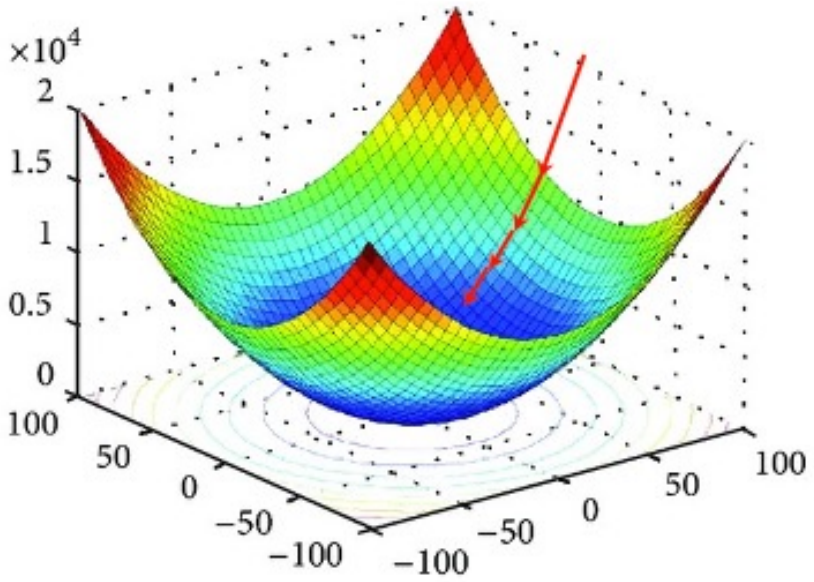
CS 505: Introduction to Natural Language Processing

Wayne Snyder
Boston University

Lecture 09: **Prelude to Deep Learning:** Linear Regression, Logistic Regression, Classification with Logistic Regression, Finding Solutions with Gradient Descent

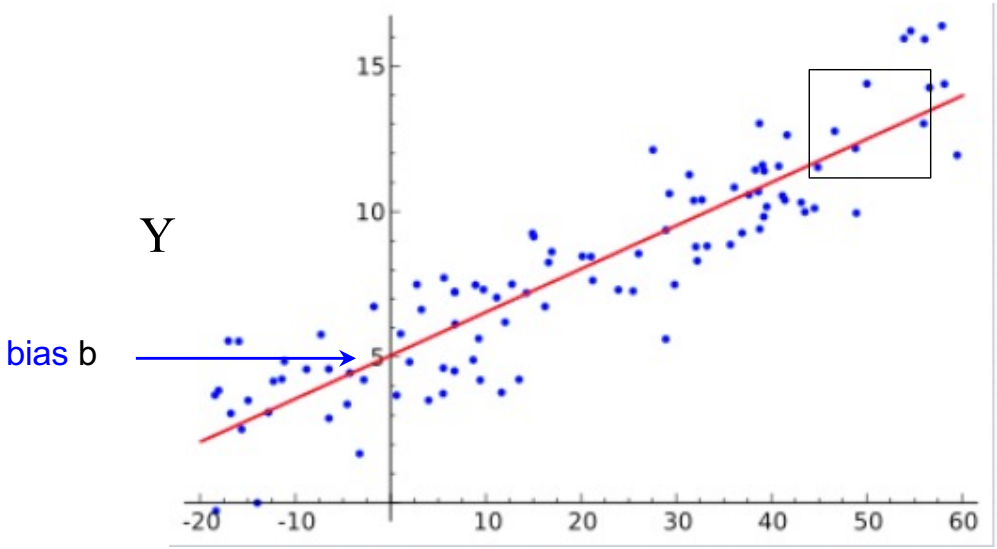


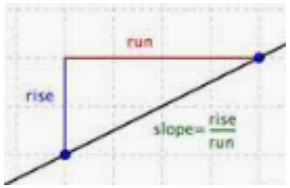
Linear Regression

Linear Regression relates some number of independent variables (real numbers)

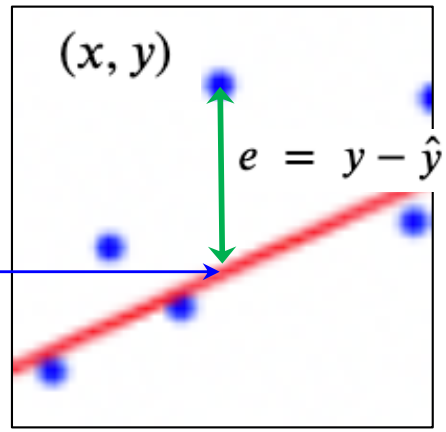
$$X_1,\ X_2,\ \ldots,\ X_n$$

with a dependent or response variable Y. The output is the set of parameters (here, slope m and bias b) showing the best approximation of the linear relationship of the variables.





$$\hat{y} = mx + b$$



Linear Regression

Linear regression can be calculated for any number of dimensions with a magically simple formula from linear algebra:

We thus have $Y = X \cdot W + E$ or

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_1^{(2)} & \dots & x_n^{(2)} \\ & & \vdots & & \\ 1 & x_1^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \times \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{bmatrix}.$$

The least-squares estimates for W are given by the following formula:

$$W = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = (X^T X)^{-1} X^T Y$$

Introduction to Deep Learning

Linear Regression: What is ”least” about the least-squares approximation of a line?

In linear regression, we define the error of the prediction as

the MSE (mean squared errors) of the predictions

$$\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 = \frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

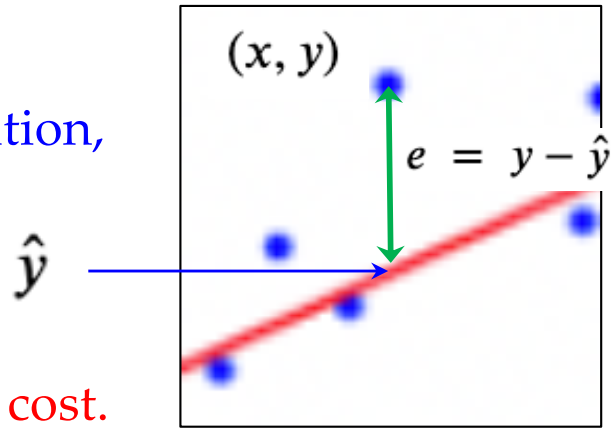
```
def MSE(X,Y,m):
    return np.mean( [ (Y[k] - m*X[k])**2 for k in range(len(X)) ] )
```

We seek the least amount of error in the approximation,

hence the “least squares” line.

In machine learning, we generally call this the

cost function, so we seek an approximation of least cost.



Introduction to Deep Learning

Linear Regression: Using Gradient Descent to find minimal-cost solution

The magic formula

$$W = (X^T X)^{-1} X^T Y$$

gives us an analytic solution with the smallest possible cost.

But what if there is no magic formula??

If there is no analytical solution (a formula), then we must use a search algorithm called **Gradient Descent** to find the parameter values which minimize this cost.

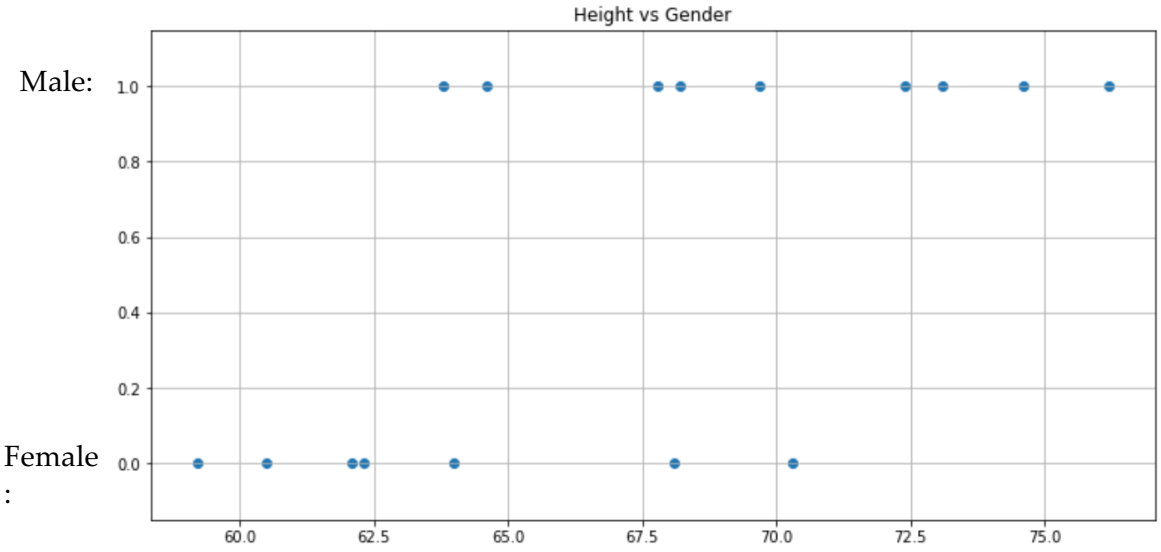
We'll return to Gradient Descent in a bit, but let's look at the most important application of regression to classification problems….

