Linear Regression

Abdus Salam Azad

Supervised Learning

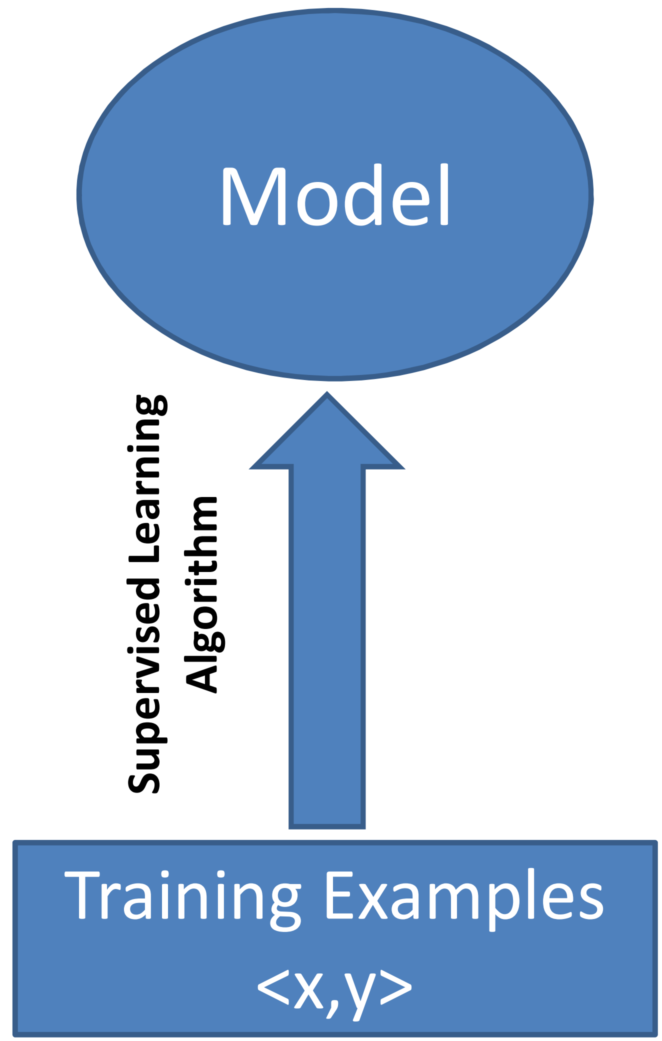
- “Classification” problem is a member of a broader class of Supervised Learning

Supervised Learning

- Given a **training set** of N example input-output pairs (x1, y1),(x2, y2), . . .(xN, yN)
  - each y_j was generated by an unknown function y = f(x)
  - x is a vector
  - Labeled dataset
- Discover a function h that approximates the true function f.
- The function h is called a **hypothesis**

