Deep Learning for Natural Language Processing

Cornelia Caragea

Computer Science
University of Illinois at Chicago

Credits for slides: Piyush Rai, Andrew Ng

Basics of Machine Learning

Today

Lecture Structure:

- Fundamental Problems of Learning
- Univariate Linear Regression
- Gradient Descent
- Multivariate Linear Regression

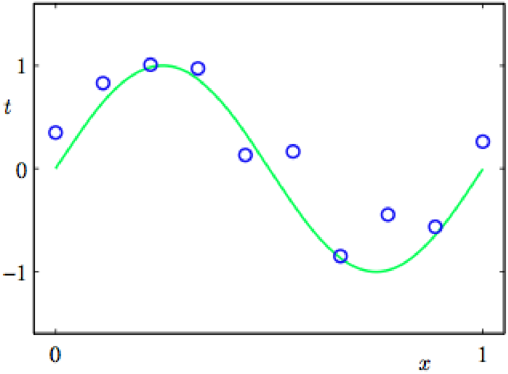
Required reading: DL Section 4.3

Three Fundamental Problems of Learning

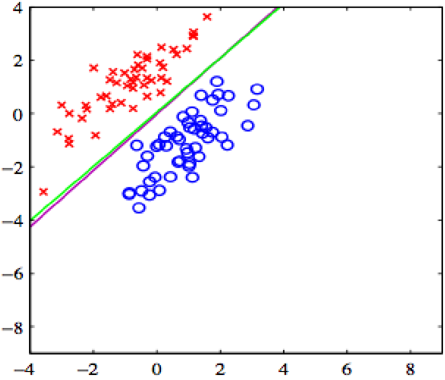
- **Regression**: Learning to predict continuous outputs associated with given observations
  - Example: how long does it take to bike to Northampton? How much does it cost to visit Florida? How much money can I make if do a PhD in CS?
- **Classification**: Learning to predict discrete labels associated with given observations.
  - Binary classification: positive vs. negative examples
  - Multiclass classification: digit recognition
- **Unsupervised learning**: Learning to group objects into categories, without any training labels.
  - Examples: density estimation, clustering

Regression

Plot of a training data set of $N = 10$ points, shown as blue circles, each comprising an observation of the input variable x along with the corresponding target variable t. The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x, without knowledge of the green curve.

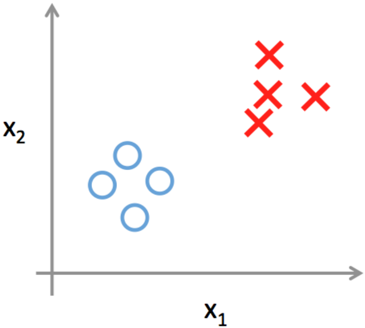


Linearly Separable Classification

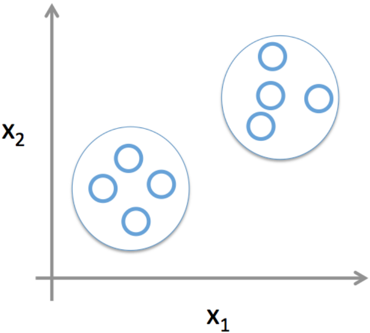


- Spam vs. not spam
- Breast cancer (malignant, benign)

Unsupervised Learning



Supervised learning

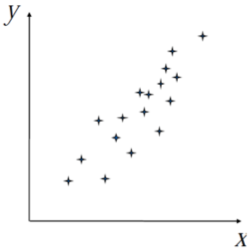


Unsupervised learning

Linear Regression with One Variable

Linear Regression: One-Dimensional Case

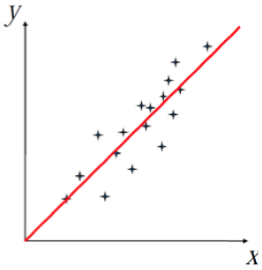
Regression: we observe a real-valued input x and we wish to use x to predict the value of a real-valued target t (or y).



- **Given:** a set of N input-target pairs
  - The inputs (x) and targets (y) are one dimensional scalars
- **Goal:** Model the relationship between x and y

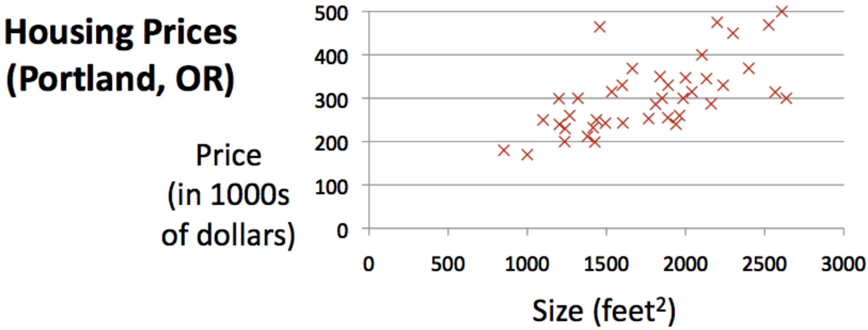
Linear Regression: One-Dimensional Case

- **Assumption:** the relationship between x and y is linear
- Linear relationship - defined by a straight line with parameter w
- Equation of the straight line: $y = wx$ $(y = w_0 + w_1x_1)$