Introduction to Deep Learning

Logistic Regression: A Motivating Example

But linear regression doesn't work for many problems! Suppose we attempt to classify 16 people as male or female depending on a single feature: their height. Men in general are taller than women (the average height of an American man is 5′ 9″ and for women 5′ 4″),
X = height against Y = gender (1 for male, 0 for female):

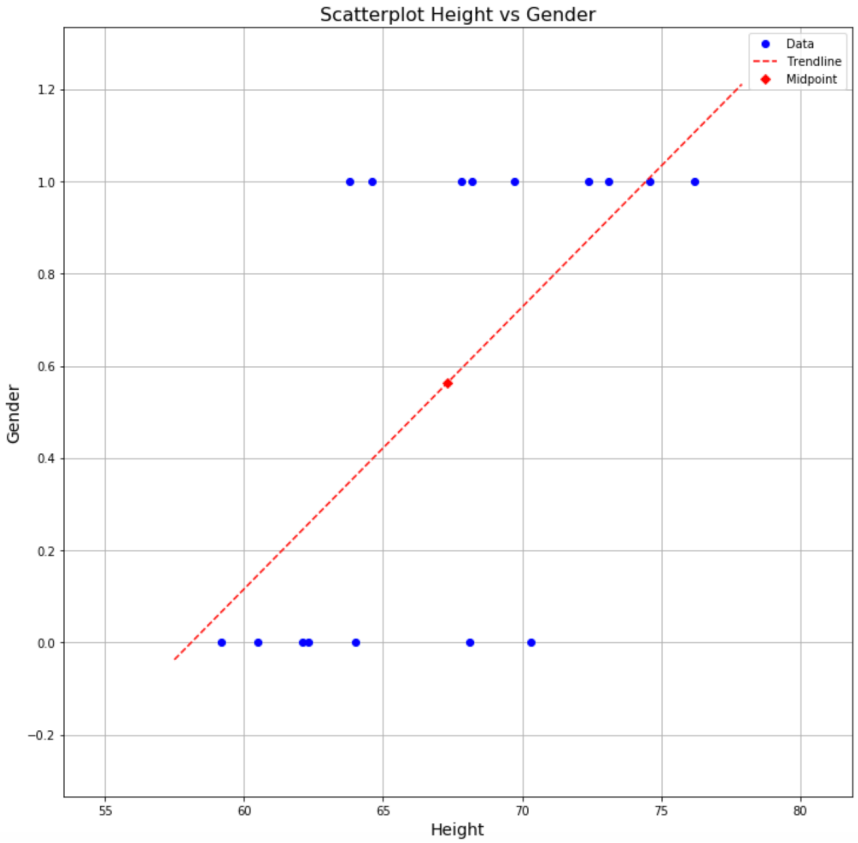
```
Heights: [59.2, 60.5, 62.1, 62.3, 63.8, 64.0, 64.6, 67.8, 68.1, 68.2, 69.7, 70.3, 72.4, 73.1, 74.6, 76.2]
Gender: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
```



Introduction to Deep Learning

Logistic Regression: Motivating Example

If we plug this into the linear regression algorithm, we get the following:



rho: 0.6187 r^2: 0.3828

Residual SS: 2.4303 Explained SS: 1.5072 Total SS: 3.9375

Regression Line: y = 0.0612 * x - 3.554

There are many issues with this:

How can we use this to predict someone's gender from their height?

How to give the probability of their gender?

There is clearly no linear trend, so what does the line even mean?

Introduction to Deep Learning

Logistic Regression: The Logit Transformation

In order to solve this, we will transform the scale of Y into a new domain, in this case into the real interval [0..1] used for probabilities. This is called the **Logit Transformation**, and is based on the notion of a **sigmoid function** $s : \mathcal{R} \rightarrow [0..1]$ of the form

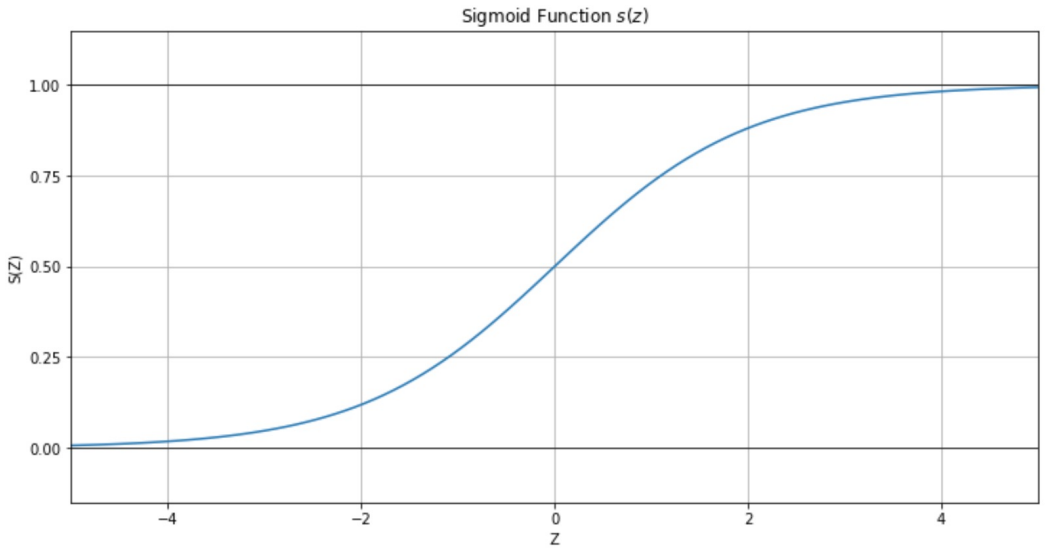
$$s(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

```
def s(z):
    return 1/(1+np.exp(-z))
```

$$\lim_{z \to \infty} \frac{1}{1 + e^{-z}} = 1$$

$$\lim_{z \to -\infty} \frac{1}{1 + e^{-z}} = 0$$

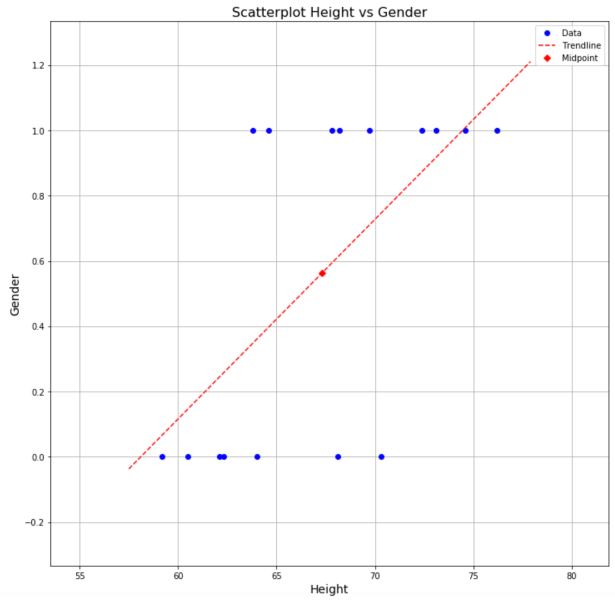
$$s(0) = \frac{1}{1 + e^0} = \frac{1}{1 + 1} = 0.5$$