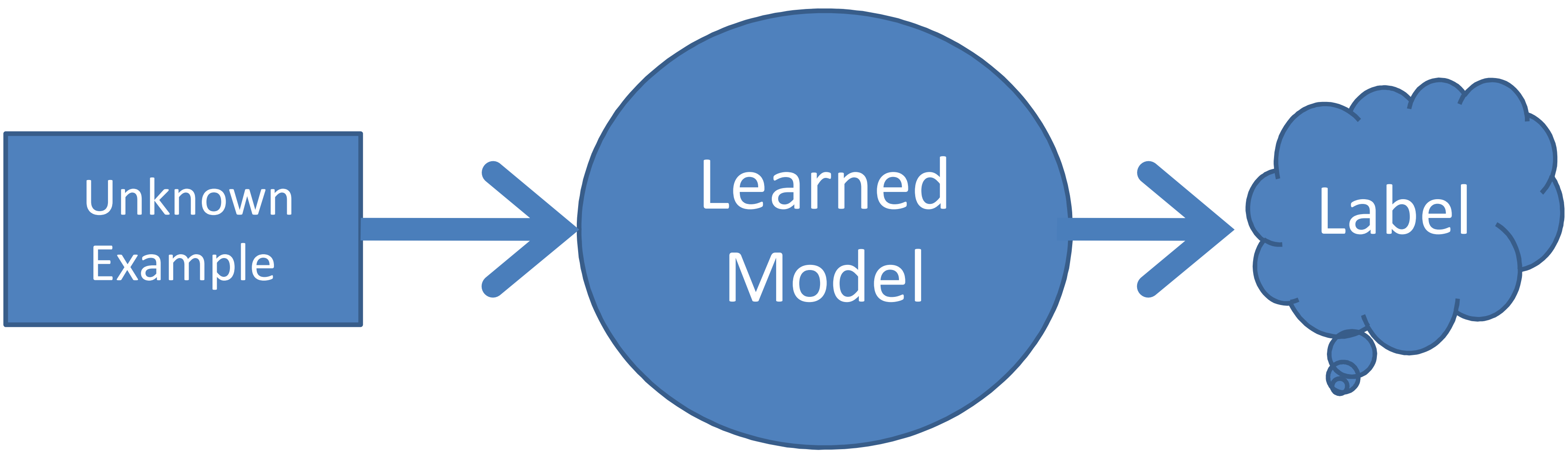
Supervised Learning - Training Phase

- Training Phase: The supervised learning algorithm first learn the model using the labeled example



Supervised Learning - Test Phase

Use the model to predict the label of an unknown example



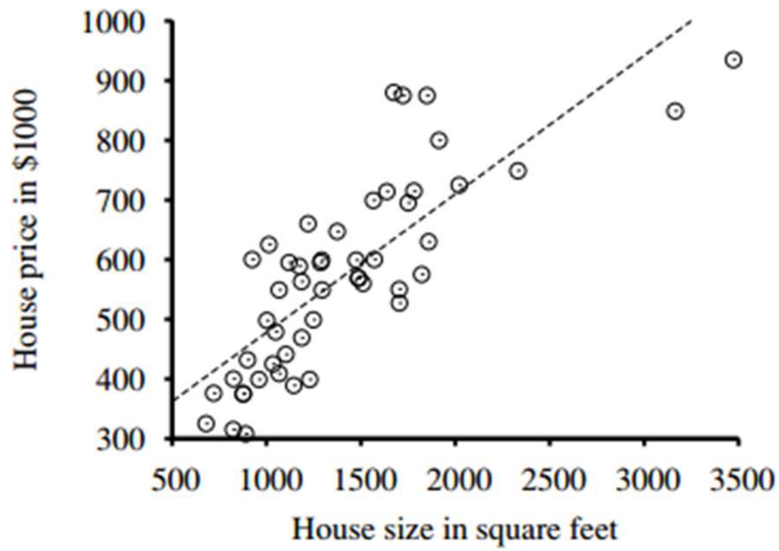
Let us consider a new problem!!!

- Given 100s examples of house price with its floor space, learn from the data a model which can predict the price of a house, given the floor space as input

House Size in Square Feet	House Price in $
750	400,000
2500	720,000
1800	600,000
…	…
…	…
…	…
3500	900,000

Let us consider a new problem!!!

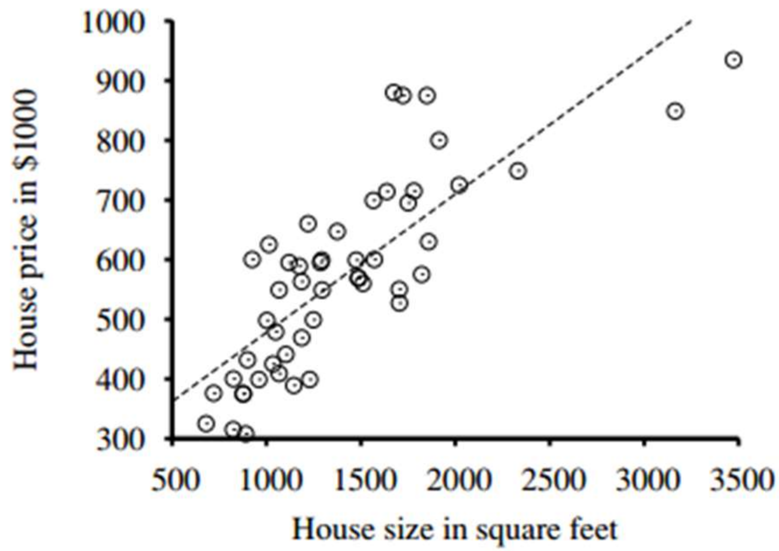
- Given 100s examples of house price with its floor space, learn from the data a model which can predict the price of a house, given the floor space as input



Have you seen any method/ tool that can solve this problem????

