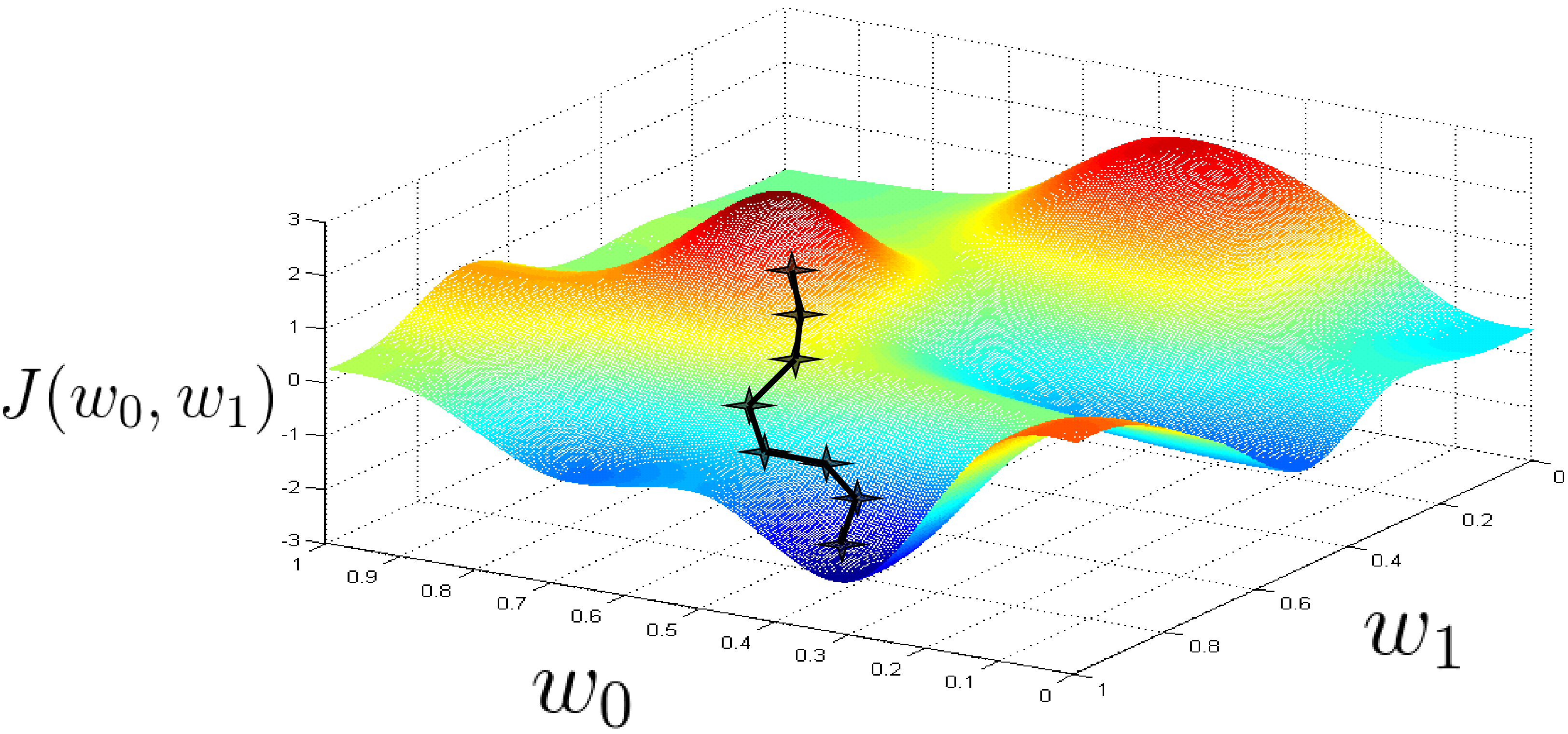
Gradient Descent for Optimization

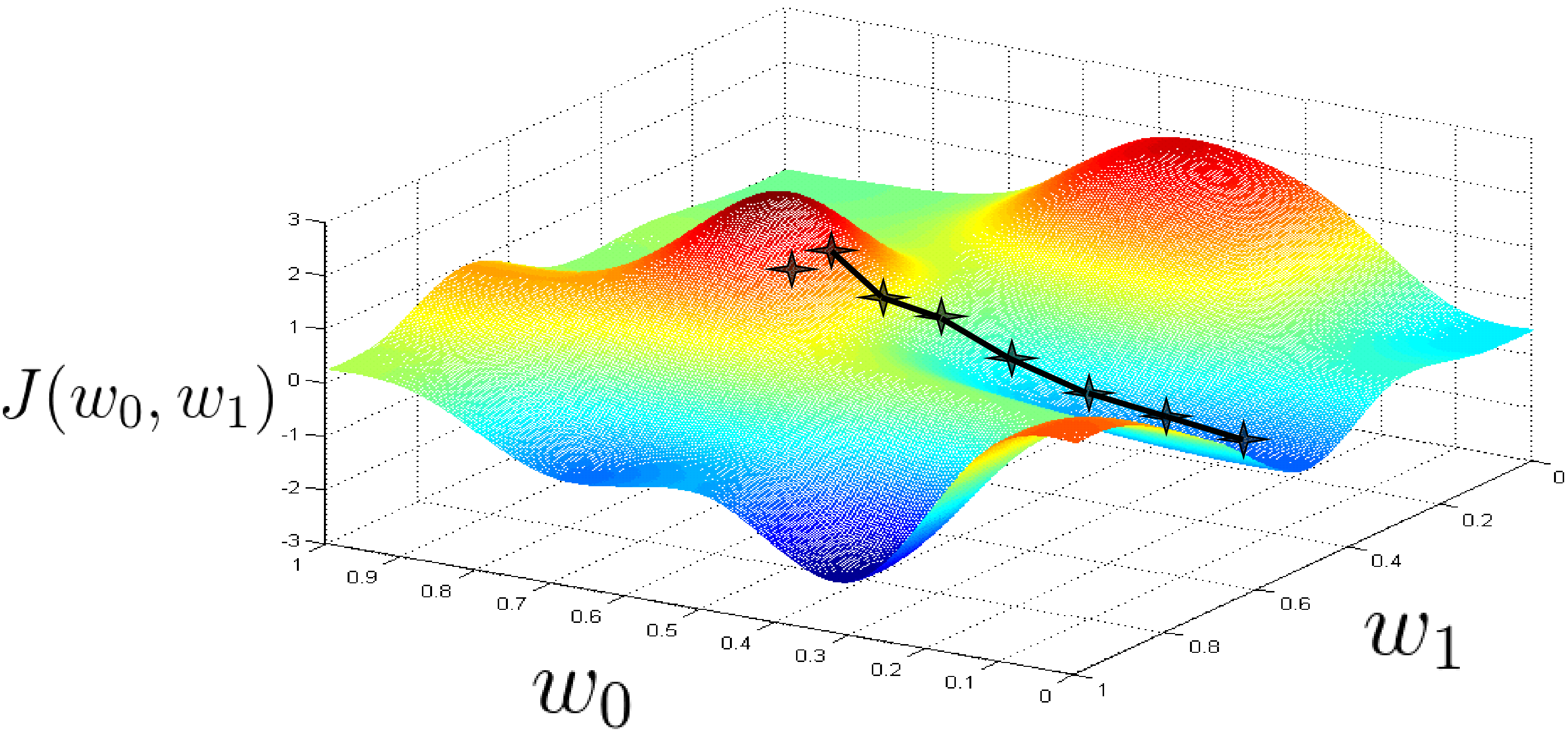
Given some objective function $J(w_0, w_1)$

Want to optimize $\min_{w_0, w_1} J(w_0, w_1)$

Outline:

- Start with some w_0, w_1
- Keep changing w_0, w_1 to reduce $J(w_0, w_1)$ until we hopefully end up at a minimum


$J(w_0, w_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
w_0
0
0.2
0.4
0.6
0.8
1
w_1


$J(w_0, w_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
w_0
0
0.2
0.4
0.6
0.8
1
w_1

Gradient descent algorithm

$$\text{initialize} \quad w_j \quad j = 0, 1$$

$$\text{repeat until convergence } \{$$

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1) \quad \text{(simultaneously update } j = 0 \text{ and } j = 1\text{)}$$

$$\}$$

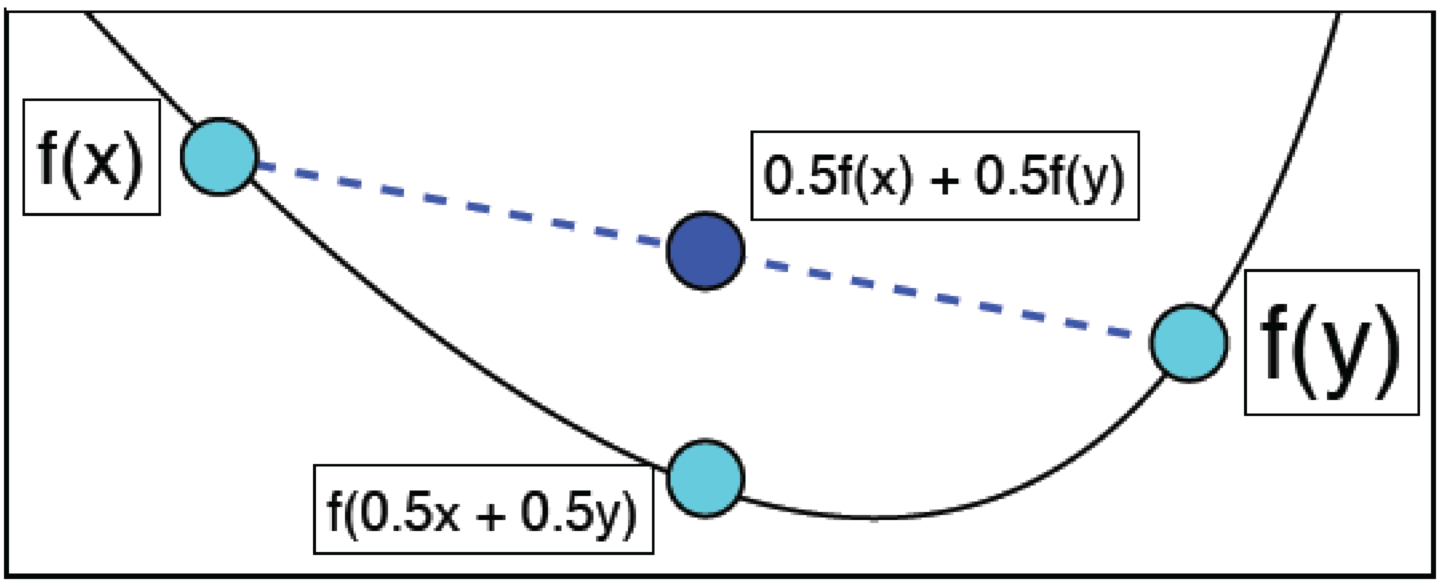
learning rate parameter
(rule of thumb: 0.1)

Convex Function

- A real-valued function f is **convex** if

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \quad \forall 0 \leq \theta \leq 1$$

- Function is **below** a linear interpolation from x to y.
- The negative of a convex function is a **concave** function
- **Convex**: Implies that all local minima are global minima.