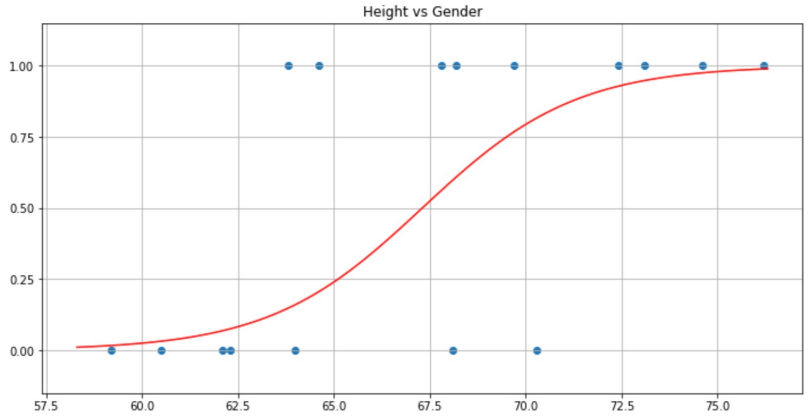


Logistic Regression: The Logit Transformation

Linear Regression: $-\infty < Y < \infty$

Logistic Regression: $0 \leq Y \leq 1$





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

$$y = \frac{e^x}{1+e^x}$$
$$\Leftrightarrow \quad y + ye^x = e^x$$
$$\Leftrightarrow \quad y = e^x - ye^x$$
$$\Leftrightarrow \quad y = (1-y)e^x$$
$$\Leftrightarrow \quad \frac{y}{(1-y)} = e^x$$
$$\Leftrightarrow \quad ln\left(\frac{y}{1-y}\right) = x$$
$$\Leftrightarrow \quad s^{-1}(p) = ln\left(\frac{p}{1-p}\right)$$

$s : \mathcal{R} \rightarrow [0..1]$

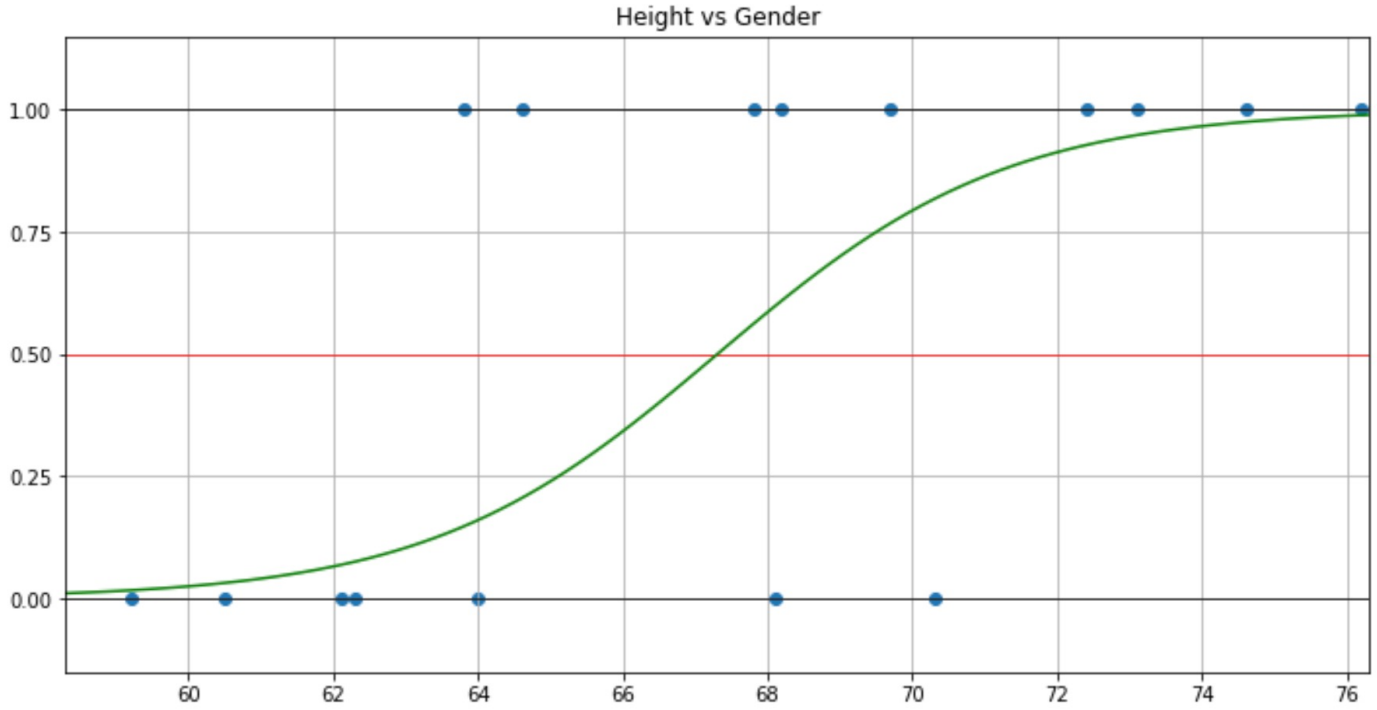
$s^{-1} : [0..1] \rightarrow \mathcal{R}$

$$s^{-1}(p) = ln\left(\frac{p}{1-p}\right)$$

This is called the "logit" or the "log odds ratio."

Logistic Regression: The Logit Transformation

The punchline here is that we will transform the regression line into a sigmoid, and use it to give us the probability that a given individual is male, and then define as a **decision boundary** a threshold (typically 0.5) by which we will decide if the binary output is 1 or 0:



Caveat: Such decision boundaries are typically not used in neural networks, so the output is between 0 and 1.

However, there is no analytical solution (no magic formula!) once we use the logit transformation, so we will need to use gradient descent to minimize an appropriate cost function.

Linear Regression Redux: Gradient Descent to find $\hat{\theta}_0$ and $\hat{\theta}_1$

In linear regression, we have explicit formulae for finding the parameters for the slope m and bias b of the regression line which minimizes the MSE.

But what if we didn't? We could then use an iterative approximation algorithm called **Gradient Descent** to find an approximation of the values which minimize the MSE.

Basic idea: Define a **cost or loss function J(...)** which gives the cost or penalty measuring how well the model parameters fit the actual data (high cost = bad fit), and then search for the parameters which minimize this cost.

$$J(\hat{m}, \hat{b}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - (\hat{b} + \hat{m}x_i))^2$$