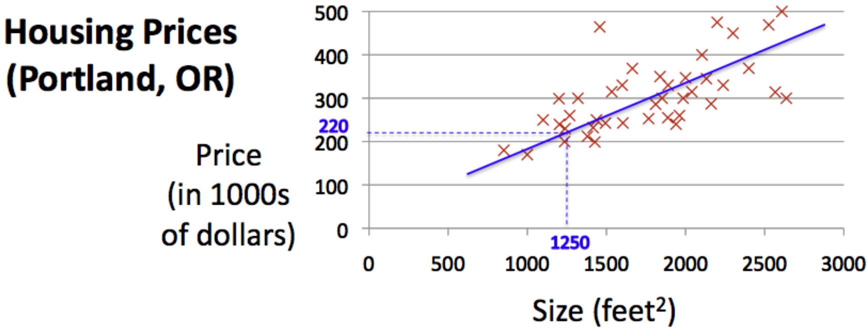


- The line may not fit the data *exactly*
- But we can try making the line a reasonable approximation

Example - House Price Prediction



Example - House Price Prediction



Data Representation

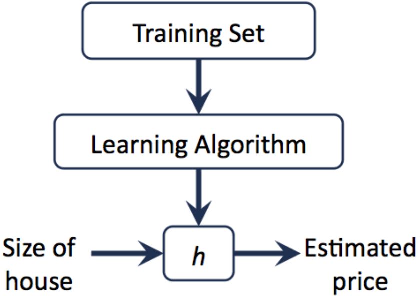
Training set of housing prices (Portland, OR):

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

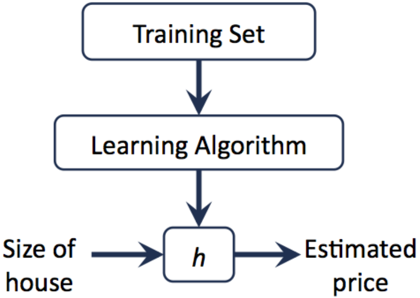
Notation:

- N = number of training examples
- x's = "input" variable / features
- y's = "output" variable / "target" variable
- (x, y) - one training example
- $(x^{(i)}, y^{(i)})$ - the i^{th} training example

Model Representation



Model Representation



How do we represent h?

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

Regression: Estimating $h_w(x)$ of x that minimizes a cost function.

Model Representation

Training set of housing prices (Portland, OR):

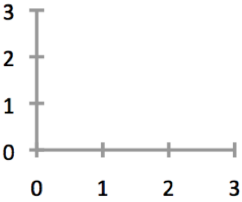
Size in feet$^2 (x)$	**Price** $(\$)$ **in** 1000**'s** (y)
2104	460
1416	232
1534	315
852	178
...	...

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

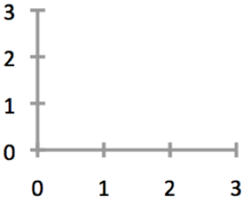
- Model parameters: w_i's
- How to choose w_i's?