Convex Function: Examples

- Some simple convex functions

$$f(\mathbf{x}) = c$$
$$f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x}$$
$$f(\mathbf{x}) = a\mathbf{x}^2 + b \ (for\ a > 0)$$
$$f(\mathbf{x}) = \exp(a\mathbf{x})$$
$$f(\mathbf{x}) = \mathbf{x} \log \mathbf{x} \ (for\ \mathbf{x} > 0)$$
$$f(\mathbf{x}) = \|x\|^2$$
$$f(\mathbf{x}) = \max_i \{x_i\}$$

- Some other notable examples

$$f(x, y) = \log(e^x + e^y)$$
$$f(X) = \log \det X \ \text{(for X is positive-definite)}$$
$$f(x, Y) = x^\top Y^{-1} x \ \text{(for Y is positive-definite)}$$

Operations that Preserve Convexity

- Non-negative weighted sum:

$$f(x) = \theta_1 f_1(x) + \theta_2 f_2(x) \qquad \theta_1 \geq 0, \theta_2 \geq 0$$

- Composition with affine mapping:

$$g(x) = f(Ax + b)$$

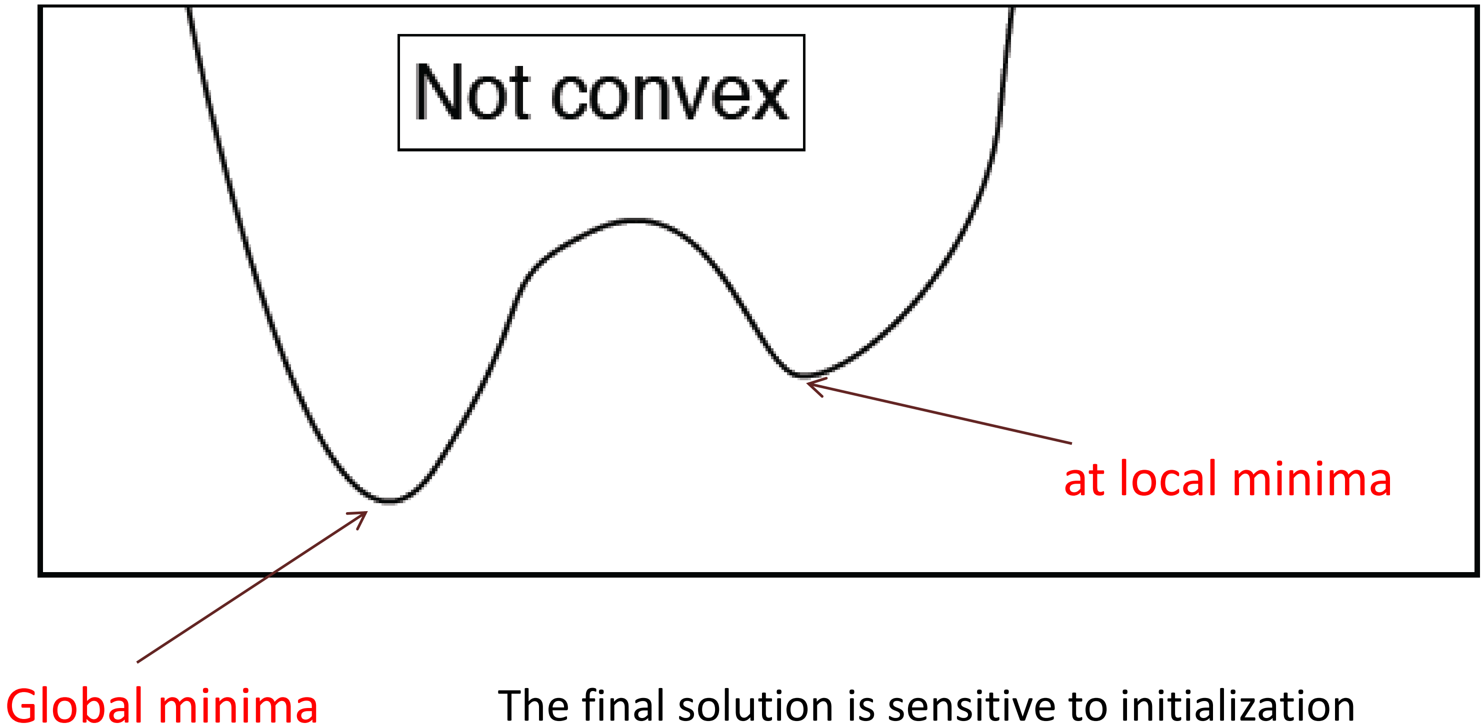
- Pointwise maximum:

$$f(x) = \max_i f_i(x)$$

- More about convex optimization
  - Online Convex Optimization book: (Stephen Boyd)
  - https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

Non-Convex Function

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



The final solution is sensitive to initialization

Gradient Descent for Linear Regression

Gradient descent algorithm

$$\text{repeat until convergence } \{$$
$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1)$$
$$(\text{for } j = 1 \text{ and } j = 0)$$
$$\}$$

Linear Regression Model

$$h(x) = w_0 + w_1 x$$

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x_i) - y_i\right)^2$$

$$\frac{\partial}{\partial w_j} J(w_0, w_1) = \frac{\partial}{\partial w_j} \left(\frac{1}{2m} \sum_{i=1}^{m} \left((w_0 + w_1 x_i) - y_i\right)^2 \right)$$

$$\frac{\partial}{\partial w_0} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)$$

$$\frac{\partial}{\partial w_1} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i) \cdot x_i$$

Gradient descent algorithm

repeat until convergence {

$$w_0 := w_0 - \alpha \boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h(x_i) - y_i\right)} \quad \frac{\partial}{\partial w_0} J(w_0, w_1)$$

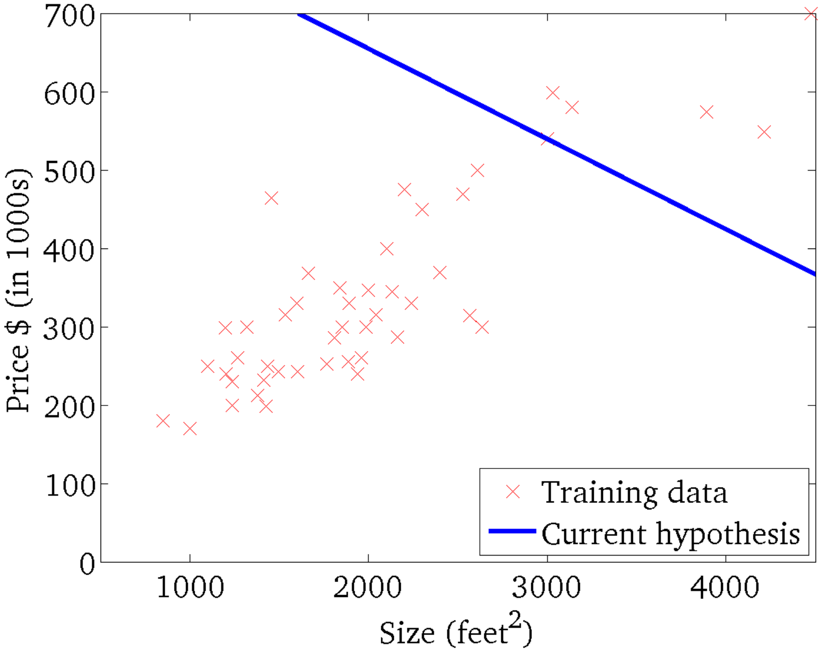
$$w_1 := w_1 - \alpha \boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h(x_i) - y_i\right) \cdot x_i} \quad \frac{\partial}{\partial w_1} J(w_0, w_1)$$

}

update w_0 and w_1 simultaneously

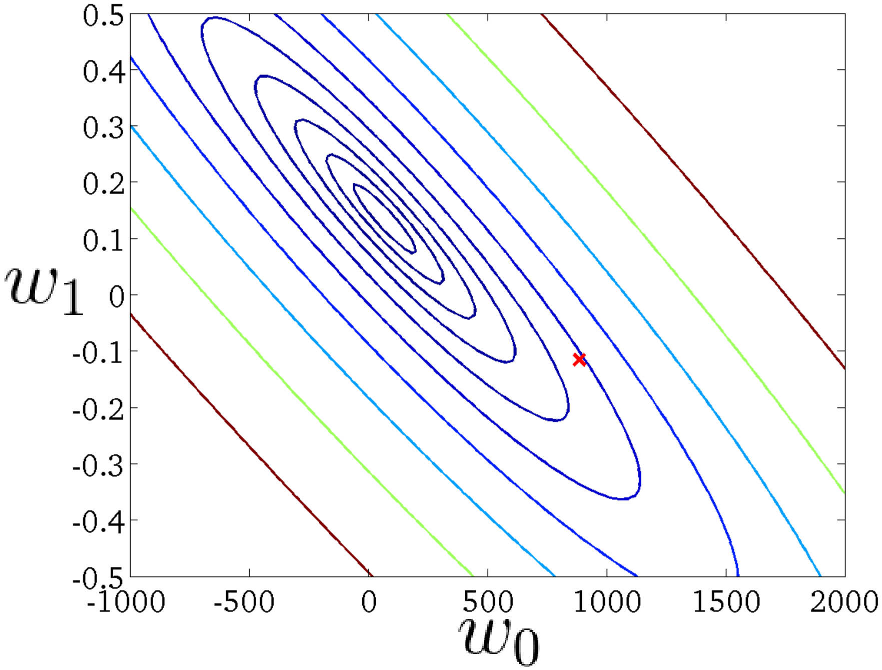
$$h(x)$$

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



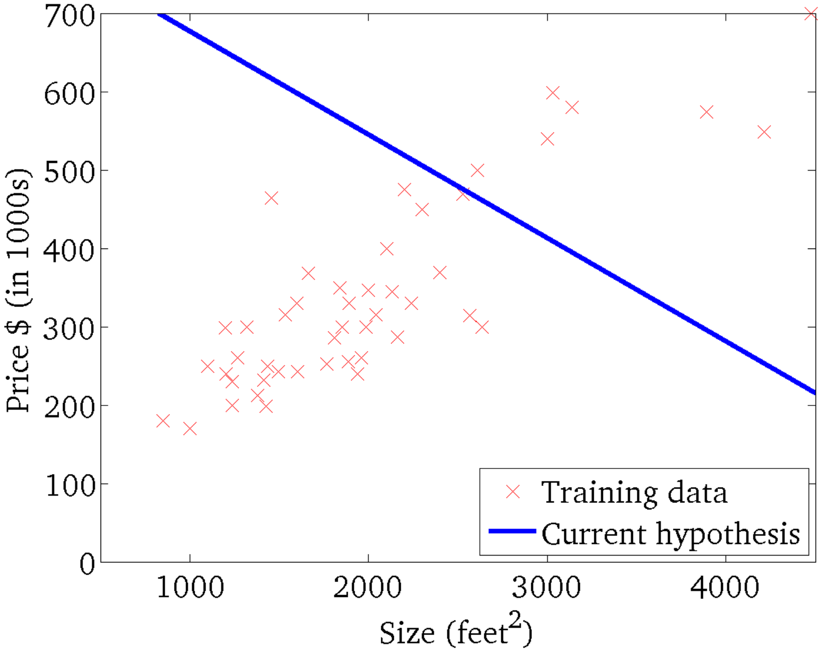
$$J(w_0, w_1)$$

(function of the parameters w_0, w_1)