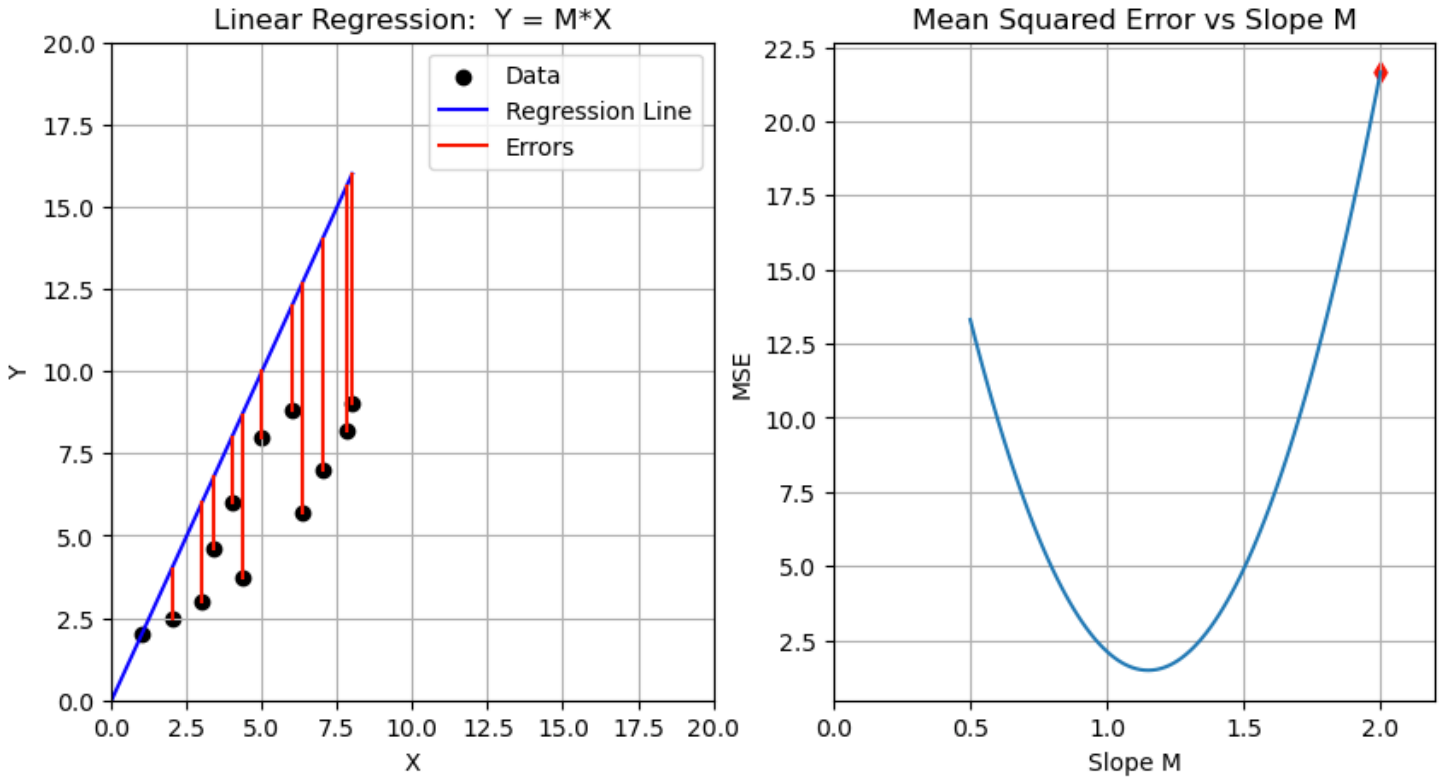
Cost Function
= MSE

The **J** in the cost function is used in machine learning and refers to the **J**acobian Matrix.

Reference: https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

Introduction to Deep Learning

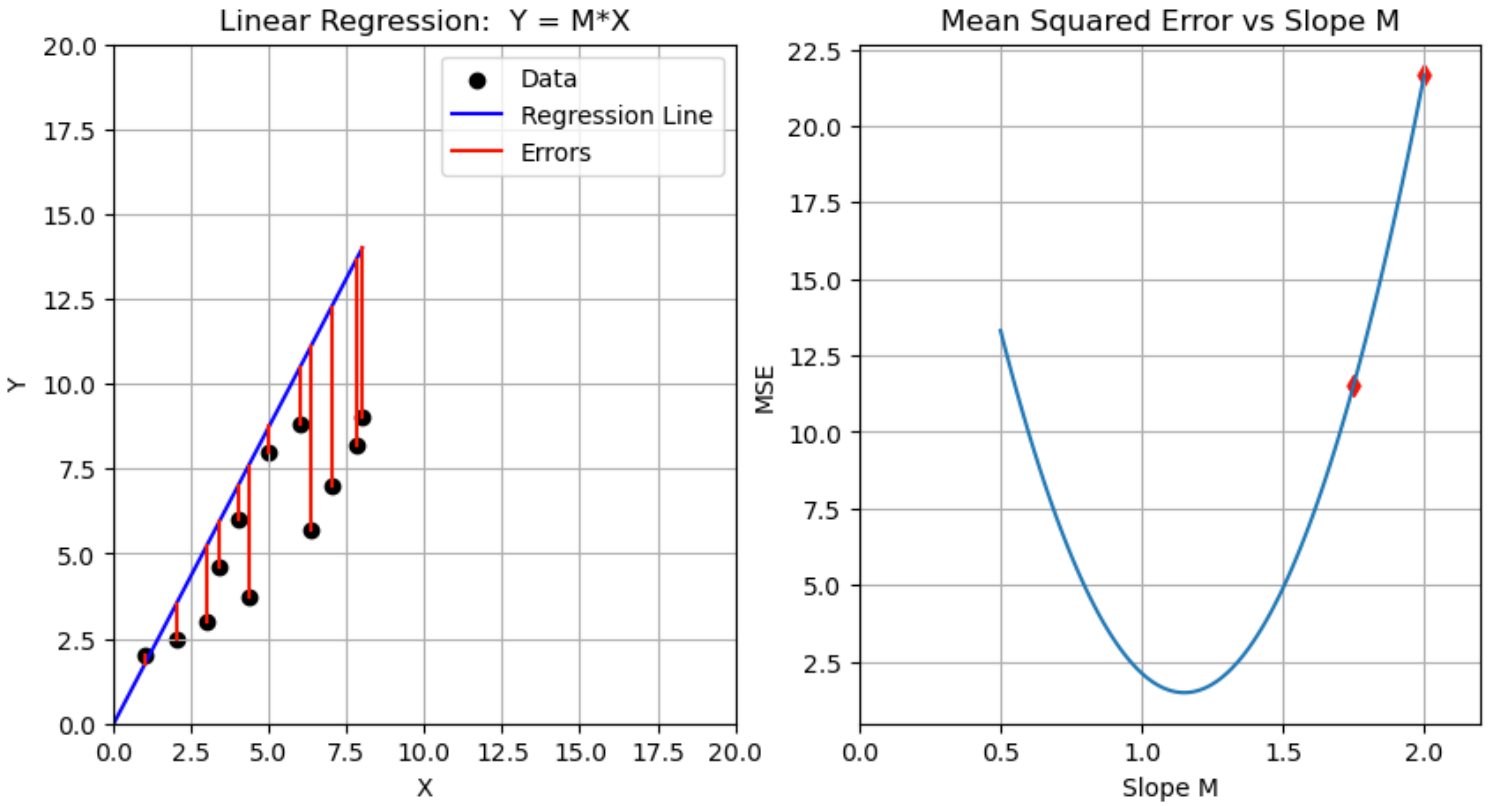
Here's a very simple example: Suppose we want to find a regression line satisfying **Y = m*X** (i.e., there is no bias term b). The cost function is quadratic, so we get a parabola when we graph the slope M against the MSE:




```
Slope M = 2
Mean Squared Error: 21.653042
```

Introduction to Deep Learning

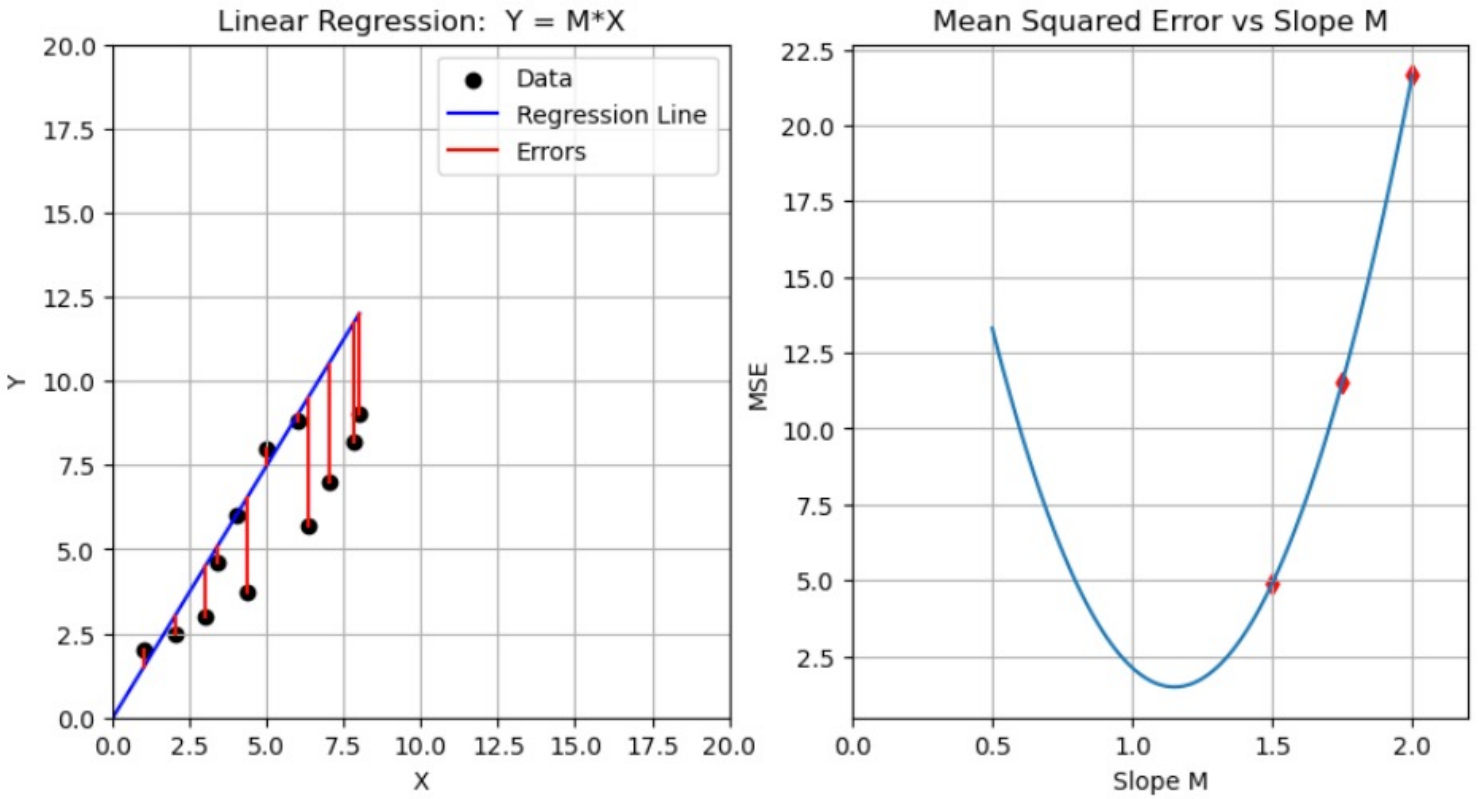
Gradient Descent is an iterative approximation algorithm, which "tweaks" the parameters to move in the direction of lower cost (smaller errors).




```
Slope M = 1.75
Mean Squared Error: 11.530544
```