Hypothesis

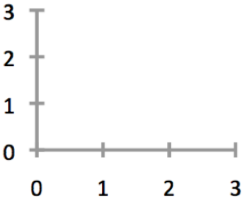
$$h_w(x) = w_0 + w_1 x$$



$w_0 = 1.5$
$w_1 = 0$



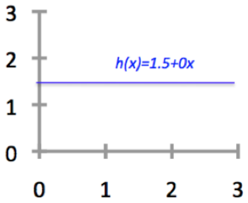
$w_0 = 0$
$w_1 = 0.5$

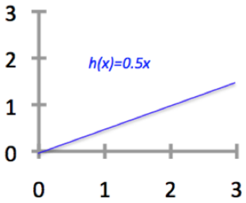


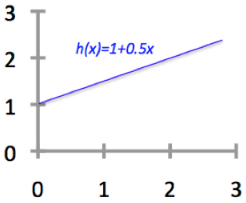
$w_0 = 1$
$w_1 = 0.5$

Hypothesis

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







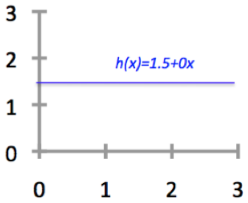
$w_0 = 1.5$
$w_1 = 0$

$w_0 = 0$
$w_1 = 0.5$

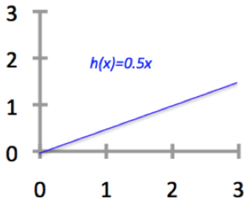
$w_0 = 1$
$w_1 = 0.5$

Hypothesis

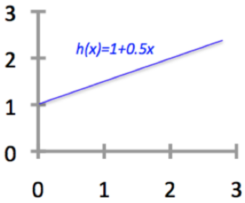
$$h_w(x) = w_0 + w_1 x$$



$w_0 = 1.5$
$w_1 = 0$



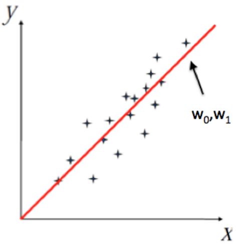
$w_0 = 0$
$w_1 = 0.5$

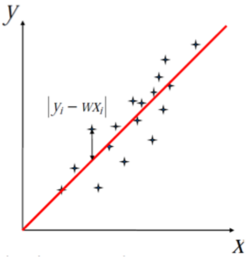


$w_0 = 1$
$w_1 = 0.5$

How to choose a hypothesis?

Intuition





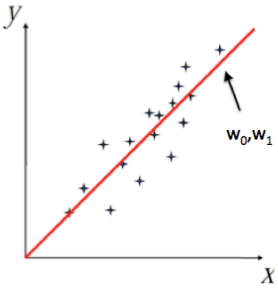
Idea: Choose w_0, w_1 s.t. $h_w(x)$ is close to y for our training examples (x, y).

- Error for the pair (x_i, y_i) pair: $e_i = y_i - wx_i$
- The total squared error:

$$E = \frac{1}{2N} \sum_{i=1}^{N} e_i^2 = \frac{1}{2N} \sum_{i=1}^{N} (y_i - wx_i)^2$$

- The best fitting line is defined by w minimizing the total error E

The Cost Function



Idea: Choose w_0, w_1 s.t. $h_w(x)$ is close to y for our training examples (x, y).

$$\min_{w_0, w_1} \frac{1}{2N} \sum_{i=1}^{N} (h_w(x_i) - y_i)^2 \, ,$$

where

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

$$E(w_0, w_1) = \frac{1}{2N} \sum_{i=1}^{N} (h_w(x_i) - y_i)^2 \, ,$$

Hence,

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

Cost Function Intuition

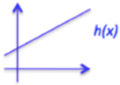
Simplified:

Hypothesis:

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

Parameters:

$$w_0, w_1$$



Cost Function:

$$E(w_0, w_1) = \frac{1}{2N}\sum_{i=1}^{N}(h_w(x_i) - y_i)^2,$$

Goal:

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

Hypothesis:

$$h_w(x) = w_1 x, \boxed{w_0 = 0}$$

Parameters:

$$w_1$$

Cost Function:

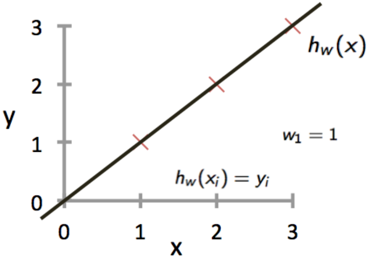
$$E(w_1) = \frac{1}{2N}\sum_{i=1}^{N}(h_w(x_i) - y_i)^2,$$

Goal:

$$\min_{w_1} E(w_1),$$

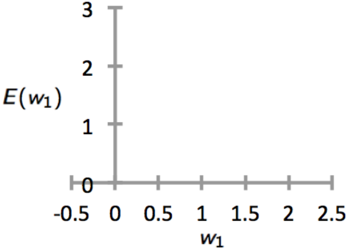
$h_w(x)$

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

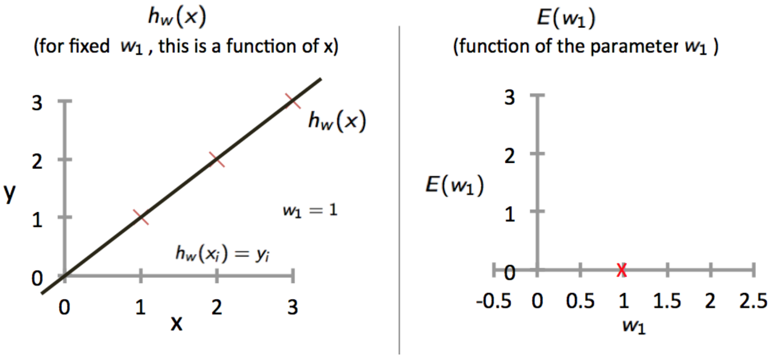


$E(w_1)$

(function of the parameter w_1)