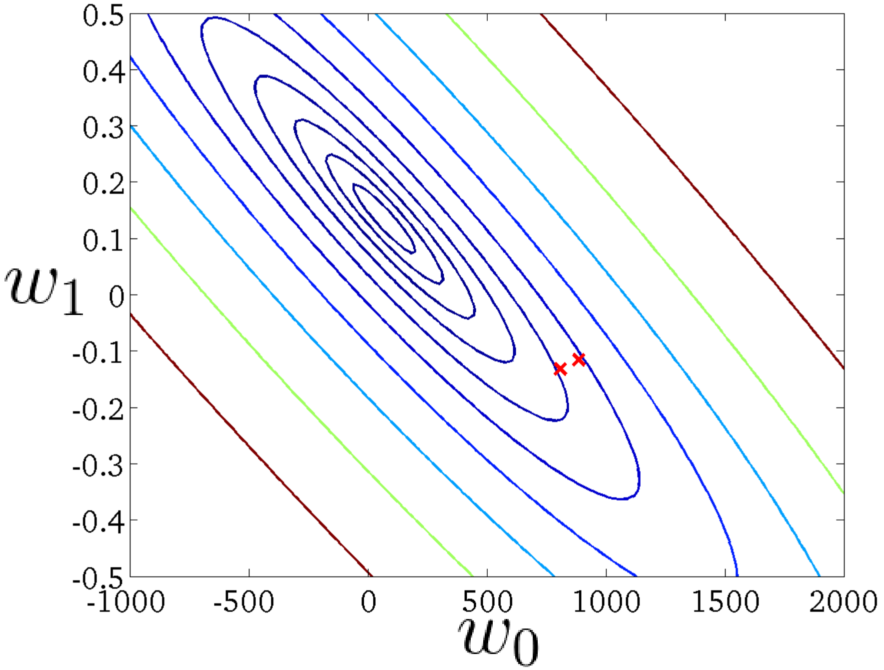
$$h(x)$$

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



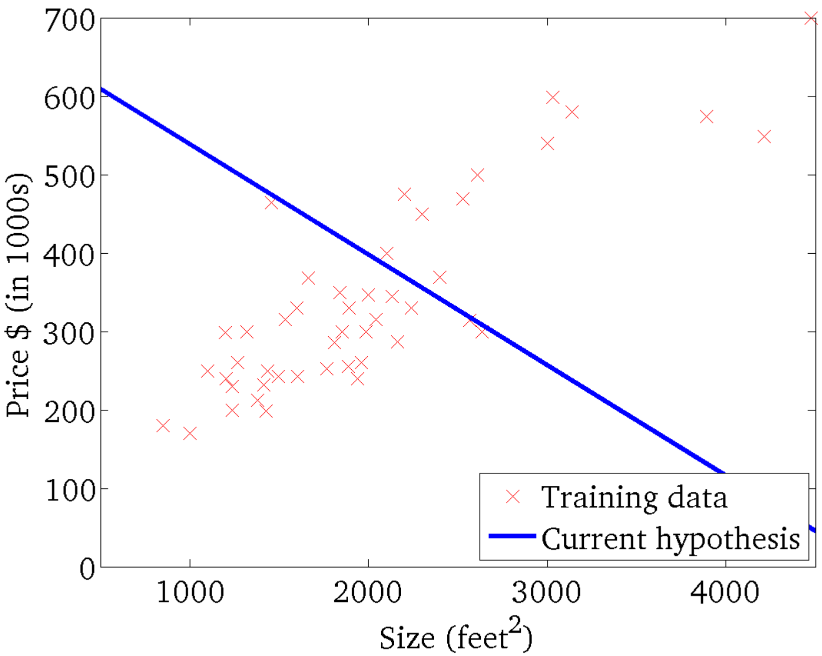
$$J(w_0, w_1)$$

(function of the parameters w_0, w_1)



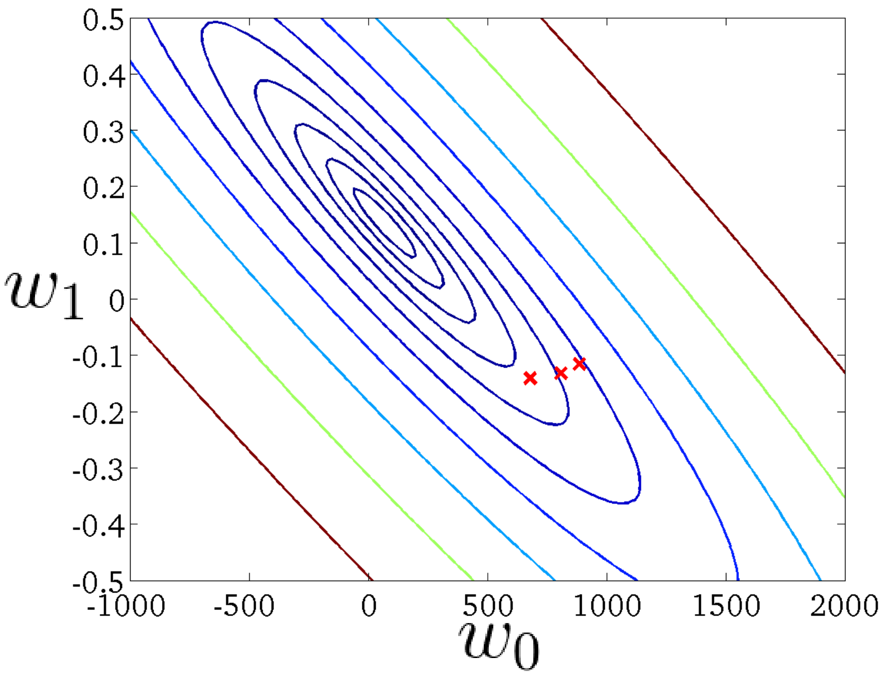
$$h(x)$$

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



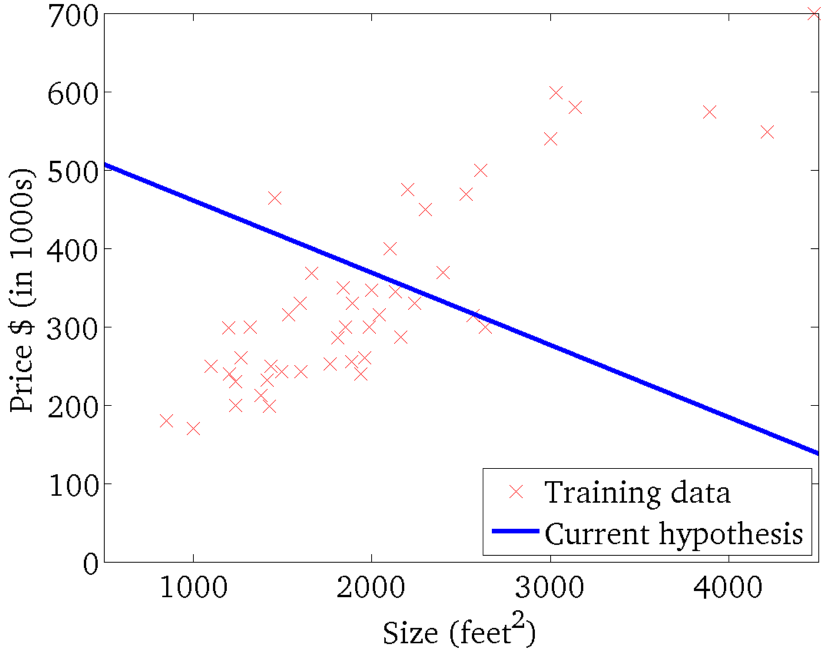
$$J(w_0, w_1)$$

(function of the parameters w_0, w_1)