Introduction to Deep Learning

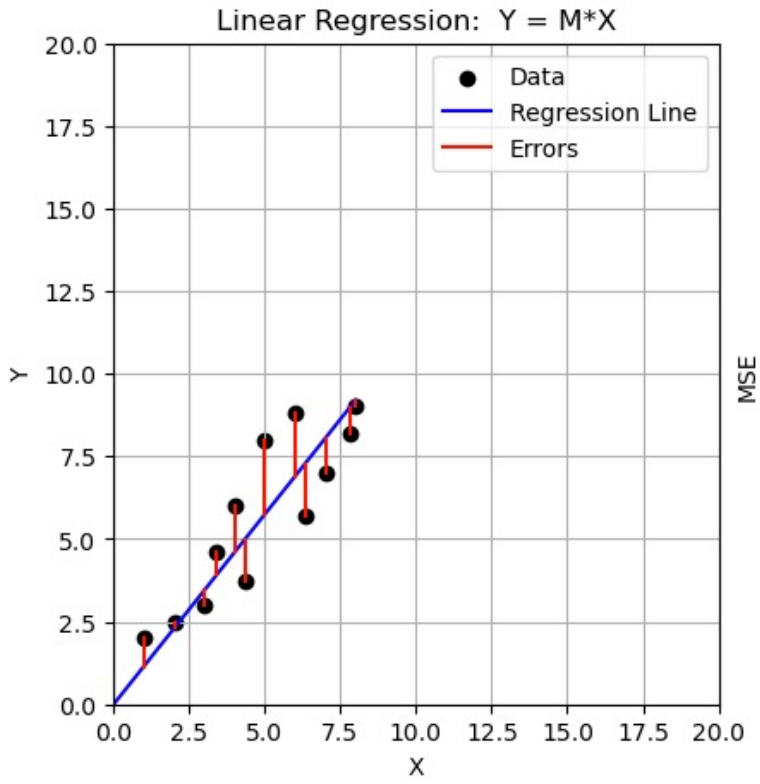
Gradient Descent is an iterative approximation algorithm, which "tweaks" the parameters to move in the direction of lower cost (smaller errors).

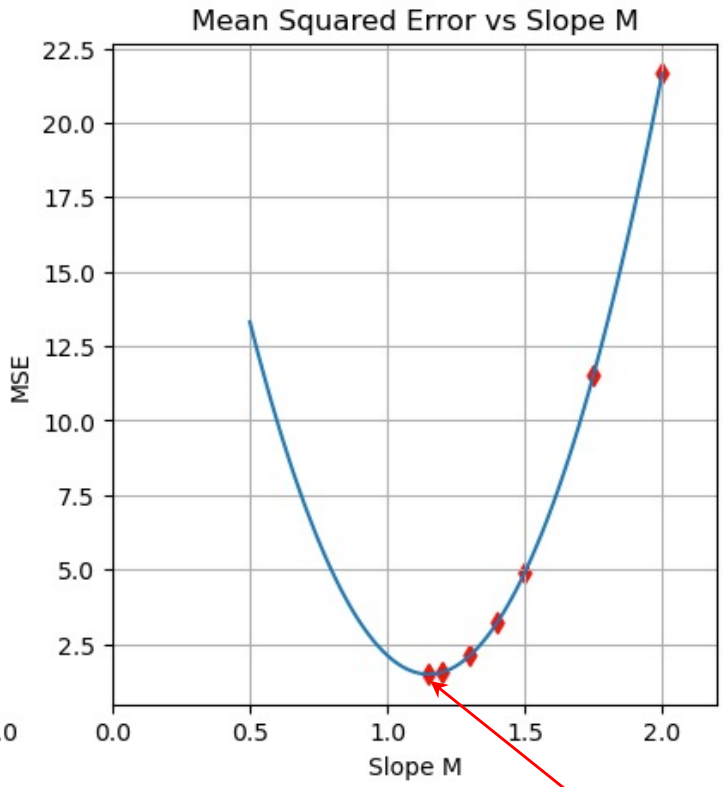



```
Slope M = 1.5
Mean Squared Error: 4.900802
```

Introduction to Deep Learning

Gradient Descent is an iterative approximation algorithm, which "tweaks" the parameters to move in the direction of lower cost (smaller errors).