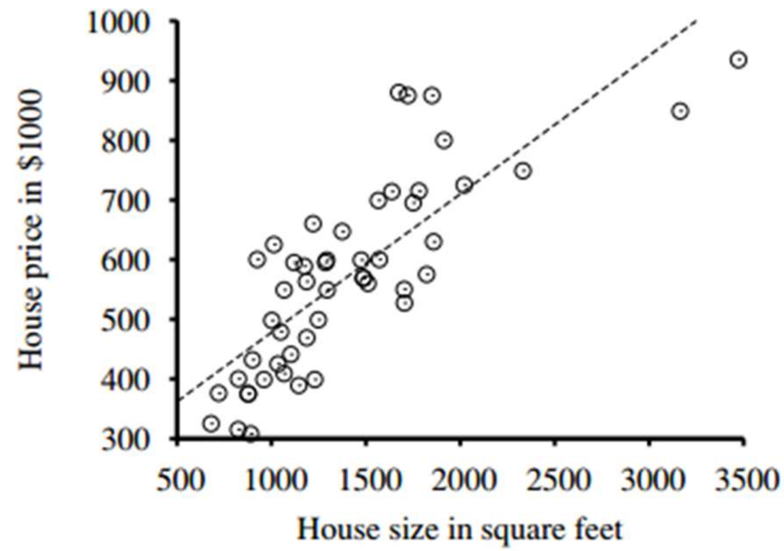
Univariate Linear Regression

- We will now consider hypotheses, that are linear functions of continuous values
  - Lines in n-dimensional space



Univariate Linear Regression

- For this particular problem we will “fit a straight line”
- The learned model will be a straight line in a 2-D space



Univariate linear regression

- A univariate linear function (a straight line) with input x and output y has the form
  $y = w_1 x + w_0$,
  - where w_0 and w_1 are real-valued coefficients to be learned.
- The $\mathbf{w}$s are called weights
- The weight vector $[w_0, w_1]$
  - Remember the term "weight vector", you will hear about this a lot

Hypothesis

- The Hypothesis is parameterized by the weight vector “w”

$$h_{\mathbf{w}}(x) = w_1 x + w_0$$

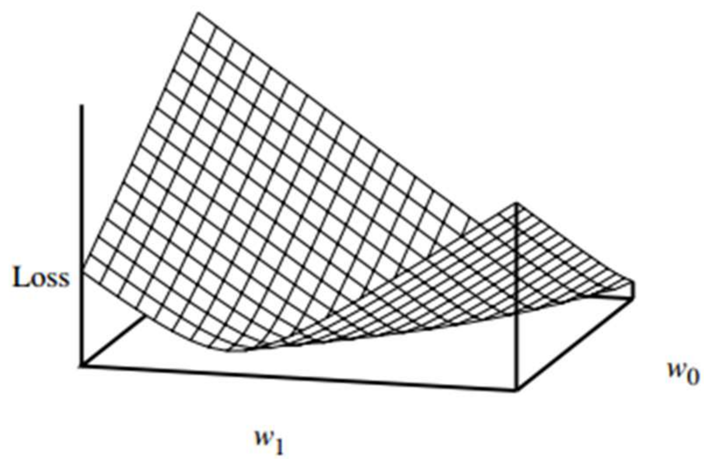
- We define the “loss function” as,

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

- We want to find $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} Loss(h_{\mathbf{w}})$

Weight Space

- Here, the loss function is **convex**
- implies that there are no local minima



Weight Space

- *every* linear regression problem with an L2 loss function is convex
- So, we can have closed forms
- However, non-linear models may not have any closed form, and hence we need a generic approach

Greedy Local Search

- As, the Loss Function is convex, there are no local minima
  - For every linear regression problem with L2 loss function, has convex loss functions
- Greedy Local Searches can find global optimum point too