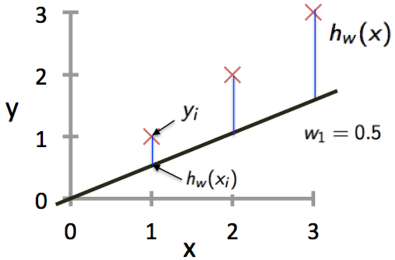
$$
\begin{aligned}
E(w_1) &= \frac{1}{2N}\sum_{i=1}^{N}(h_w(x_i) - y_i)^2 = \frac{1}{2N}\sum_{i=1}^{N}(w_1 x_i - y_i)^2 \\
&= \frac{1}{2N}(0^2 + 0^2 + 0^2) = 0
\end{aligned}
$$



$$
\begin{aligned}
E(w_1) &= \frac{1}{2N}\sum_{i=1}^{N}(h_w(x_i)-y_i)^2 = \frac{1}{2N}\sum_{i=1}^{N}(w_1x_i-y_i)^2 \\
&= \frac{1}{2N}(0^2+0^2+0^2) = 0
\end{aligned}
$$

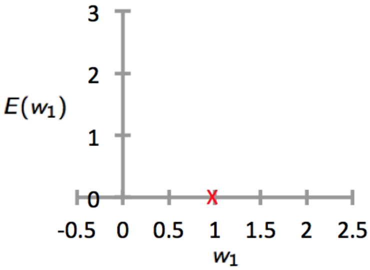
$h_w(x)$

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

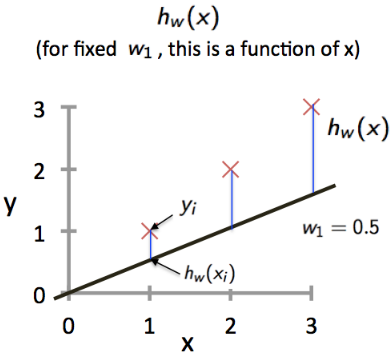


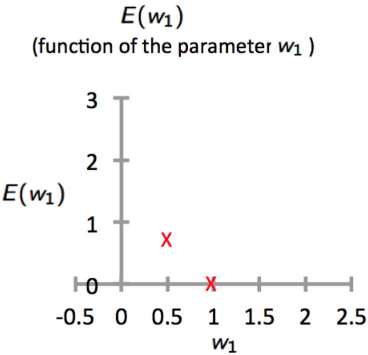
$E(w_1)$

(function of the parameter w_1)



$$
\begin{aligned}
E(0.5) &= \frac{1}{2N}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] \\
&= \frac{1}{2\times 3}(3.5) \approx 0.58
\end{aligned}
$$

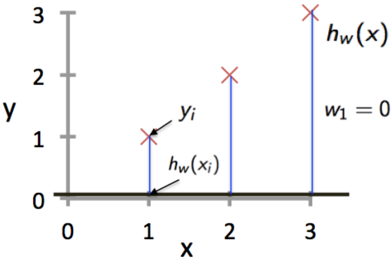




$$
\begin{aligned}
E(0.5) &= \frac{1}{2N}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] \\
&= \frac{1}{2\times 3}(3.5) \approx 0.58
\end{aligned}
$$

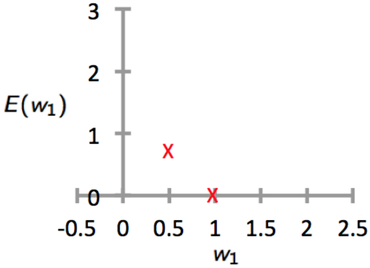
$h_w(x)$

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



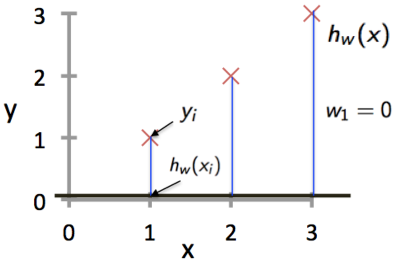
$E(w_1)$

(function of the parameter w_1)

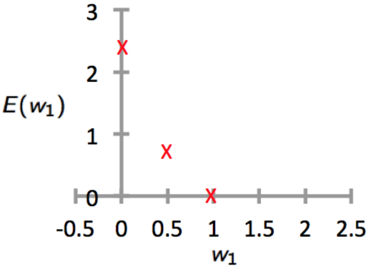


$$
\begin{aligned}
E(0) &= \frac{1}{2N}\left[1^2 + 2^2 + 3^2\right] \\
&= \frac{1}{2 \times 3}(14) \approx 2.3
\end{aligned}
$$

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



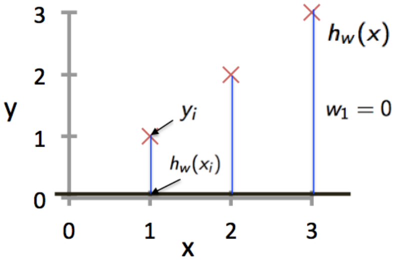
$E(w_1)$
(function of the parameter w_1)



$$
\begin{aligned}
E(0) &= \frac{1}{2N}\left[1^2 + 2^2 + 3^2\right] \\
&= \frac{1}{2 \times 3}(14) \approx 2.3
\end{aligned}
$$

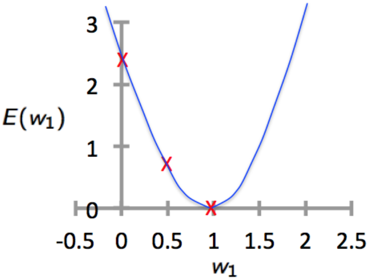
$h_w(x)$

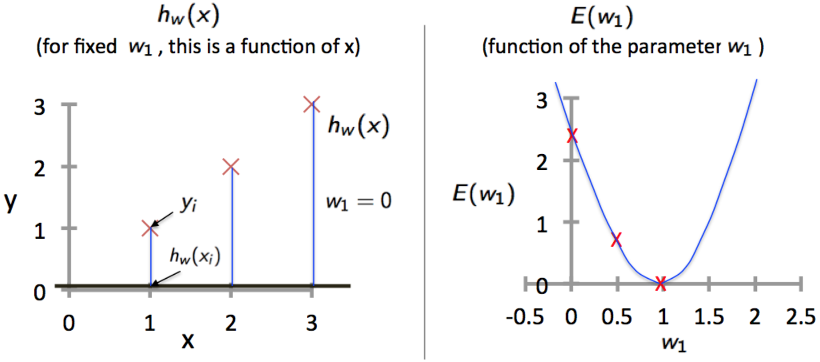
(for fixed w_1, this is a function of x)



$E(w_1)$

(function of the parameter w_1)





How to minimize E(w)?

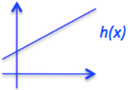
Cost Function Intuition - Unsimplified Version

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

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

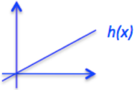
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Simplified

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

$\boxed{\theta_0 = 0}$

$$\theta_1$$



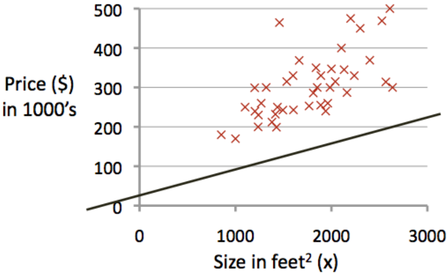
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

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

Notice the change in notation.

$h_\theta(x)$

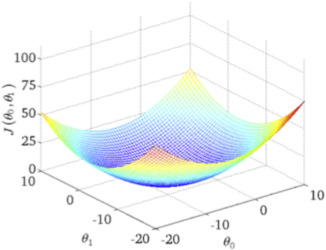
(for fixed θ_0, θ_1, this is a function of x)



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

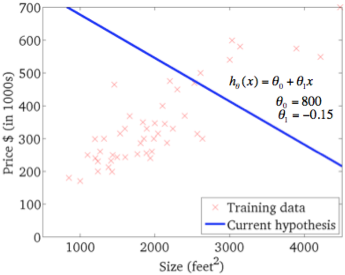
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



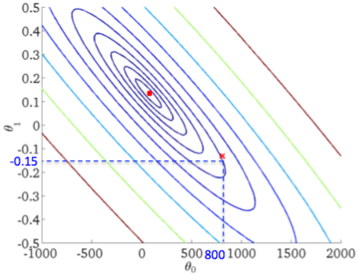
$h_\theta(x)$

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



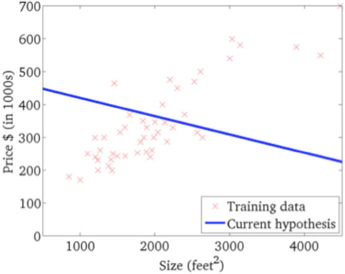
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



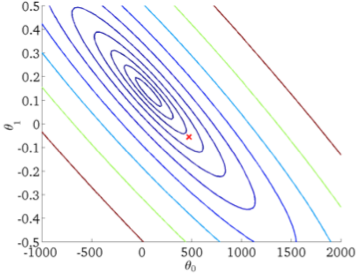
$h_\theta(x)$

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



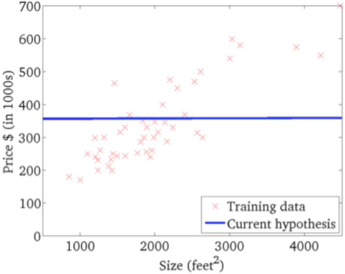
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



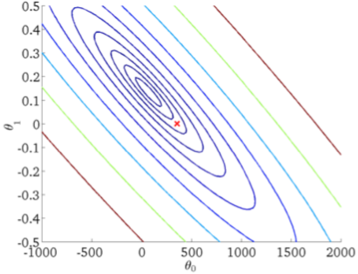
$h_\theta(x)$

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



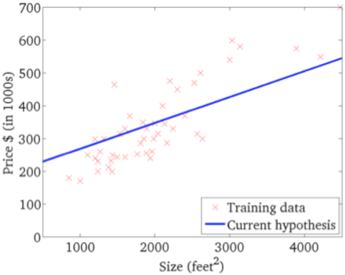
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



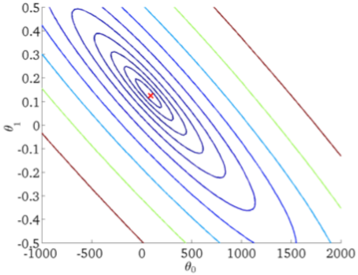
$h_\theta(x)$

(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)



Gradient Descent

How to minimize our cost function?

Given: Some function $E(w_0, w_1)$
Want: $\min_{w_0,w_1} E(w_0, w_1)$

Outline:

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

Gradient Descent Algorithm

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

- α - the learning rate
- $\frac{\partial}{\partial w_j} E(w_0, w_1)$ - derivative of E.

Correct: Simultaneous update

temp0 $:= w_0 - \alpha \frac{\partial}{\partial w_0} E(w_0, w_1)$
temp1 $:= w_1 - \alpha \frac{\partial}{\partial w_1} E(w_0, w_1)$
$w_0 :=$ temp0
$w_1 :=$ temp1

Incorrect

temp0 $:= w_0 - \alpha \frac{\partial}{\partial w_0} E(w_0, w_1)$
$w_0 :=$ temp0
temp1 $:= w_1 - \alpha \frac{\partial}{\partial w_1} E(w_0, w_1)$
$w_1 :=$ temp1

Gradient Descent Algorithm

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

- α - the learning rate
- $\frac{\partial}{\partial w_j} E(w_0, w_1)$ - derivative of E.

Correct: Simultaneous update

temp0 $:= w_0 - \alpha \frac{\partial}{\partial w_0} E(w_0, w_1)$
temp1 $:= w_1 - \alpha \frac{\partial}{\partial w_1} E(w_0, w_1)$
$w_0 :=$ temp0
$w_1 :=$ temp1

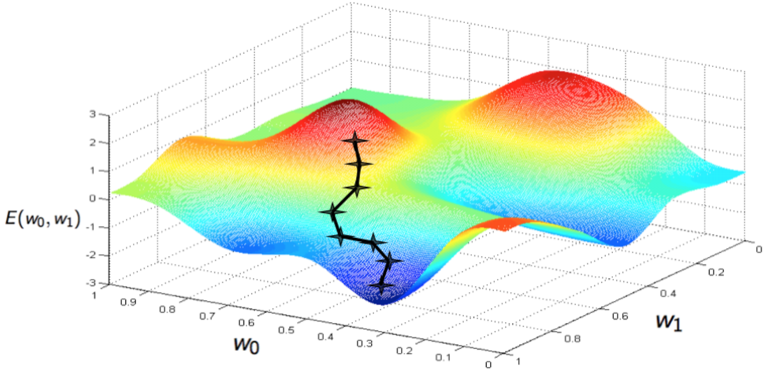
Incorrect

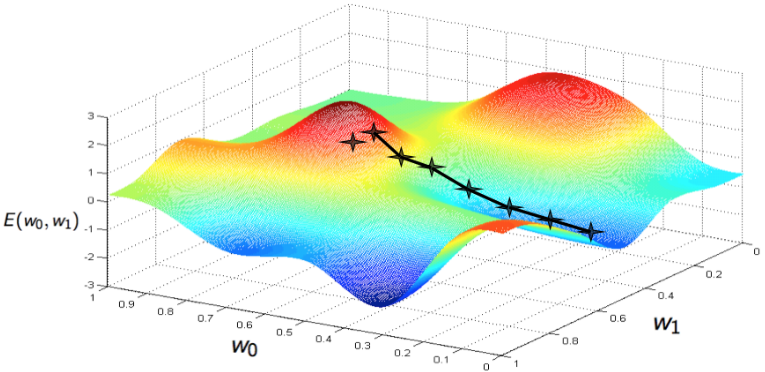
temp0 $:= w_0 - \alpha \frac{\partial}{\partial w_0} E(w_0, w_1)$
$w_0 :=$ temp0
temp1 $:= w_1 - \alpha \frac{\partial}{\partial w_1} E(w_0, w_1)$
$w_1 :=$ temp1

- 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, or even diverge.

General Cost Function

In the general case, cost function may have multiple local minima.




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

Gradient Descent for Linear Regression

Gradient Descent Algorithm for the Linear Regression Model

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

where

$h_w(x) = w_0 + w_1 x$
$E(w_0, w_1) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_w(x_i) - y_i\right)^2$

$$
\begin{aligned}
\frac{\partial}{\partial w_j} E(w_0, w_1) &= \frac{\partial}{\partial w_j} \frac{1}{2N} \sum_{i=1}^{N} (h_w(x_i) - y_i)^2 \\
&= \frac{\partial}{\partial w_j} \frac{1}{2N} \sum_{i=1}^{N} (w_0 + w_1 x_i - y_i)^2
\end{aligned}
$$

$$
\begin{aligned}
j = 0 : \frac{\partial}{\partial w_0} E(w_0, w_1) &= \frac{1}{N} \sum_{i=1}^{N} (h_w(x_i) - y_i) \\
j = 1 : \frac{\partial}{\partial w_1} E(w_0, w_1) &= \frac{1}{N} \sum_{i=1}^{N} (h_w(x_i) - y_i)\, x_i
\end{aligned}
$$

Gradient Descent for Linear Regression

Gradient Descent Algorithm for the Linear Regression Model

repeat until convergence {

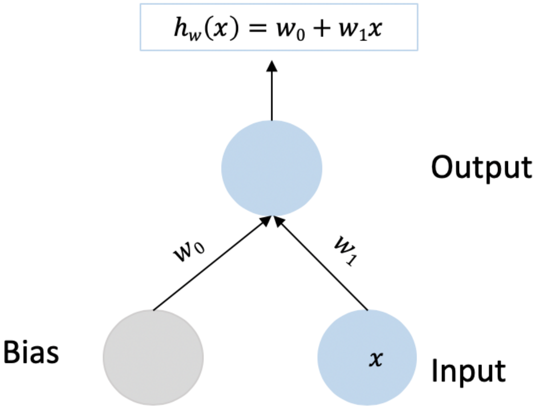
$w_0 := w_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} (h_w(x_i) - y_i)$

$w_1 := w_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} (h_w(x_i) - y_i)\, x_i$

}

Update w_0 and w_1 simultaneously.

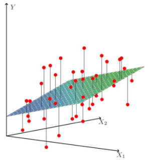
Univariate Linear Regression as a Neural Network: Linear Perceptron



Multivariate Linear Regression

Linear Regression: In Higher Dimensions

- **Analogy to line fitting**: In higher dimensions, fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?
- **Intuition**: Choose the one closest to the targets y
  - Linear regression uses the sum-of-squared error notion of closeness
- Similar intuition carries over to higher dimensions too
  - Fitting a D-dimensional hyperplane to the data (hard to visualize)
- The hyperplane is defined by parameters $\mathbf{w}$ (a $D \times 1$ weight vector)

Example - House Price Prediction

Multiple features (variables):

Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	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
- $x^{(i)}$ = input (features) of the i^{th} training example
- $x_j^{(i)}$ = value of feature j in the i^{th} training example

Model Representation

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

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

Model Representation

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

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$
$$h_w(x) = 80 + 0.1 x_1 + 0.001 x_2 + 3 x_3 + 2 x_4$$

Model Representation

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

$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$
$h_w(x) = 80 + 0.1 x_1 + 0.001 x_2 + 3 x_3 + 2 x_4$

More generally,

$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$
For convenience of notation, define $x_0 = 1$. Hence,

$$h_w(x) = \sum_{j=0}^{n} w_j x_j = \mathbf{w}^T \mathbf{x} = [w_0 w_1 \cdots w_n] \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$$

Linear Regression: In Higher Dimensions (Formally)

- Given training data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$
- Inputs $\mathbf{x}^{(i)}$: n-dimensional vectors (R^n), targets $y^{(i)}$: scalars (R)
- In the linear model: target is a linear function of the model parameters

$$y = h_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j = w_0 + \mathbf{w}^T \mathbf{x}$$

- $\mathbf{x} = [x_1, \cdots, x_n]$
- w_j's and w_0 are the model parameters (w_0 is an offset)
  - $\mathbf{w} = [w_1, \cdots, w_n]$, the **weight vector** (to be learned from $\mathcal{D}$)
  - Parameters define the mapping from the inputs to targets

Linear Regression: In Higher Dimensions (Formally)

- Given training data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$
- Inputs $\mathbf{x}^{(i)}$: n-dimensional vectors (R^n), targets $y^{(i)}$: scalars (R)
- In the linear model: target is a linear function of the model parameters

$$y = h_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j = w_0 + \mathbf{w}^T \mathbf{x}$$

- $\mathbf{x} = [x_1, \cdots, x_n]$
- w_j's and w_0 are the model parameters (w_0 is an offset)
  - $\mathbf{w} = [w_1, \cdots, w_n]$, the **weight vector** (to be learned from $\mathcal{D}$)
  - Parameters define the mapping from the inputs to targets

How to choose **w**?

Linear Regression: Gradient Descent Solution

- One solution: *Iterative* minimization of the cost function

  Cost Function:

$$E(w_0, w_1, \cdots, w_n) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right)^2,$$

  Goal:

$$\min_{w_0, w_1, \cdots, w_n} E(w_0, w_1, \cdots, w_n),$$

- How: Using Gradient Descent (GD)
- A general recipe for iteratively optimizing similar loss functions
- Gradient Descent rule:
  - Initialize the weight vector $\mathbf{w} = \mathbf{w}^0$
  - Update $\mathbf{w}$ by moving along the direction of negative gradient $-\frac{\partial E}{\partial \mathbf{w}}$

Linear Regression: Gradient Descent Solution

- Initialize $\mathbf{w} = \mathbf{w}^0$
- Repeat until convergence:

$$\begin{aligned} w_j &:= w_j - \alpha \frac{\partial E}{\partial w_j} \\ &:= w_j - \alpha \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}_j^{(i)} \end{aligned}$$

- Simultaneously update for every $j = 0, \cdots, n$
- α is the learning rate
- **Stop:** When some criteria is met (e.g., max. # of iterations), or the rate of decrease of E falls below some threshold.

Gradient Descent for Linear Regression

Gradient Descent for Univariate Linear Regression

$n = 1$

repeat until convergence {

$w_0 := w_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)}\right)$

$w_1 :=$
$w_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)}\right) x^{(i)}$

}

Update w_0 and w_1 simultaneously.

Gradient Descent for Multivariate Linear Regression

$n > 1$

repeat until convergence {

$w_0 :=$
$w_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)}\right) x_0^{(i)}$

$w_1 :=$
$w_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)}\right) x_1^{(i)}$

$w_2 :=$
$w_2 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)}\right) x_2^{(i)}$

$\vdots$

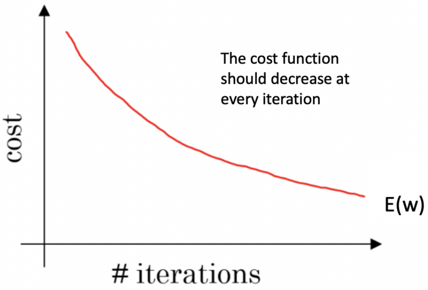
$w_n :=$
$w_n - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)}\right) x_n^{(i)}$

}

Update $w_0, w_1, \cdots, w_n$ simultaneously.

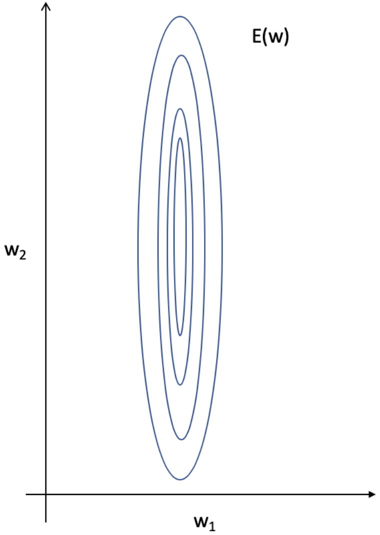
Practical Advise: Learning Rate

- To choose α, try:
  - $\cdots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \cdots$

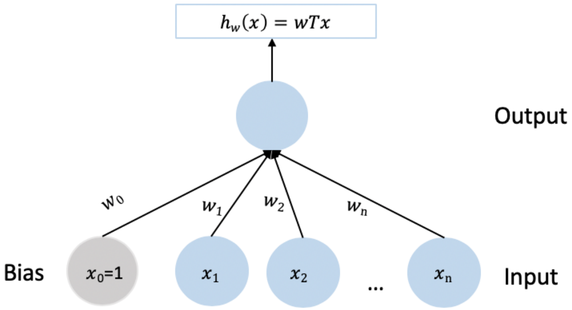


Practical Advice: Feature Scaling

- When using multivariate linear regression, make sure that the features are on a similar scale.
- Ideally, features should be in the $[-1, 1]$ interval.
- Otherwise, the gradient descent can take a long time to find the global minimum.
- E.g., if x_1 = size (0 - 2000 feet) and x_2 = number of bedrooms (1-5)



Multivariate Linear Regression as a Neural Network: Linear Perceptron



Summary

We discussed:

- Fundamental Problems of Learning
- Univariate Linear Regression
- Gradient Descent
- Multivariate Linear Regression