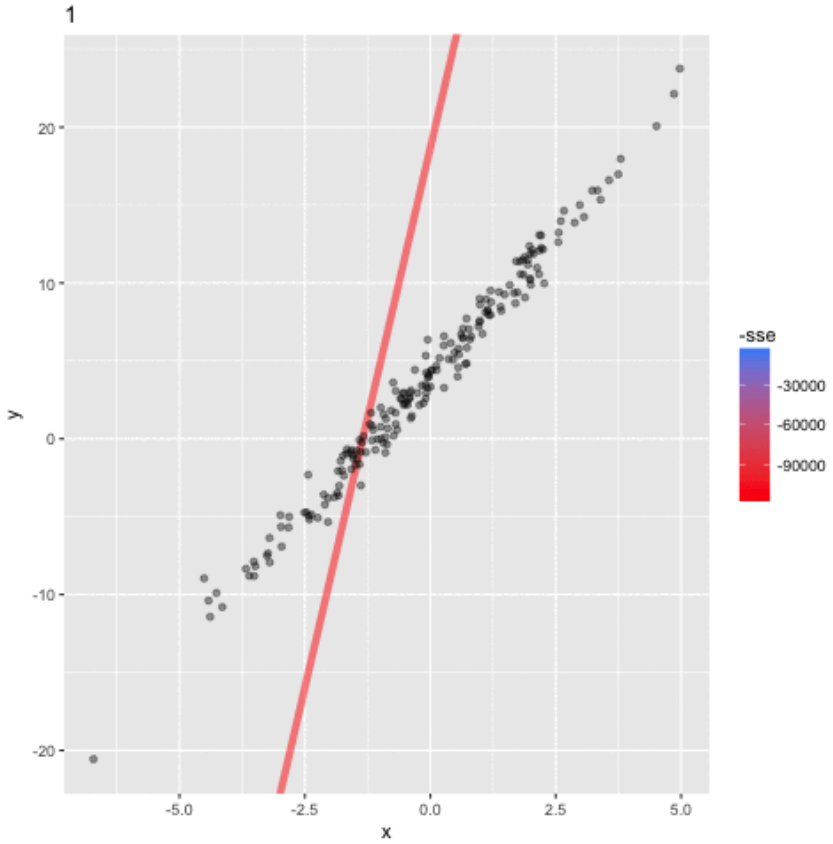


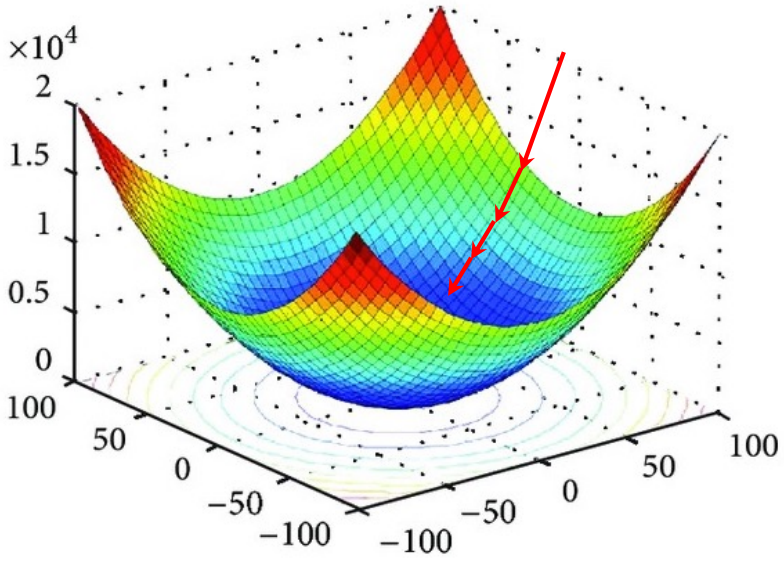

```
Slope M = 1.15045
Mean Squared Error: 1.486986
```

Least cost solution

Introduction to Deep Learning

With MSE, we always get a convex cost function, even in higher dimensions:



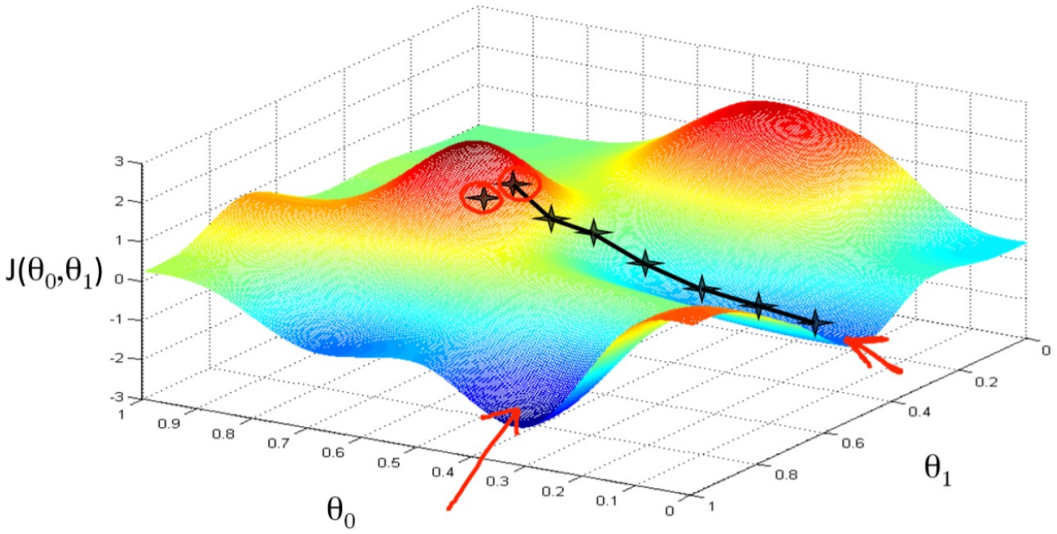


Linear Regression Redux: Gradient Descent

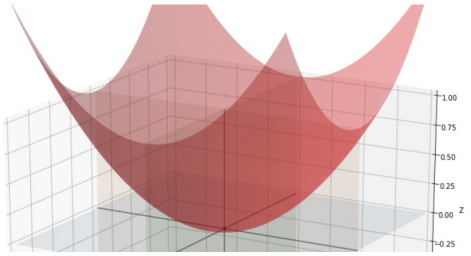
The Gradient Descent Algorithm: A **gradient** is a generalization of a derivative to functions of more than one variable:

> "Like the derivative, the gradient represents the slope of the tangent of the graph of the function. More precisely, the gradient points in the direction of the greatest rate of increase of the function, and its magnitude is the slope of the graph in that direction." - Wikipedia

In gradient descent, we pick a place to start, and move **down** the gradient until we find a minimum point:



When the search space is convex, such as a paraboloid, there will be a single minimum!



Another nice summary: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

Linear Regression Redux: Gradient Descent to find m and b

To find the minimum value along one axis we will work with only one of the partial derivatives as a time, say the bias b:

$$J'(b,m) = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} -2(y_i - (b + mx_i)) \\ \frac{1}{N}\sum_{i=1}^{N} -2x_i(y_i - (b + mx_i)) \end{bmatrix}$$

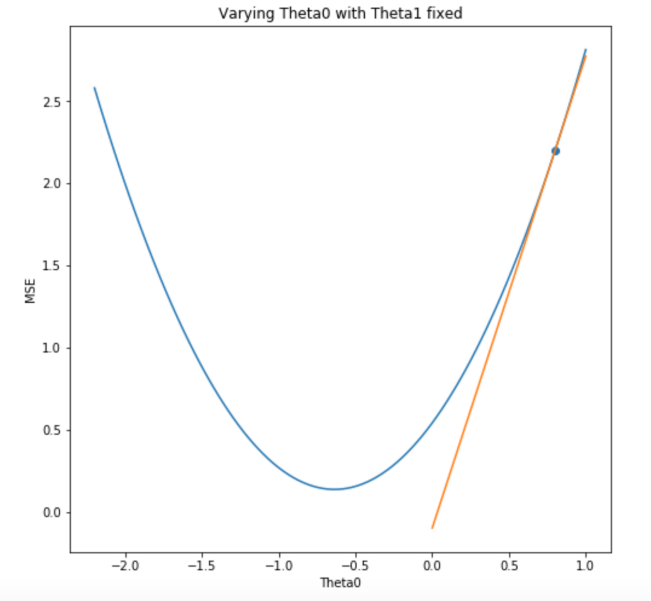
Partial derivative of cost function with respect to parameter b.

Step One: Choose an initial point b_0.

Step Two: Choose a step size or learning rate λ and threshold of accuracy ε.

Step Three: Move that distance along the axis, in the decreasing direction (the negative of the slope), and repeat until the distance moved is less than ε.

Step Four: Output b_{n+1} as the minimum.



1. Choose b_0;
2. Choose λ;
3. Repeat $b_{n+1} = b_n - J'(b_n) \cdot \lambda$
   Until $|b_{n+1} - b_n| < \epsilon$
4. Output b_{n+1}.

Linear Regression Redux: Gradient Descent to fi $\hat{\theta}_0$ a $\hat{\theta}_1$

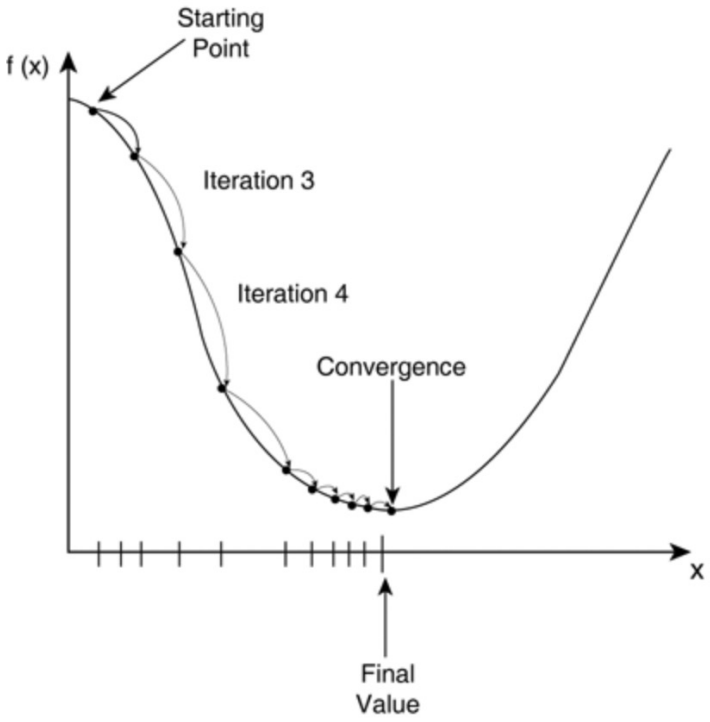
Gradient Descent for Linear Regression:

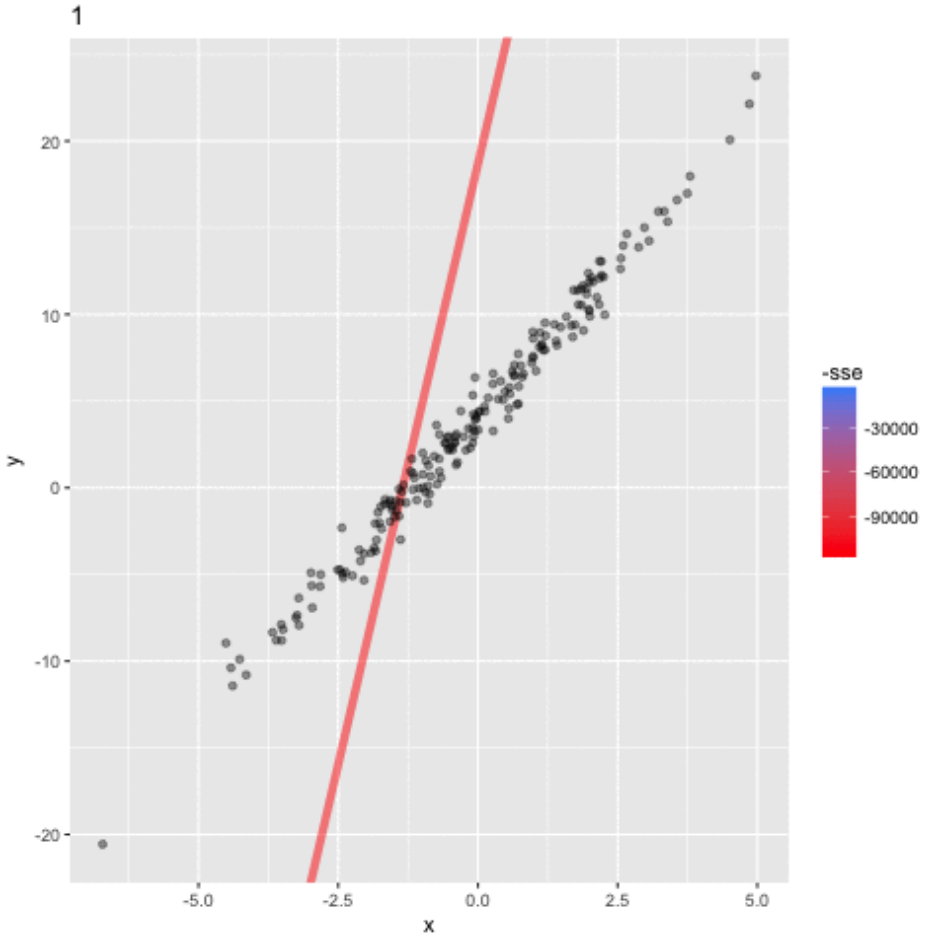
To find a point in multiple dimensions, we simply do all dimensions in the same way at the same time. Here is the algorithm:

```
def update_weights(m, b, X, Y, learning_rate):
    m_deriv = 0
    b_deriv = 0
    N = len(X)
    for i in range(N):
        # Calculate partial derivatives
        # -2x(y - (mx + b))
        m_deriv += -2*X[i] * (Y[i] - (m*X[i] + b))

        # -2(y - (mx + b))
        b_deriv += -2*(Y[i] - (m*X[i] + b))

    # We subtract because the derivatives point in direction of steepest ascent
    m -= (m_deriv / float(N)) * learning_rate
    b -= (b_deriv / float(N)) * learning_rate

    return m, b
```

Linear Regression Redux: Gradient Descent to fi $\hat{\theta}_0$ a $\hat{\theta}_1$

As the parameters are "tuned" to minimize the cost (= measuring how well the parameters fit the model) you get a better and better fit between the model and the data. You can run the gradient descent model as long as you wish to get a better fit. Obviously, defining the cost function and picking the learning rate and threshold are critical decisions, and much research has been devoted to different cost models and different approaches to gradient descent





Text Classification: Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

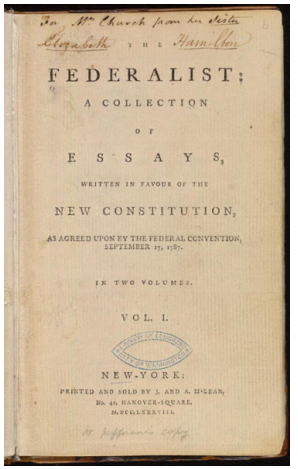
You can now access the latest news by using the link below to login to Stanford University News Forum.

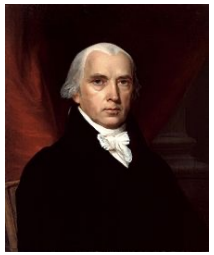
http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

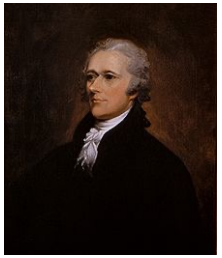
Who wrote which Federalist papers?

- 1787-8: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton.
- Authorship of 12 of the letters in dispute
- 1963: solved by Mosteller and Wallace using Bayesian methods


FEDERALIST;
A COLLECTION
OF
ESSAYS,
WRITTEN IN FAVOUR OF THE
NEW CONSTITUTION,
AS AGREED UPON BY THE FEDERAL CONVENTION,
SEPTEMBER 17, 1787.
IN TWO VOLUMES.
VOL. I.
NEW-YORK:
PRINTED AND SOLD BY J. AND A. M'LEAN,
No. 41, HANOVER-SQUARE.



James Madison



Alexander Hamilton

What is the subject of this medical article?

MEDLINE Article

?

MeSH Subject Category Hierar

- Antogonists and Inhibitors
- Blood Supply
- Chemistry
- Drug Therapy
- Embryology
- Epidemiology
- …

Positive or negative movie review?

\+ *...zany characters and richly applied satire, and some great plot twists*

– *It was pathetic. The worst part about it was the boxing scenes...*

\+ *...awesome caramel sauce and sweet toasty almonds. I love this place!*

– *...awful pizza and ridiculously overpriced...*

Positive or negative movie review?

\+ *...zany characters and richly applied satire, and some great plot twists*

– *It was pathetic. The worst part about it was the boxing scenes...*

\+ *...awesome caramel sauce and sweet toasty almonds. I love this place!*

– *...awful pizza and ridiculously overpriced...*

Text Classification: Definition

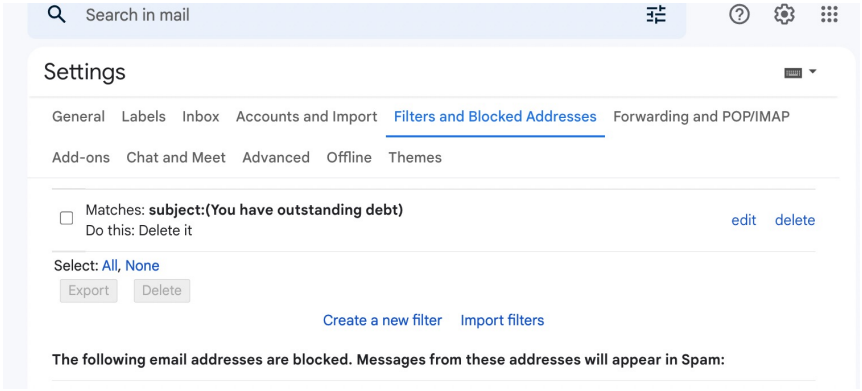
- ***Input*:**
  - a document *d*
  - a fixed set of labels/classes $C = \{c_1, c_2, \ldots, c_J\}$

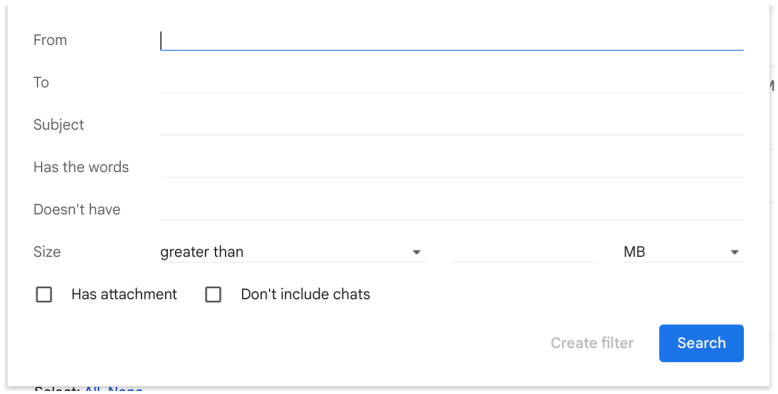
- ***Output*: a predicted class $c \in C$**

Caveats: In general, an algorithm will return probabilities for all document classes: this can be used to find the single best class, or—by setting a threshold or a bound on the number of classes—a set of classes.