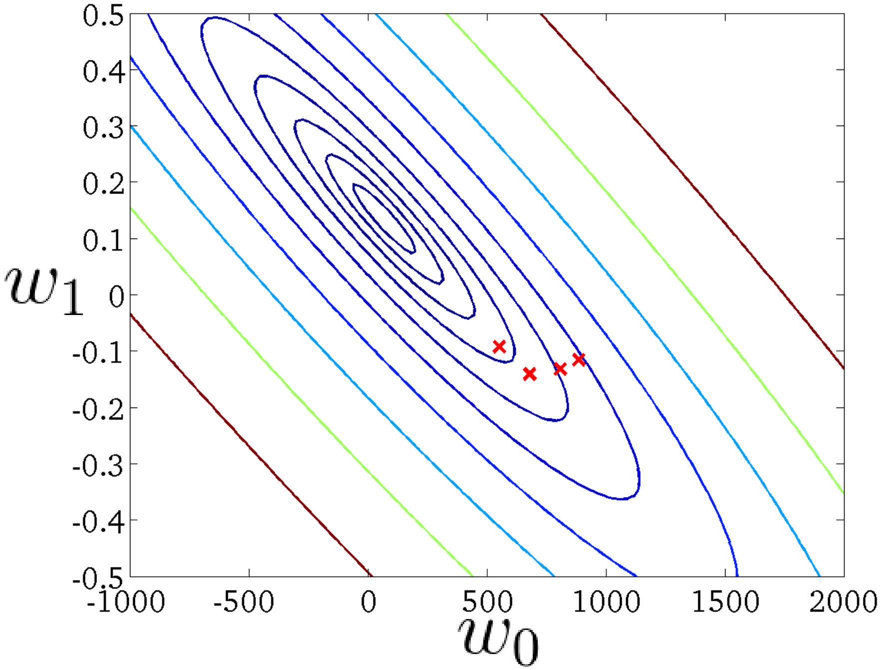
$$h(x)$$

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



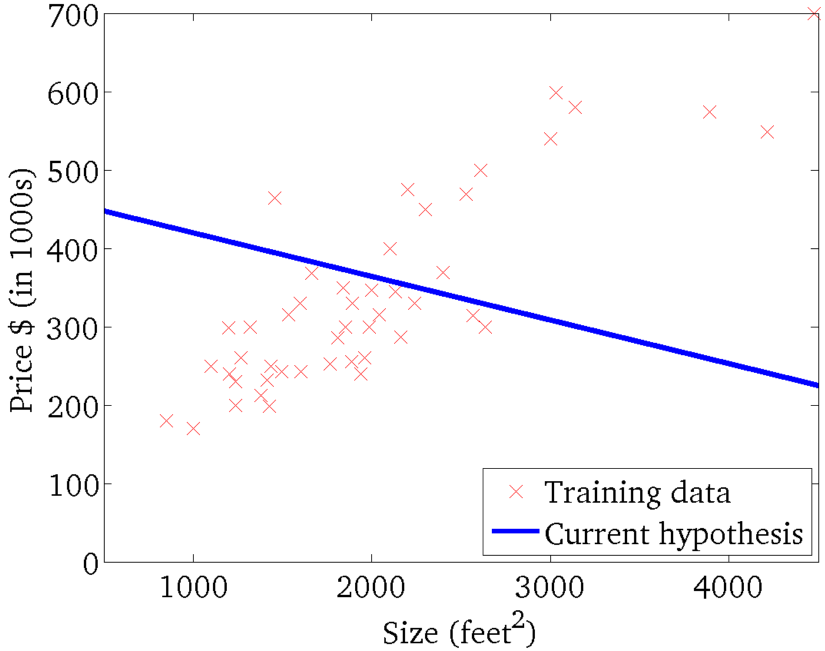
$$J(w_0, w_1)$$

(function of the parameters w_0, w_1)



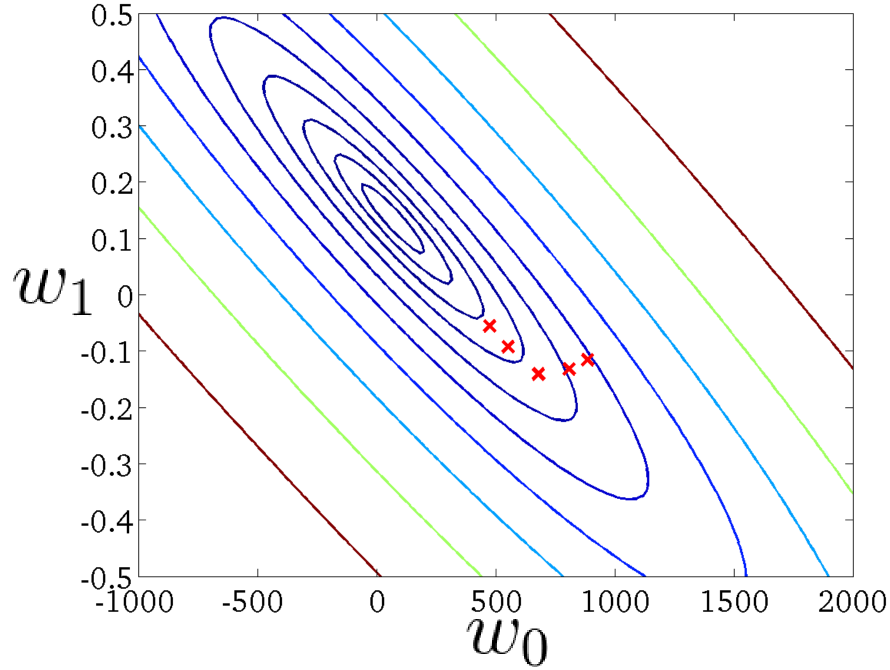
$$h(x)$$

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



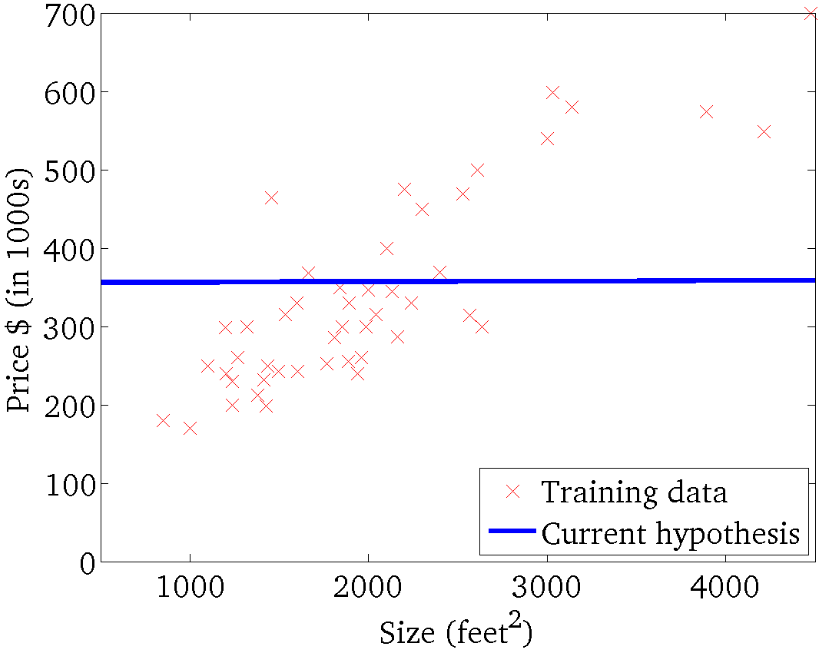
$$J(w_0, w_1)$$

(function of the parameters w_0, w_1)



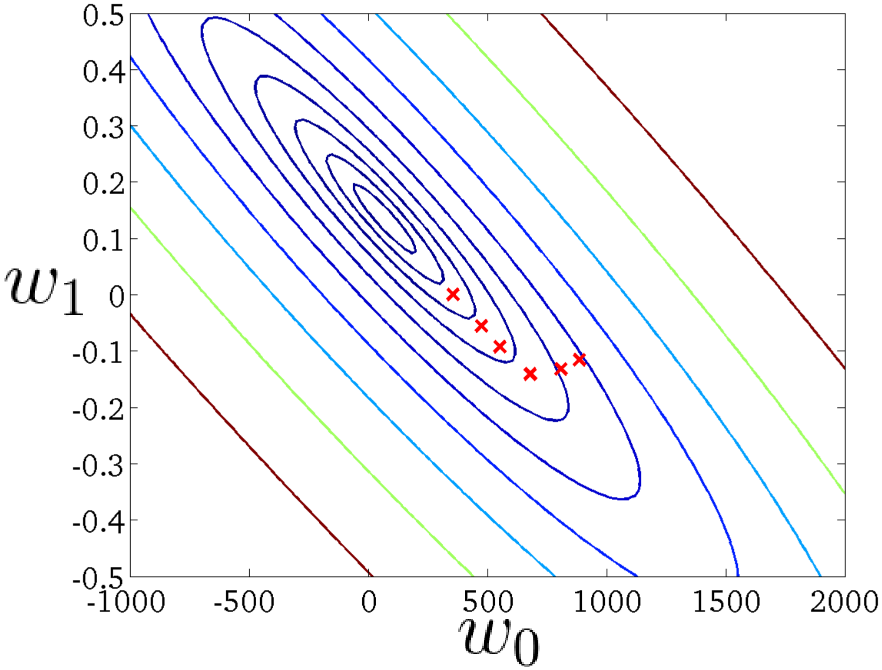
$$h(x)$$

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



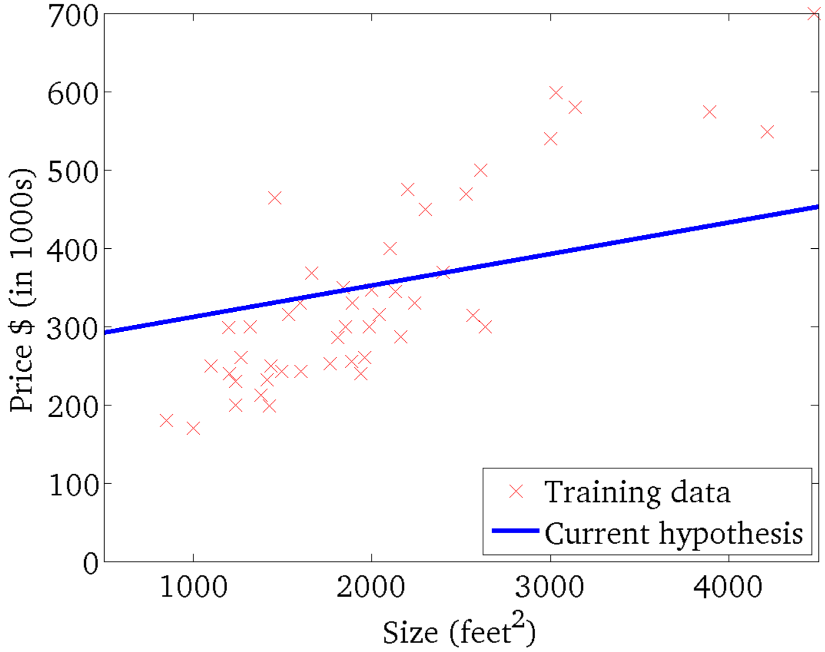
$$J(w_0, w_1)$$

(function of the parameters w_0, w_1)



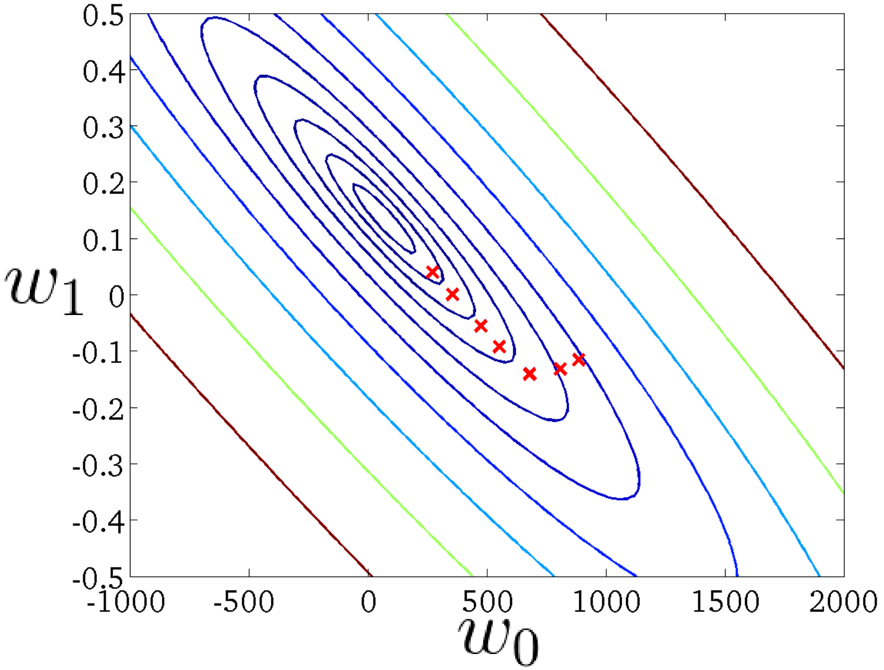
$h(x)$

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



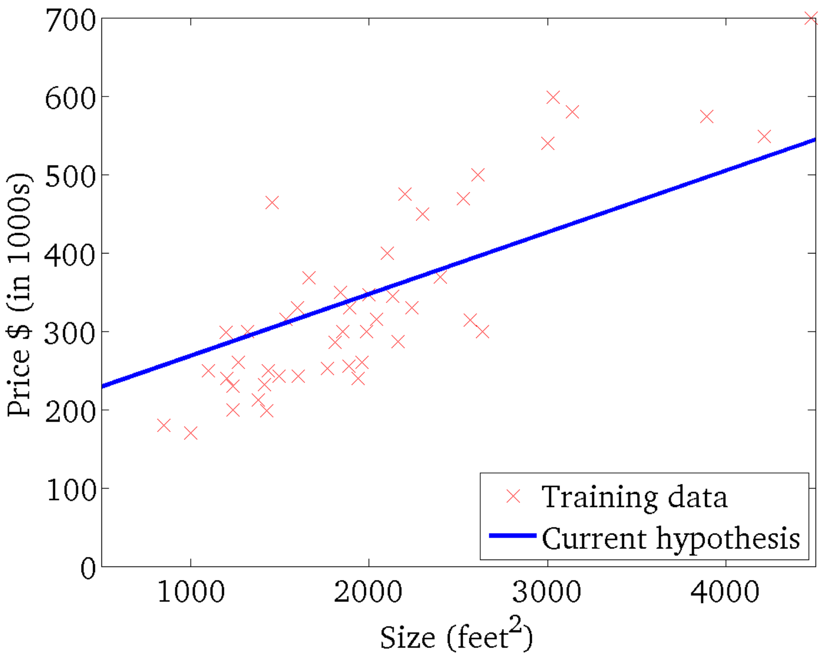
$J(w_0, w_1)$

(function of the parameters w_0, w_1)



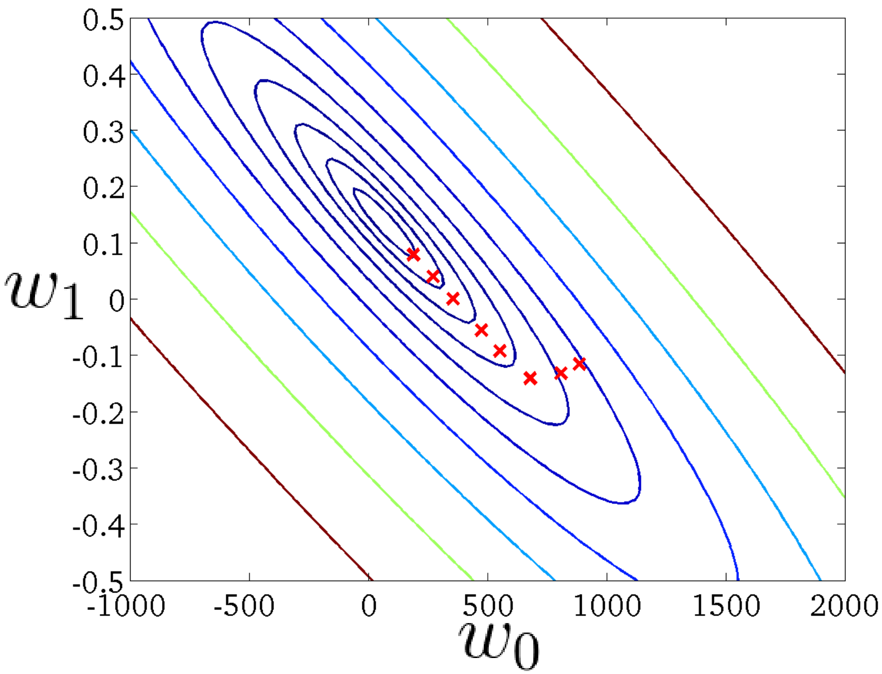
$$h(x)$$

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



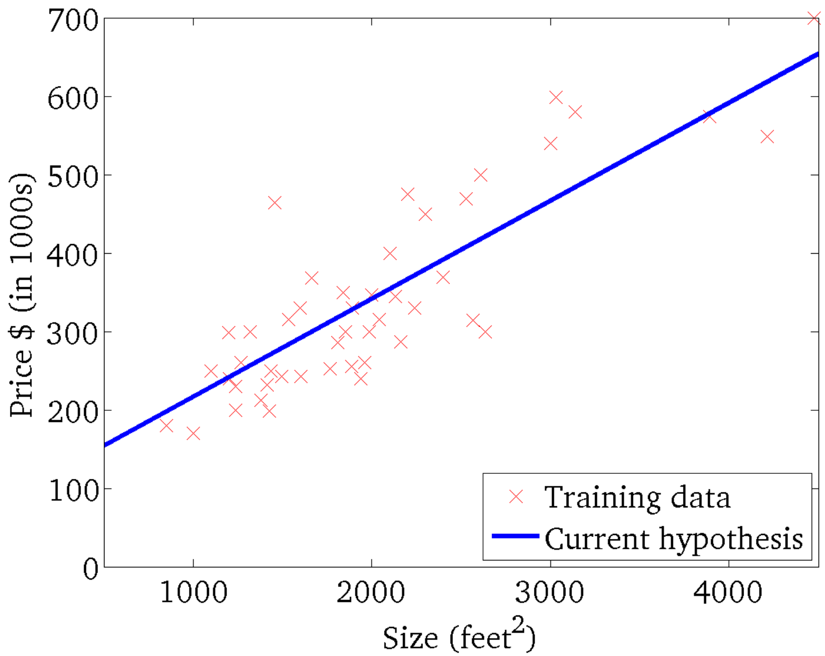
$$J(w_0, w_1)$$

(function of the parameters w_0, w_1)



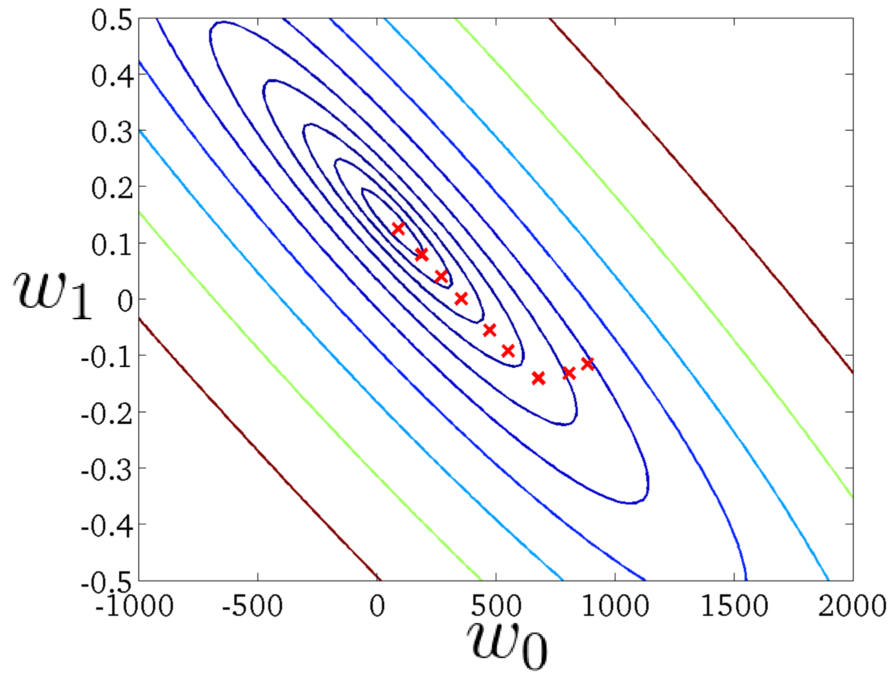
$$h(x)$$

(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)



"Batch" Gradient Descent

"Batch": Each step of gradient descent uses all the training examples.

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i\right) \\
&\quad w_1 := w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i\right) \cdot x_i \\
&\}
\end{aligned}
$$

Linear Regression with Multiple Variables

Multivariate Linear Regression

Multiple features (variables).

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Notation:

n = number of features

$\mathbf{x}_i$ = input (features) of i^{th} training example.

x_{ij} = value of feature j in i^{th} training example.

Multivariate Linear Regression

Hypothesis:

Previously: $h(x) = w_0 + w_1 x$

$\mathbf{x} \in \mathbb{R}^n \qquad h(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$

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

$$h(\mathbf{x}) = \sum_{j=0}^{n} w_j x_j = \mathbf{w}^T \mathbf{x} = \langle \mathbf{w}, \mathbf{x} \rangle$$

$$\mathbf{x} \in \mathbb{R}^{n+1} \quad \mathbf{w} \in \mathbb{R}^{n+1}$$

Gradient Descent for Multivariate Linear Regression

Hypothesis: $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

Parameters: $\mathbf{w} = (w_0, w_1, \ldots, w_n)^T$

Cost function: $J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^{m} (h(\mathbf{x}_i) - y_i)^2$

Gradient descent:

Repeat $\{$

$$\mathbf{w} := \mathbf{w} - \alpha \nabla J(\mathbf{w})$$

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

$$\nabla J(\mathbf{w}) = \left(\frac{\partial}{\partial w_0} J(\mathbf{w}), \frac{\partial}{\partial w_1} J(\mathbf{w}), \ldots, \frac{\partial}{\partial w_n} J(\mathbf{w}) \right)$$

Univariate LR vs. Multivariate LR

Gradient Descent

Previously (n=1):

Repeat $\{$

$$w_0 := w_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}\left(h(x_i) - y_i\right)}_{\frac{\partial}{\partial w_0}J(w_0, w_1)}$$

$$w_1 := w_1 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}\left(h(x_i) - y_i\right) \cdot x_i}_{\frac{\partial}{\partial w_1}J(w_0, w_1)}$$

$\}$

(simultaneously update θ_0, θ_1)

New algorithm $(n \geq 1)$:

Repeat $\{$

$$\mathbf{w} := \mathbf{w} - \alpha \nabla J(\mathbf{w})$$

$\}$

$$\nabla J(\mathbf{w}) = \left(\frac{\partial}{\partial w_0}J(\mathbf{w}), \frac{\partial}{\partial w_1}J(\mathbf{w}), \ldots, \frac{\partial}{\partial w_n}J(\mathbf{w})\right)$$

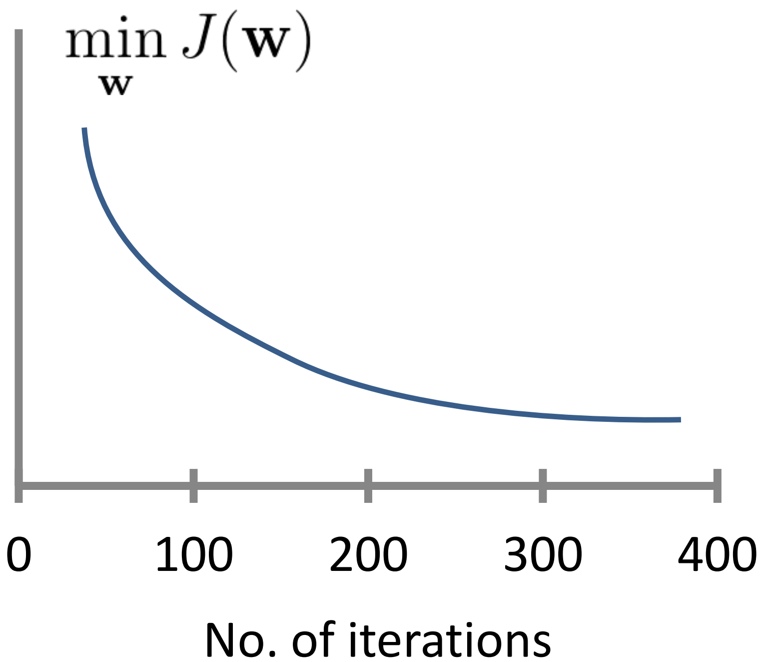
$$\frac{\partial}{\partial w_0}J(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}_i) - y_i)$$

$$\frac{\partial}{\partial w_1}J(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}_i) - y_i)x_{i1}$$

$\vdots$

Convergence and Learning Rate

$$\mathbf{w} := \mathbf{w} - \alpha \nabla J(\mathbf{w})$$

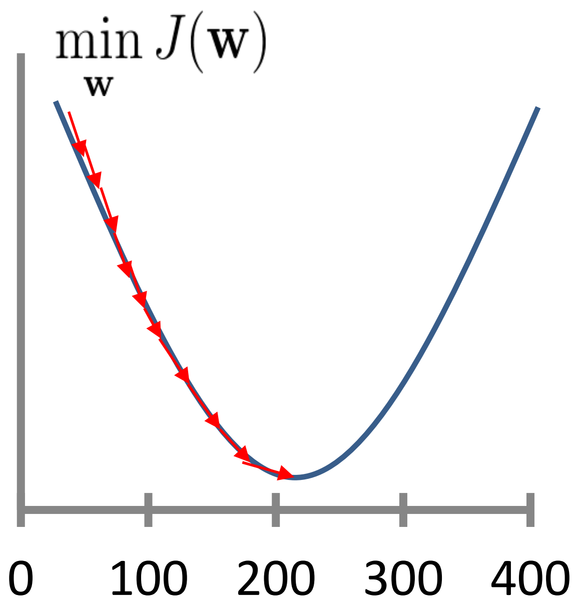


Example automatic convergence test:

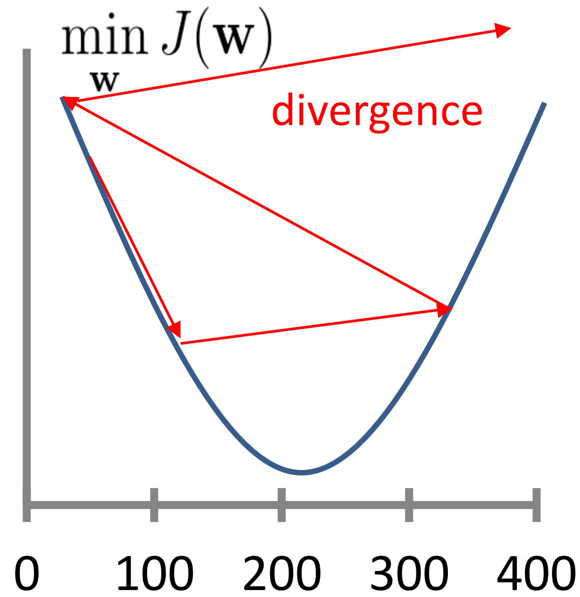
Declare convergence if $J(\mathbf{w})$ decreases by less than 10^{-3} in one iteration.

For sufficiently small α , $J(\mathbf{w})$ should decrease on every iteration.
But if α is too small, gradient descent can be slow to converge.
If α is too large: $J(\mathbf{w})$ may not decrease on every iteration; may not converge.

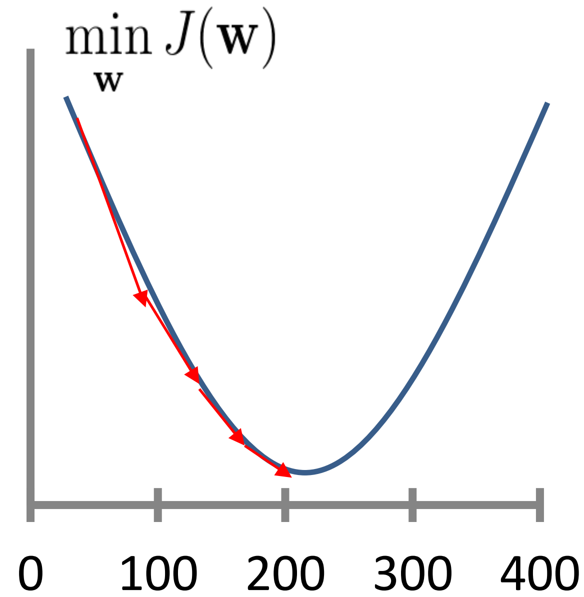
Learning Rate



too small constant



too large



gradually decreased

$$\alpha_t = \frac{\alpha}{t}$$

Normal Equation for Linear Regression

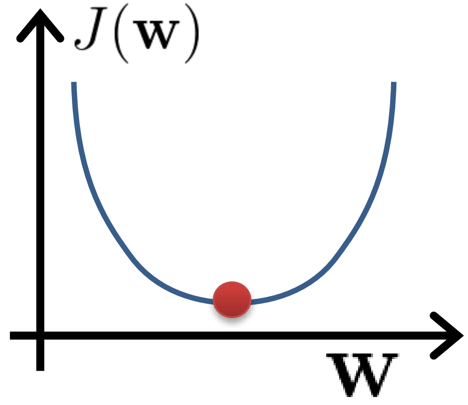
Normal Equation

Gradient Descent

- Iterative approach

Normal Equation

- Analytical method to solve $\mathbf{w}$



Intuition Example: If 1D

$$J(w) = aw^2 + bw + c$$

$$\mathbf{w} \in \mathbb{R}^{n+1} \quad J(\mathbf{w}) = \frac{1}{2m}\sum_{i=1}^{m}(h(\mathbf{x}_i) - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(\mathbf{w}^T\mathbf{x}_i - y_i)^2$$

$$\nabla J(\mathbf{w}) = \mathbf{0} \longrightarrow$$ **Solve equation to find w**

Normal Equation

$$J(\mathbf{w}) = \frac{1}{2m}\sum_{i=1}^{m}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 = \frac{1}{2m}(X\mathbf{w}-\mathbf{y})^T(X\mathbf{w}-\mathbf{y})$$

$$(\mathbf{w}^T\mathbf{x}_1 - y_1)$$

$$X\mathbf{w}-\mathbf{y} = \begin{pmatrix} x_{10} & x_{11} & \dots & x_{1n} \\ x_{20} & x_{21} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m0} & x_{n1} & \dots & x_{mn} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

$$m \times (n+1) \qquad (n+1)\times 1 \qquad m \times 1$$

Normal Equation

- Matrix-vector formulation

$$J(\mathbf{w}) = \frac{1}{2m}(X\mathbf{w} - \mathbf{y})^T(X\mathbf{w} - \mathbf{y})$$

$$\nabla J(\mathbf{w}) = \nabla_{\mathbf{w}}\left(\frac{1}{2m}(X\mathbf{w} - \mathbf{y})^T(X\mathbf{w} - \mathbf{y})\right)$$

$$= X^T X\mathbf{w} - X^T\mathbf{y} = \mathbf{0}$$

$$X^T X\mathbf{w} = X^T\mathbf{y}$$

- Analytical solution

$$\mathbf{w} = \left((X^T X)^{-1} X^T\right)\mathbf{y} = X^{\dagger}\mathbf{y}$$

$$X^{\dagger} = (X^T X)^{-1} X^T$$

The Pseudo-inverse $X^{\dagger}$

$$\mathbf{w} = \left((X^T X)^{-1} X^T\right) \mathbf{y} = X^{\dagger} \mathbf{y}$$

$$X^{\dagger} = (X^T X)^{-1} X^T$$

$$\underbrace{\left(\underbrace{\left[X^T X \right]}_{(n+1)\times(n+1)} \right)^{-1} \underbrace{\left[\quad X^T \quad \right]}_{(n+1)\times m}}_{(n+1)\times m}$$

Example

Examples: $m = 4.$

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$\mathbf{w} = \left((X^T X)^{-1} X^T\right) \mathbf{y} = X^\dagger \mathbf{y}$$

$(X^T X)^{-1}$ is inverse of matrix $X^T X$.

m **training examples,** n **features.**

Gradient Descent

- Need to choose α.
- Needs many iterations.
- Works well even when n is large.

Normal Equation

- No need to choose α.
- Don't need to iterate.
- Need to compute $(X^T X)^{-1}$
- Slow if n is very large.

Summary

- Supervised Learning
- Linear Models for Regression
  - One Variable
  - Multiple Variables
- Basics of Optimization
  - Convexity
  - Iterative Solution : Gradient Descent
  - Analytical Solution: Normal Equation
- Appendix:
  - Linear Basis Function Models

QUESTIONS?!

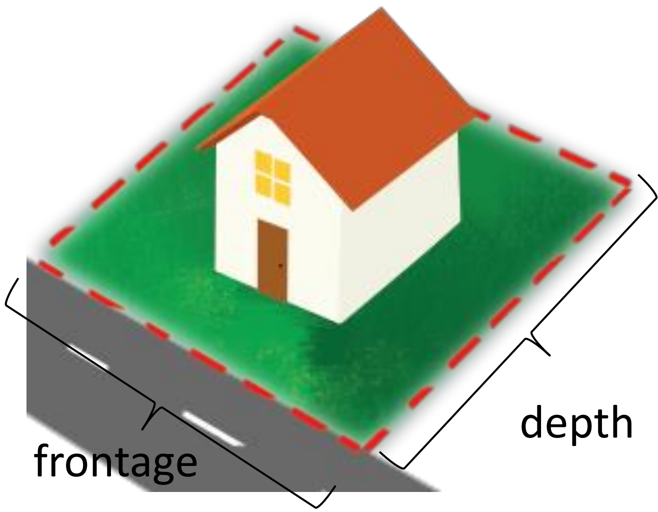
Linear Basis Function Models

Linear Regression

Housing prices prediction

$$h(x) = w_0 + w_1 \times frontage + w_2 \times depth$$

x1 x2

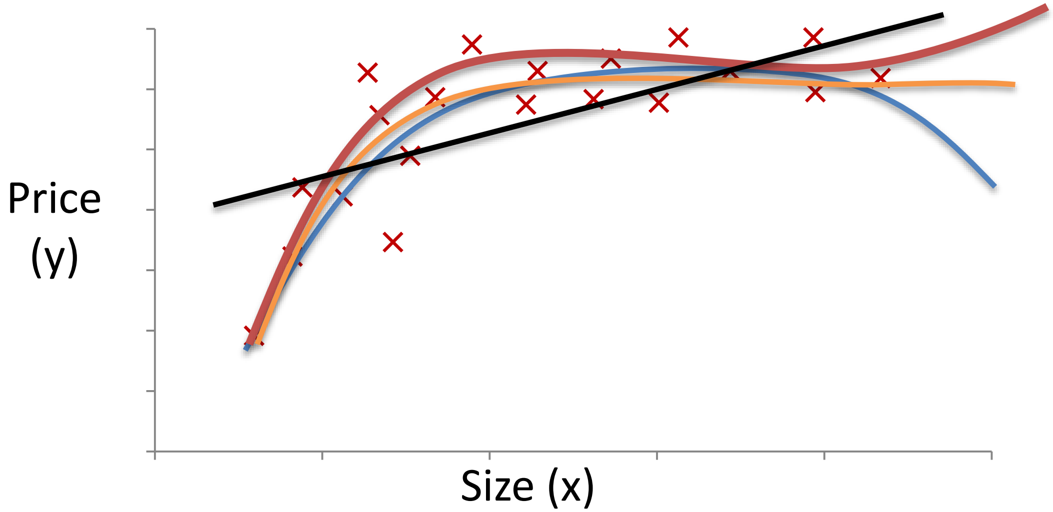


Creating a new variable:

Land Area x = *frontage * depth*

$$h(x) = w_0 + w_1 \times x$$

Polynomial Regression



$$w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

$$w_0 + w_1 x + w_2 x^2$$

"Polynomial Curve Fitting"

$$h(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$
$$= w_0 + w_1(size) + w_2(size)^2 + w_3(size)^3$$

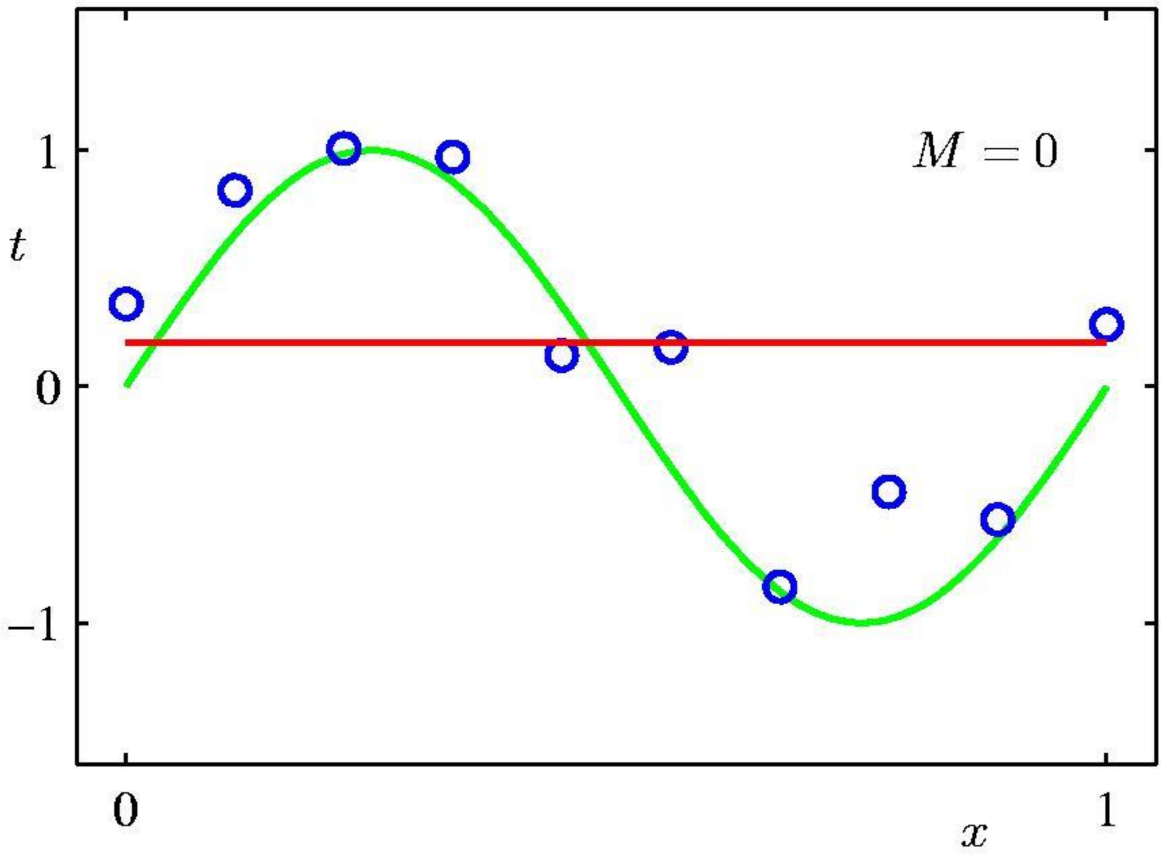
$$x_1 = (size)$$
$$x_2 = (size)^2$$
$$x_3 = (size)^3$$

Choices of proper features

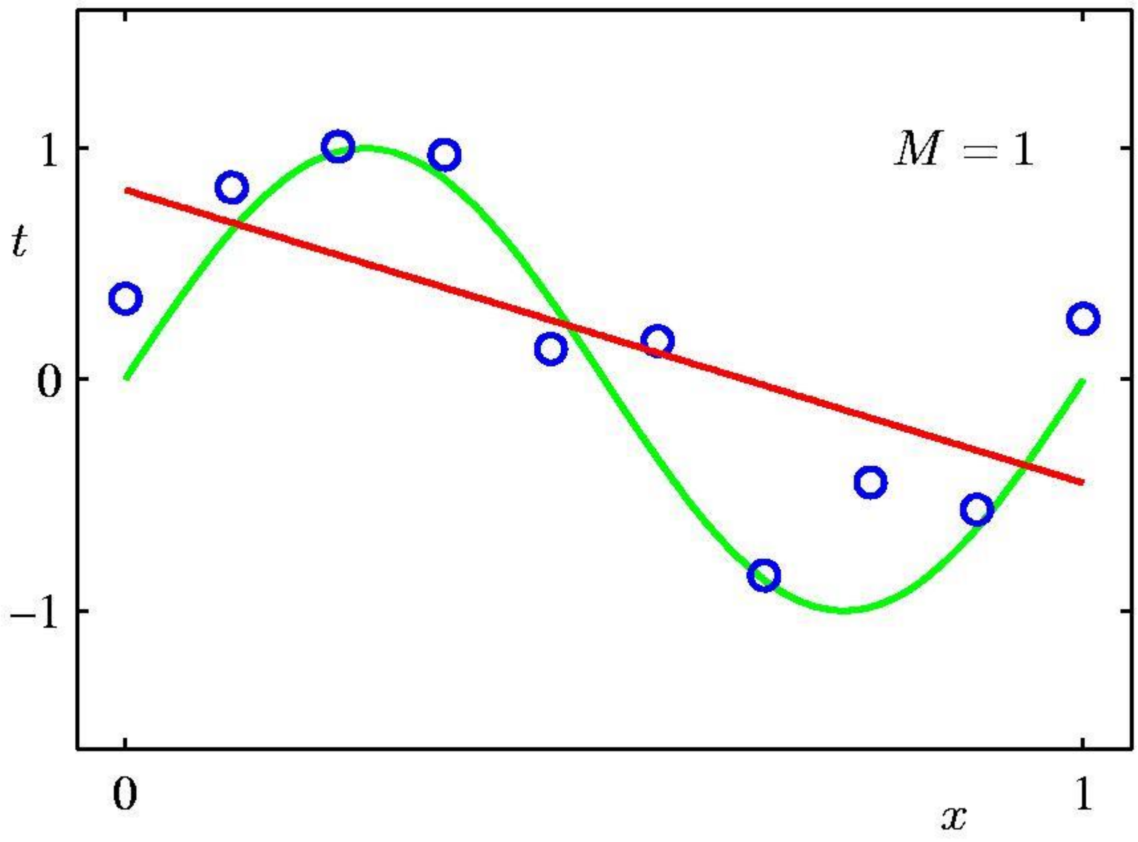
$$h(x) = w_0 + w_1(size) + w_2(size)^2$$

$$h(x) = w_0 + w_1(size) + w_2\sqrt{(size)}$$

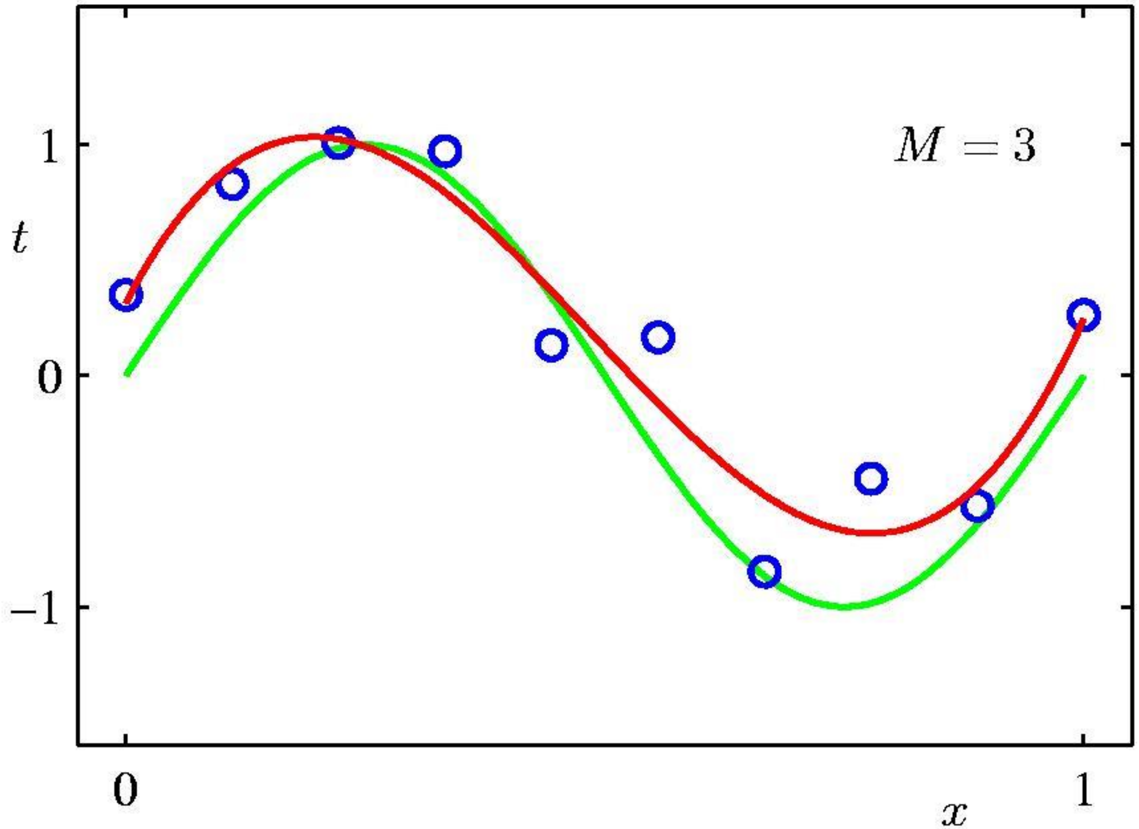
0^{th} Order Polynomial



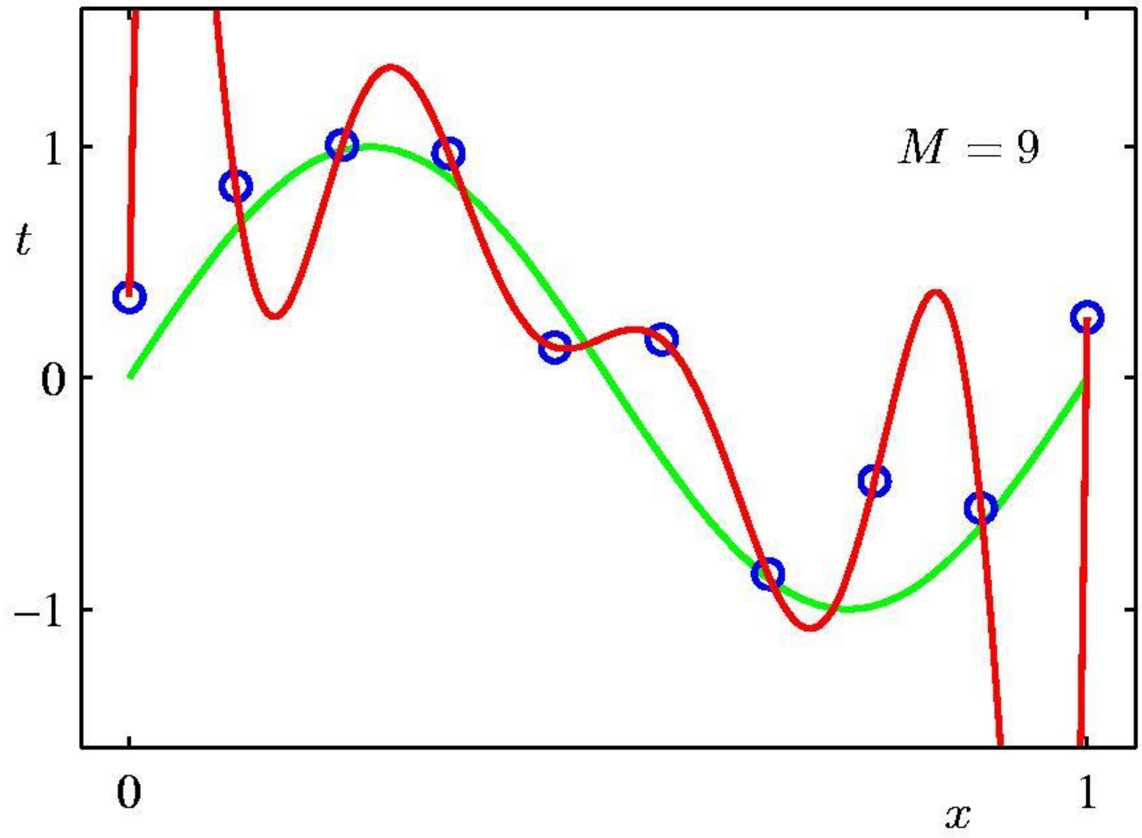
1st Order Polynomial



3rd Order Polynomial



9^{th} Order Polynomial



Linear Basis Function Models

- Linear combination of input variables

$$h(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_n x_n$$

- Linear combination of fixed non-linear functions (basis functions) of input variables

$$h(\mathbf{x}) = w_0 + w_1 \phi_1(\mathbf{x}) + \ldots + w_B \phi_B(\mathbf{x})$$

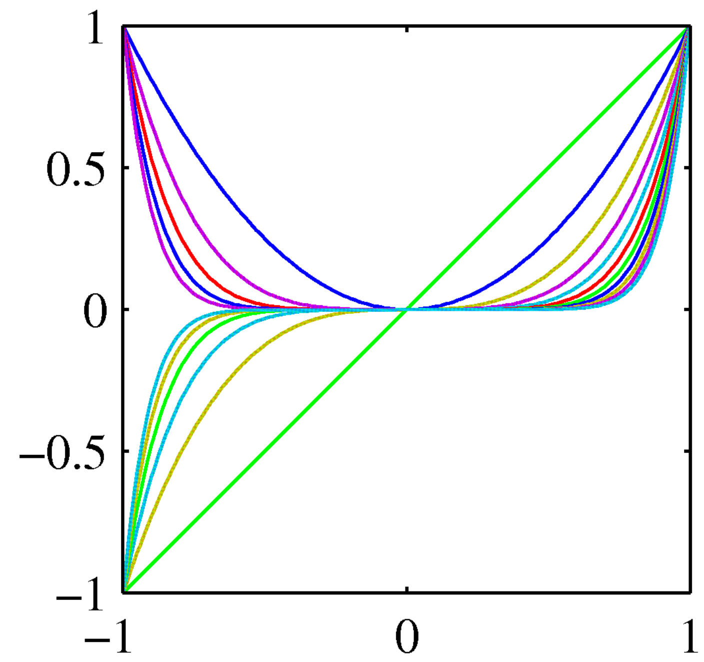
- where ϕ_j ‘s are known as *basis functions*.

Linear Basis Function Models (2)

Polynomial basis functions:

$$\phi_j(x) = x^j.$$

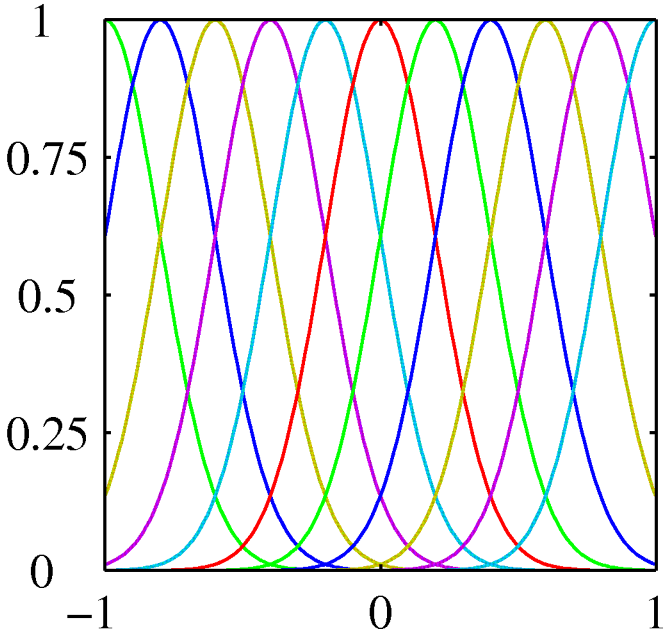
These are global; a small change in x affect all basis functions.



Linear Basis Function Models (3)

Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$



These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).

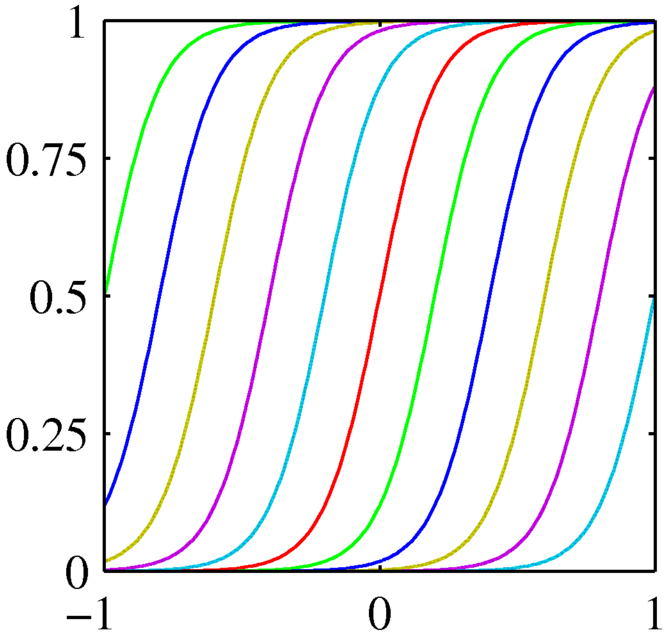
Linear Basis Function Models (4)

Sigmoidal basis functions:

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$



Also these are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (slope).

Solutions

- Problem: **m** data points in **n** dimensions

- Hypothesis

$$h(\mathbf{x}) = w_0 + w_1\phi_1(\mathbf{x}) + \ldots + w_B\phi_B(\mathbf{x})$$

- Objective Function

$$J(\mathbf{w}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h(\mathbf{x}_i) - y_i\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(\sum_{j=0}^{B} w_j\phi_j(\mathbf{x}_i) - y_i\right)^2$$

Gradient Descent Solution

- Gradient Descent

$$\mathbf{w} := \mathbf{w} - \alpha \nabla J(\mathbf{w}) \qquad\qquad \mathbf{w} \in \mathbb{R}^{B+1}$$

- Repeat until convergence

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}_i) - y_i \right) \phi_j(\mathbf{x}_i)$$

(simultaneously update for every j = 0,…, B) $\qquad \phi_0(\mathbf{x}_i) = 1$

Normal Equation for Analytical solution

- Normal Equation for Analytical solution
  - Computing the gradient and setting it to zero yields

$$\Theta = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$

  - where

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_B(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_B(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_m) & \phi_1(x_m) & \dots & \phi_B(x_m) \end{pmatrix}$$

Limitations of Fixed Basis Functions

- B basis functions along each dimension of a n-dimensional input space requires exponential number of basis functions due to the curse of dimensionality.
- **Example:**

  Consider n input variables with a general polynomial with coefficients up to order 3 would take the form:

$$h(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} w_{ijk} x_i x_j x_k$$

- In later chapters, we shall see how we can get away with fewer basis functions, by choosing these using the training data.