Gradient Descent

$\mathbf{w} \leftarrow$ any point in the parameter space

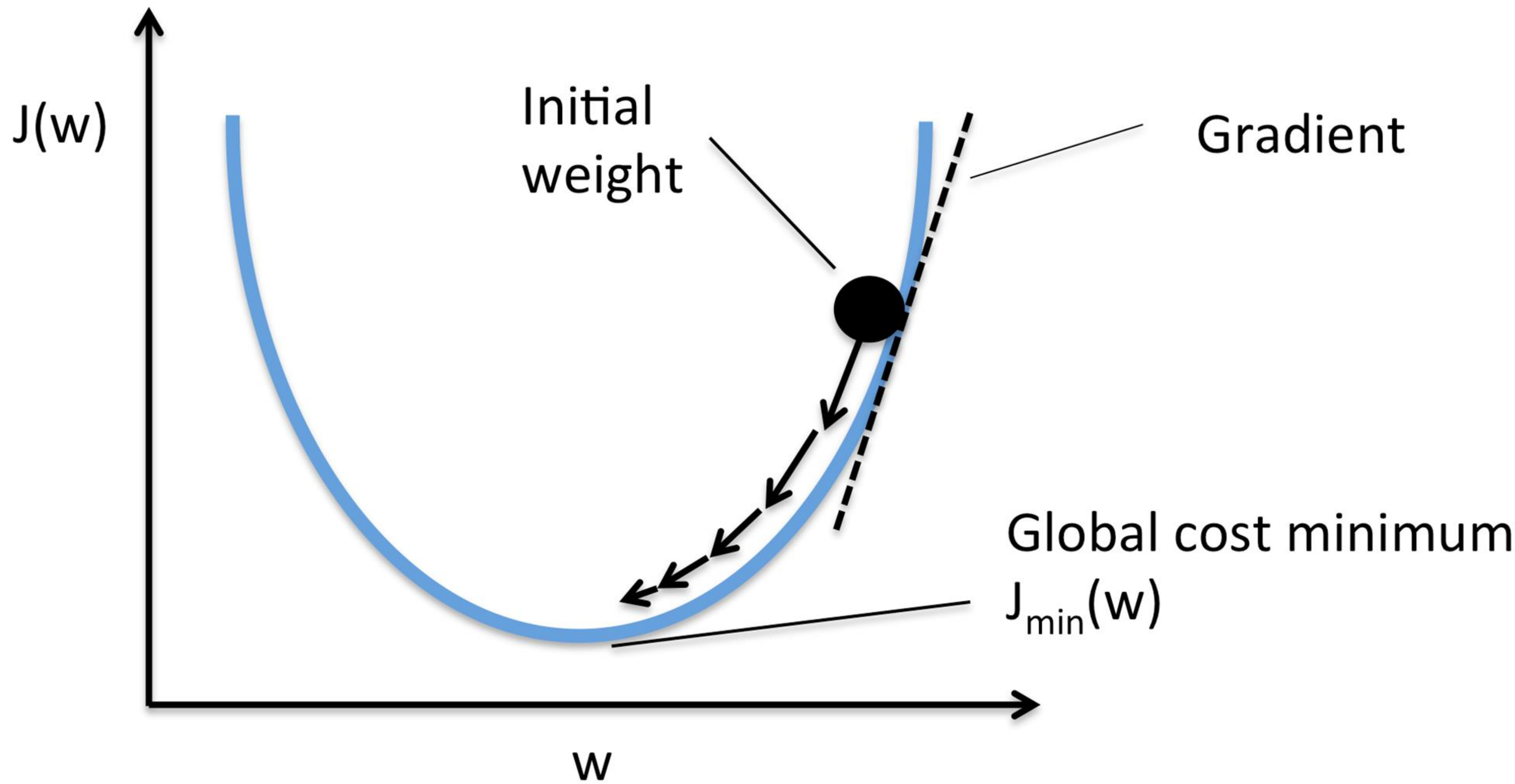
loop until convergence **do**

  for each w_i **in w do**

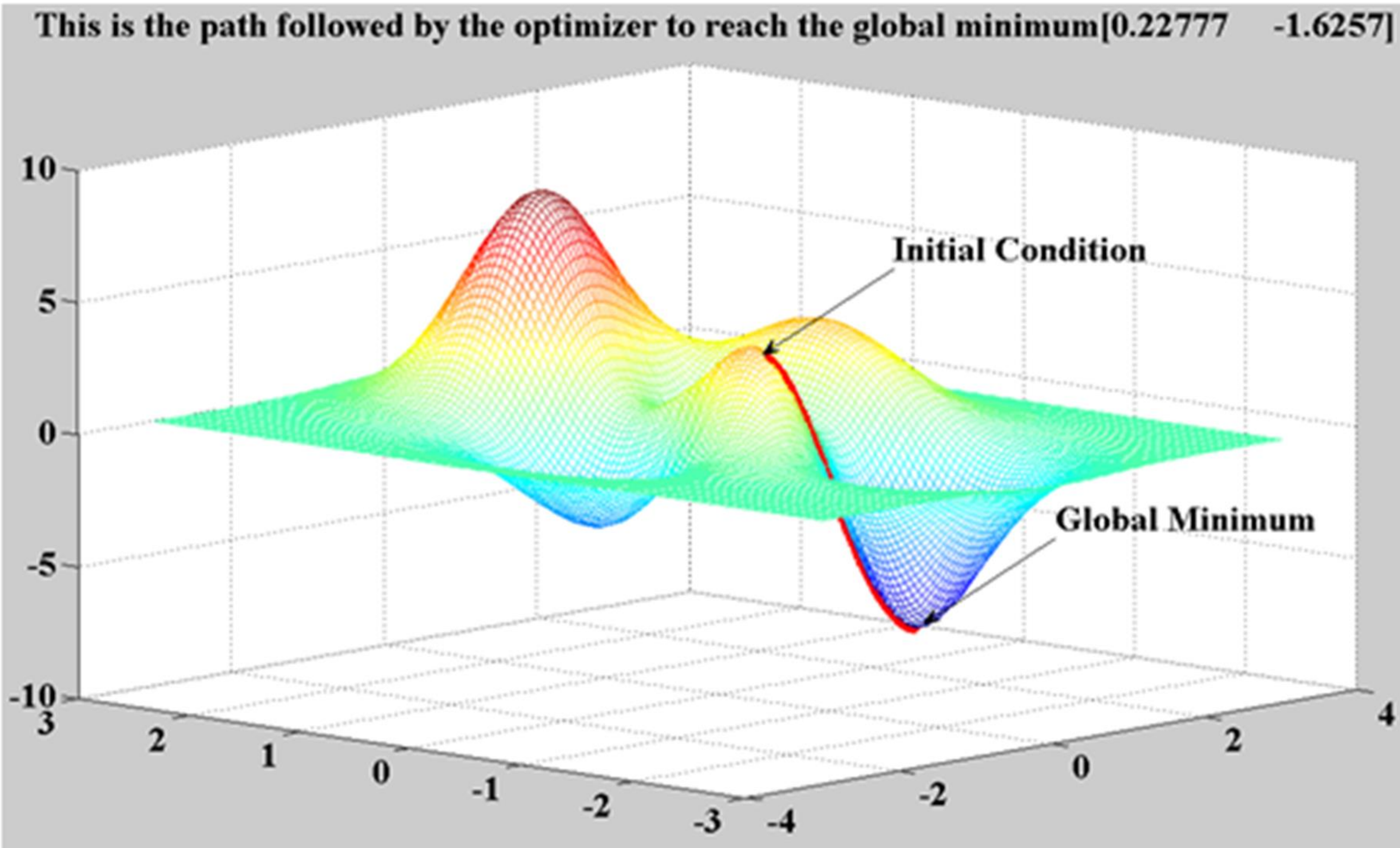
$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

- alpha is the learning rate

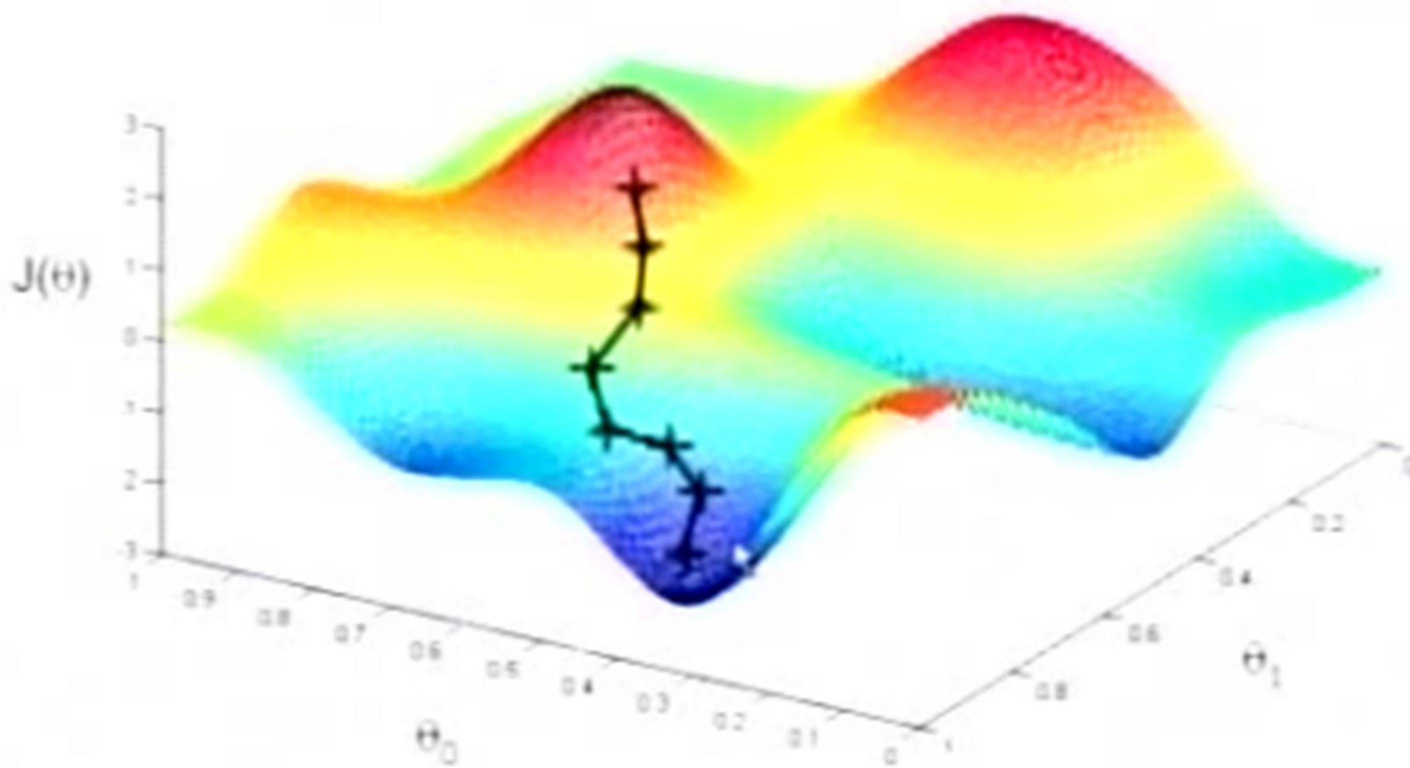
Gradient Descent



Gradient Descent



Gradient Descent



Calculating the derivatives

- Simplifying for only one training example

$$
\begin{aligned}
\frac{\partial}{\partial w_i} Loss(\mathbf{w}) &= \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2 \\
&= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x)) \\
&= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)) \,, \qquad (18.5)
\end{aligned}
$$

applying this to both w_0 and w_1 we get:

$$
\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x))\,; \qquad \frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \times x
$$

Then, plugging this back into Equation (18.4), and folding the 2 into the unspecified learning rate α, we get the following learning rule for the weights:

$$
w_0 \leftarrow w_0 + \alpha\,(y - h_{\mathbf{w}}(x))\,; \quad w_1 \leftarrow w_1 + \alpha\,(y - h_{\mathbf{w}}(x)) \times x
$$

Interpretation of the Gradient Descent Algorithm

$$w_0 \leftarrow w_0 + \alpha \, (y - h_{\mathbf{w}}(x)) \, ; \quad w_1 \leftarrow w_1 + \alpha \, (y - h_{\mathbf{w}}(x)) \times x$$

- Consider the cases
  - The prediction is lower: $h_{\mathbf{w}}(x) < y$
  - The prediction is higher: $h_{\mathbf{w}}(x) > y$
  - The prediction is equal: $h_{\mathbf{w}}(x) = y$
- How does the weight changes to fit the model ??

Batch Gradient Descent

- For N training examples, we want
  to minimize the sum of the individual losses for each example

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))\,; \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

- Convergence to the unique global minimum is guaranteed (as long as we pick α small enough)
- May be very slow: we have to cycle through all the training data for every step, and there may be many steps
- What happens if α is large???

Stochastic Gradient Descent

- Consider only a single training point at a time

- Cycle through the same training data as many times as is necessary, taking a step after considering each single example

- “Often” faster than batch gradient descent.

Stochastic Gradient Descent

- With a fixed learning rate α, however, it does not guarantee convergence;
  - it can oscillate around the minimum without settling down. In some cases, as we see later,

- A schedule of decreasing learning rates (as in simulated annealing) can guarantee convergence.

Stochastic Gradient Descent

- Can also be used in an “online” setting
  - where new data are coming in one at a time
  - And we need to update the model as new data arrives

Multivariate LR

$$h_{sw}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i}$$

- Introduce a dummy input attribute, $x_{j,0}$, which is defined as always equal to 1
  - Dot product of weight vector and input vector
  - Can also utilize matrix operations

$$h_{sw}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^\top \mathbf{x}_j = \sum_i w_i x_{j,i}$$

Multivariate LR

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \sum_j L_2(y_j, \mathbf{w} \cdot \mathbf{x}_j)$$

$$w_i \leftarrow w_i + \alpha \sum_j x_{j,i}(y_j - h_{\mathbf{w}}(\mathbf{x}_j))$$

Resource

- Russel & Norvig
  - Chapter 18: 18.6.1,18.6.2