Classification Methods: Hand-coded rules

- **Rules based on combinations of words or other features**
  - spam: black-list-address OR (“dollars” AND “you have been selected”)
- **Accuracy can be high**
  - If rules carefully refined by expert
- **But building and maintaining these rules is expensive**





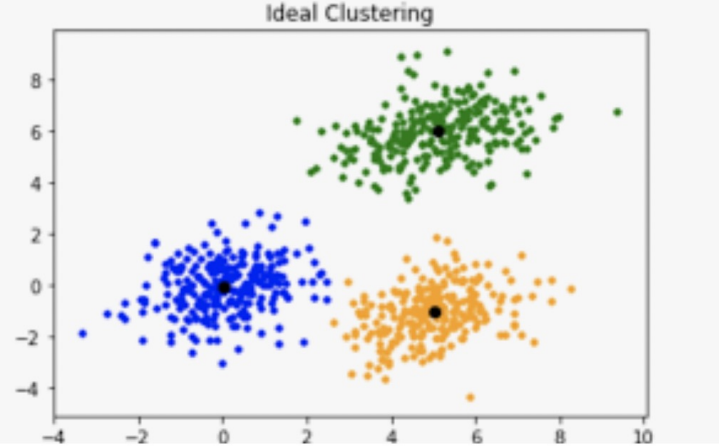
Classification Methods: Supervised ML

- ***Input:***
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - a randomly-permuted set of labeled documents $(d_1,c_1),\ldots,(d_n,c_n)$ split into
    - a training set $(d_1,c_1),\ldots,(d_m,c)$
    - a testing set $d_{m+1},\ldots,d_n$ (labels withheld)
- ***Output:***
  - A classifier $\gamma : d \rightarrow c$ trained the training set
  - The testing set with labels calculated by γ
  - Test results (confusion matrix, metrics, etc.)

Classification Methods: Supervised ML

- **There are many different kinds of classifiers for labeled data**
  - Naïve Bayes
  - Logistic regression
  - Neural networks

Classification Methods: Unsupervised ML

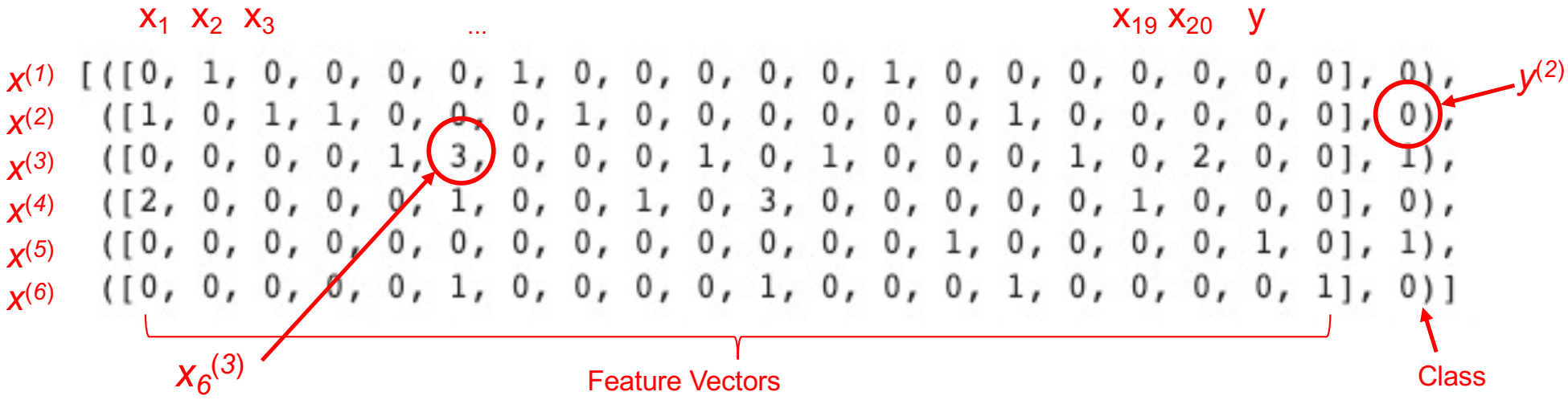


- ***Input:***
  - A set of documents $d_1,\ldots,d_n$
  - Requested number *k* of class
- ***Output:***
  - A partition of the document set into classes *1, …, k*
  - List of *k* centroids (center-point of each cluster
  - Evaluation metrics (e.g., mean distance of cluster members from centroids)

Components of a probabalistic machine learning classifier

Given *m* input/output pairs $(x^{(i)}, y^{(i)})$:

1. **A feature representation of the input. For each input observation $x^{(i)}$, a vector of features $[x_1, x_2, \ldots, x_n]$. Feature j for input $x^{(i)}$ is x_j, more completely $x_j^{(i)}$, or sometimes $f_j(x)$.**



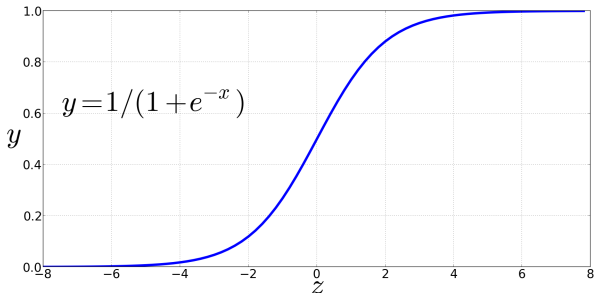
Components of a probabilistic ML classifier

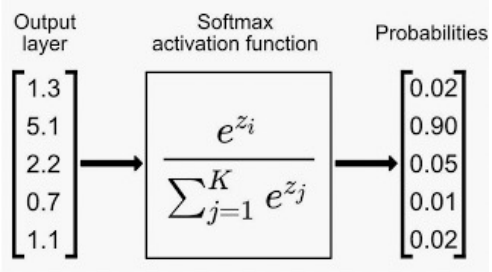
Given *m* input/output pairs $(x^{(i)}, y^{(i)})$:

1. A feature representation of the input. For each input observation $x^{(i)}$, a vector of features $[x_1, x_2, \ldots, x_n]$. Feature j for input $x^{(i)}$ is x_j, more completely $x_j^{(i)}$, or sometimes $f_j(x)$.
2. A classification function, like the sigmoid or softmax function, that uses weights $W = [w_1, w_2, \ldots, w_n]$ and b for each feature to calculate the probability for each possible class $\hat{y}$,

$$z = W \cdot x + b$$

$$P(y|x) = s(z)$$

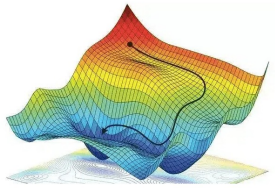




Components of a probabilistic machine learning classifier

Given *m* input/output pairs $(x^{(i)}, y^{(i)})$:

1. **A feature representation of the input. For each input observation $x^{(i)}$, a vector of features $[x_1, x_2, \ldots, x_n]$. Feature j for input $x^{(i)}$ is x_j, more completely $x_j^{(i)}$, or sometimes $f_j(x)$.**
2. **A classification function, like the sigmoid or softmax function, that uses weights $W = [w_1, w_2, \ldots, w_n]$ and b for each feature to calculate the probability for each possible class $\hat{y}$.**
3. **A learning algorithm to find the weights W and b from the training set:**
   - An objective/cost function, for estimating the errors in classification, e.g., **cross-entropy loss**.
   - A search algorithm using the objective function to find the W with least error: **stochastic gradient descent.**



The Phases of Logistic Regression: