Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

Topic 1 Subcategory of Supervised Learning Linear Model

Linear Regression

http://cs229.stanford.edu/notes/cs229-notes1.pdf
pages 82-86 *Learning From Data*
pages 19-22 & 107-118 *Hands-On Machine Learning with Scikit-Learn & TensorFlow*

A model assuming a linear relationship between the input variables and the output variable

Studied for over 200

CS/EE-UY 4563: INTRODUCTION TO MACHINE LEARNING

PROF. LINDA SELLIE

Some of the slides are from Prof. Sundeep Rangan

Some approaches are taken from CMU 18-661 Introduction to Machine Learning

Regression means real valued output

Used in statistics and Economics

THE TERM REGRESSION COMES FROM STATISTICS - WHENEVER HAVE A REAL VALUED OUTPUT WE CALL IT REGRESSION

Learning Objectives

- How to load data from a text file
- How to visualize data via a scatter plot
- Describe a linear model for data
  - Identify the label (target variable) and feature (predictor)
- Understand the objective function for linear regression
- Understand how partial derivatives can be used in a convex optimization problem
- Optimization:
  - Compute optimal parameters for the model using a closed form solution
  - Compute the optimal parameters for the model using gradient descent
- Evaluation:
  - Able to compute R^2
  - Able to visually determine goodness of fit and identify different causes for poor fit

Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
- …

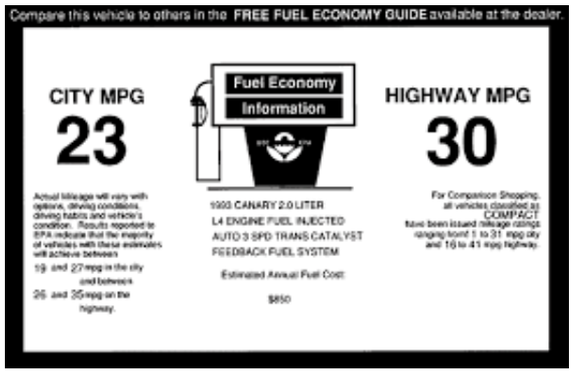
Predicting Trends

- Example: Predicting mpg for a car

mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
?	8	383.0	170.0	3563.0	10.0	70	1	dodge challenger se

Example: What Determines mpg in a Car?

- What engine characteristics determine fuel efficiency?
- Why would a data scientist be hired to answer this question?
  - Not to help purchasing a specific car
  - To guide building new cars
  - To find cars that are outside the trend



Choose the average MPG?

Keeping it simple

- Lets assume that *one* of the features can approximately predict the MPG

Lets plot the data

Demo on NYU Classes

Simple Linear Regression for Automobile mpg Data

Loading the Data

The python pandas library is a powerful package for data analysis. In this course, we will use a small portion of its features -- just reading and writing data from files. After reading the data, we will convert it to numpy for all numerical processing including running machine learning algorithms.

We begin by loading the packages.

In [86]:
```
import pandas as pd
import numpy as np
```

The data for this demo comes from a survey of cars to determine the relation of mpg to engine characteristics. The data can be found in the UCI library: https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg

You can directly read the data in the file, https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data We will load the data into ipython notebook, using the pandas library. Unfortunately, the file header does not include the names of the fields,

Pandas make it easy to work with two dimensional tables

The posted demo shows how to:

- Load data from a text file using the pandas package
- Create a scatter plot of data
- Handle missing data
- Fit a simple linear model
- Plot the linear fit with the test data
- Use a nonlinear transformation for an improved fit

Getting Data

- Data from UCI dataset library: [UCI Machine Learning Repository]

https://archive.ics.uci.edu/ml/datasets.html?format=&task=reg&att=&area=&numAtt=&num

Apps | Suggested Sites | Web Slice Gallery | Import to Mendeley | Other bookma

Browse Through: **61** Data Sets — Table View | List View

Default Task - Undo
- Classification (257)
- Regression (61)
- Clustering (52)
- Other (51)

Attribute Type
- Categorical (1)
- Numerical (52)
- Mixed (6)

Data Type
- Multivariate (56)
- Univariate (4)
- Sequential (5)
- Time-Series (16)
- Text (4)
- Domain-Theory (3)
- Other (0)

Area
- Life Sciences (9)
- Physical Sciences (8)
- CS / Engineering (25)
- Social Sciences (7)

Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
3D Road Network (North Jutland, Denmark)	Sequential, Text	Regression, Clustering	Real	434874	4	2013
Air Quality	Multivariate, Time-Series	Regression	Real	9358	15	2016
Airfoil Self-Noise	Multivariate	Regression	Real	1503	6	2014
Amazon Access Samples	Time-Series, Domain-Theory	Regression, Clustering, Causal-Discovery		30000	20000	2011
Auto MPG	Multivariate	Regression	Categorical, Real	398	8	1993

← Today's lecture

Loading the Data in Jupyter Notebook
Try 1: The Wrong Way!

```
import pandas as pd
import numpy as np
```

```
In [67]: names = ['mpg', 'cylinders','displacement', 'horsepower',
                  'weight', 'acceleration', 'model year', 'origin', 'car name']
```

```
In [122]: df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data')
```

```
In [123]: df.head(6)
```

Out[123]:

	18.0 8 307.0 130.0 3504. 12.0 70 1 "chevrolet chevelle malibu"
0	15.0 8 350.0 165.0 3693. 11...
1	18.0 8 318.0 150.0 3436. 11...
2	16.0 8 304.0 150.0 3433. 12...
3	17.0 8 302.0 140.0 3449. 10...
4	15.0 8 429.0 198.0 4341. 10...
5	14.0 8 454.0 220.0 4354. 9...

- Python pandas library
  - pd.read_csv command
  - Reads URL or file location
- Creates a dataframe object
  - http://pandas.pydata.org/pandas-docs/stable/dsintro.html#datafram
- Problems:
  - Did not parse columns
    - All data in a single column
    - Read_csv assumes columns are delimited by commas
  - Mistakes first line as header

Loading the Data in Jupyter
Try 2: Fixing the Errors

```
In [125]: df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/'+
                           'auto-mpg/auto-mpg.data',
                           header=None,delim_whitespace=True,names=names,na_values='?')
```

You can display a first few lines of the dataframe by using head command:

```
In [126]: df.head(6)
```

Out[126]:

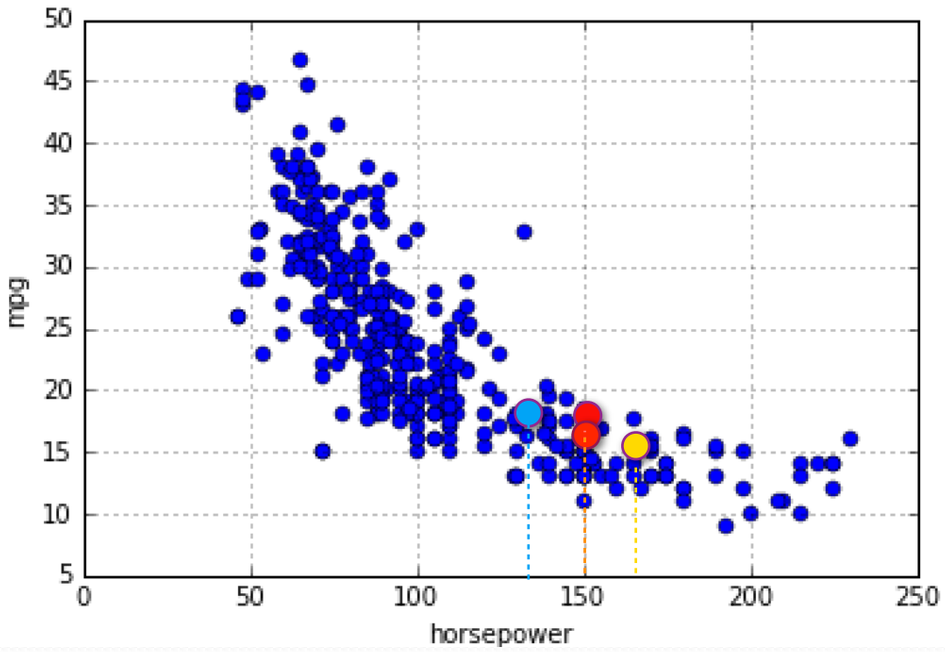
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18	8	307	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15	8	350	165	3693	11.5	70	1	buick skylark 320
2	18	8	318	150	3436	11.0	70	1	plymouth satellite
3	16	8	304	150	3433	12.0	70	1	amc rebel sst
4	17	8	302	140	3449	10.5	70	1	ford torino
5	15	8	429	198	4341	10.0	70	1	ford galaxie 500

- Fix the arguments in read_csv
- pd.read_csv has many options to deal with a wide variety of differently formatted data sets
- When you get a problem:
  - **Google is your friend!**
  - You are not the first to have these problems
  - Official documentation exists
- Ask questions on slack if you get stuck

Visualizing the Data using a scatter plot

```
In [150]: xstr = 'horsepower'
          X = np.array(df[xstr])
          y = np.array(df['mpg'])
```

- We will plot data in Python using Matplotlib
- A nice tutorial: https://matplotlib.org/users/pyplot_tutorial.html
- How could you predict the mpg of a car not in the data as a function of the horsepower? Pair share



$$X = \begin{bmatrix} 130 \\ 165 \\ 150 \\ 150 \\ \dots \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 18.0 \\ 15.0 \\ 18.0 \\ 16.0 \\ \dots \end{bmatrix}$$

$(\mathbf{x}^{(1)} = 130,\ y^{(1)} = 18)$

$(\mathbf{x}^{(2)} = 165,\ y^{(2)} = 15)$

$(\mathbf{x}^{(3)} = 150,\ y^{(3)} = 18)$

$(\mathbf{x}^{(4)} = 150,\ y^{(4)} = 17)$

...

$\mathbf{x}^{(1)} = 130 \quad \mathbf{x}^{(2)} = 165$

$\mathbf{x}^{(3)} = 150$

Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
- …

Approach

$$y \approx w_0 + w_1 \mathbf{x}$$

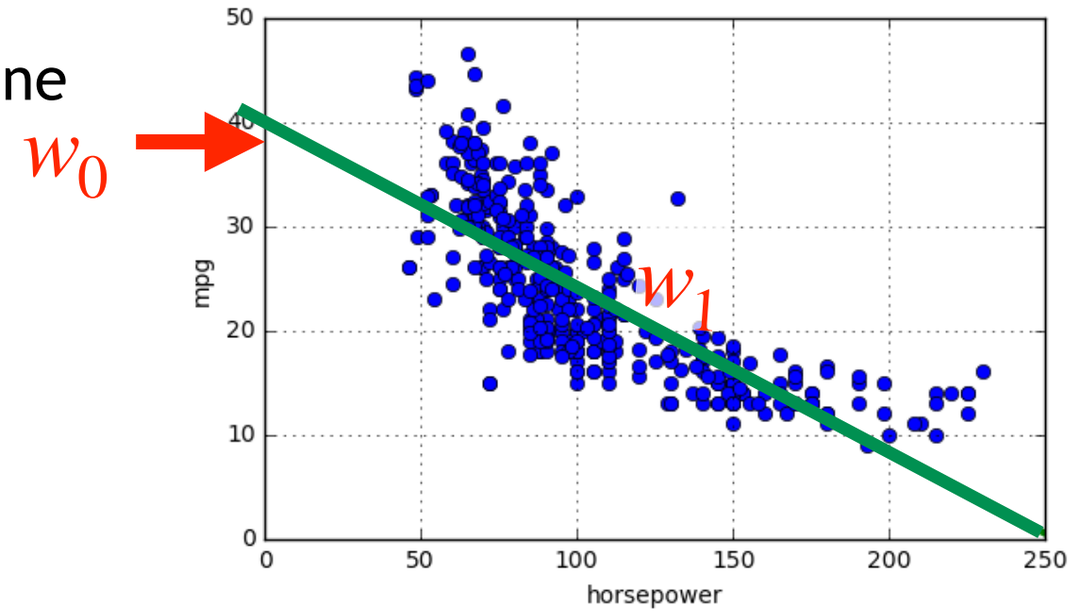
Warning! Stanford notes use θ instead of w for the parameters

MPG $\approx w_0 + w_1$ Horsepower

Learn parameters w_0, w_1 of the green line

Car 1: $w_0 + w_1 \cdot 130 = 18$

Car 2: $w_0 + w_1 \cdot 165 = 15$



We are changing the notation from what you are used to seeing when we describe a line.
Instead of $mx + b = y$, we are using $w_0 + w_1 \mathbf{x}$

Approach

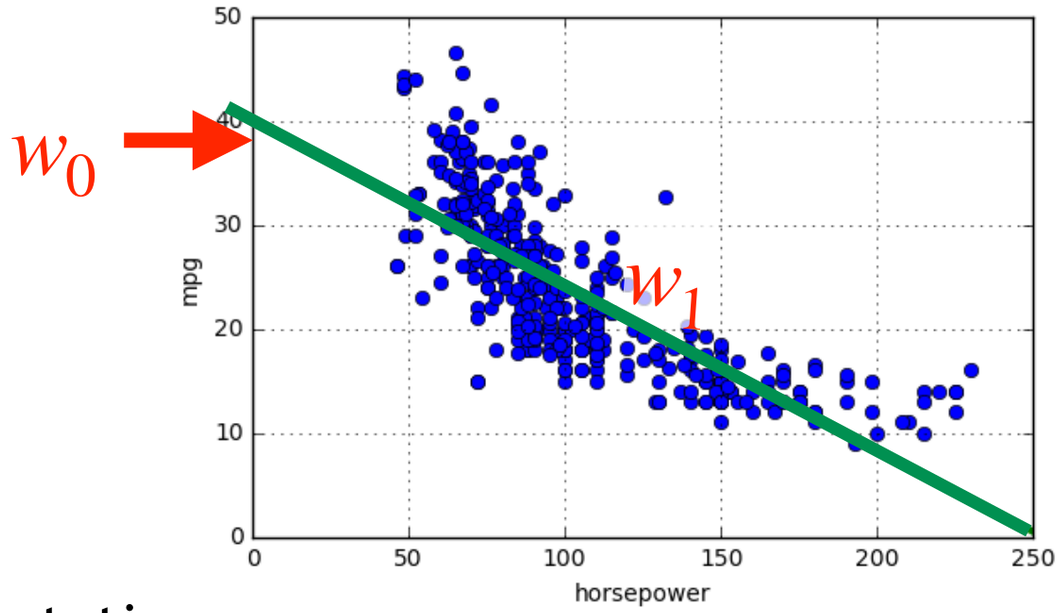
$$y \approx w_0 + w_1 \mathbf{x}$$

MPG $\approx w_0 + w_1$ Horsepower

Learn parameters w_0, w_1 of the green line

Car 1: $w_0 + w_1 \cdot 130 = 18$

Car 2: $w_0 + w_1 \cdot 165 = 15$



We can compactly represent this in matrix notation:

$$\begin{bmatrix} 1 & 130 \\ 1 & 165 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 18 \\ 15 \end{bmatrix}$$

Finding w_0, w_1 if we only use two examples

$$\begin{bmatrix} 1 & 130 \\ 1 & 165 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 18 \\ 15 \end{bmatrix}$$

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \left(\begin{bmatrix} 1 & 130 \\ 1 & 165 \end{bmatrix} \right)^{-1} \begin{bmatrix} 18 \\ 15 \end{bmatrix}$$

$$= \begin{bmatrix} 4.71 & -3.71 \\ -.03 & .03 \end{bmatrix} \begin{bmatrix} 18 \\ 15 \end{bmatrix}$$

$$= \begin{bmatrix} 29.14 \\ -0.09 \end{bmatrix}$$

If we use more data, how can we estimate w_0, w_1?

horsepower	mpg
130.0	18.0
165.0	15.0
150.0	18.0
150.0	16.0
...	...

Problem! There isn't a w_0, w_1 that will satisfy all the equations

Want to predict the best MPG based on the horsepower

MPG = $w_0 + w_1$horsepower + unexplainable_stuff

horsepower	mpg
130.0	18.0
165.0	15.0
150.0	18.0
150.0	16.0
…	…

Prediction car 1: $w_0 + w_1 \cdot 130$

Prediction car 2: $w_0 + w_1 \cdot 165$

Prediction car 3: $w_0 + w_1 \cdot 150$

Prediction car 4: $w_0 + w_1 \cdot 150$

…

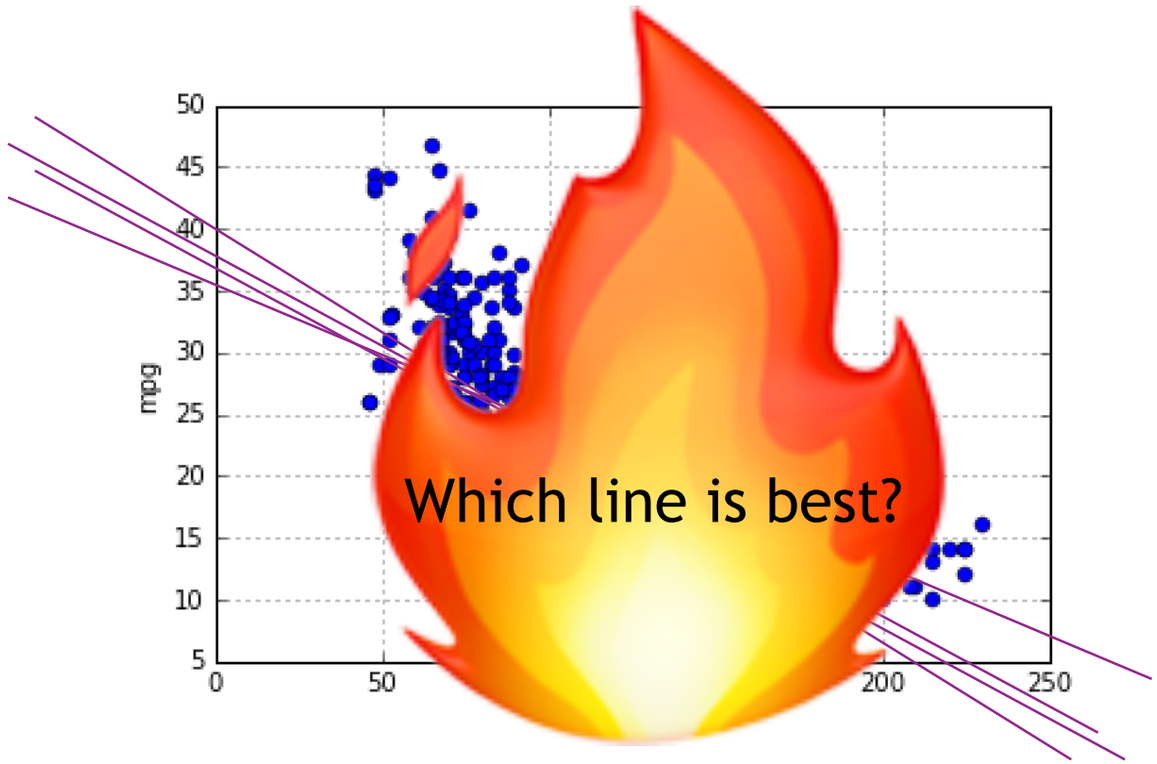
Learn w_0 and w_1

"Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.” George Box

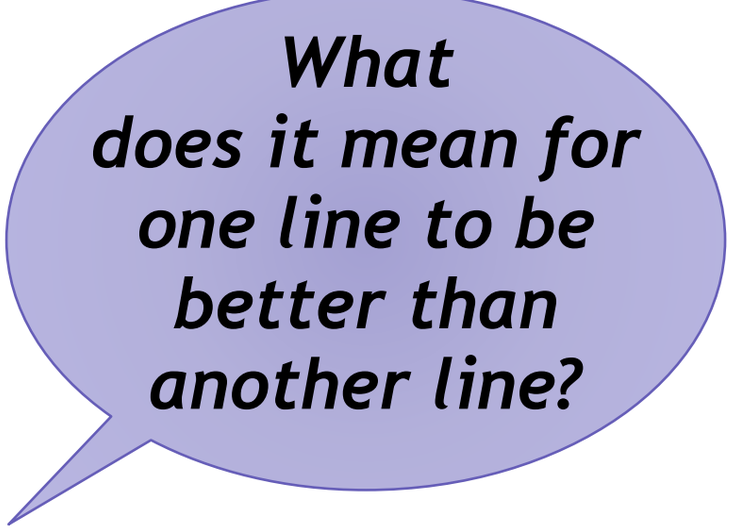
Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
- …

Linear Model



Pair share



HOW MUCH DOES A POINT NOT ON THE LINE **COST**?

WE COULD CREATE A **COST FUNCTION**. THE BEST LINE HAS THE LOWEST COST WHEN SUMMED OVER ALL THE EXAMPLES

WHAT **COST FUNCTION** SHOULD WE USE?

How to measure errors?

mpg $\approx w_0 + w_1$ horsepower

$$\hat{y} = w_0 + w_1 \mathbf{x}$$

If $w_0 = 39.94, w_1 = -0.16$

$$\hat{y} = h(\mathbf{x}) = 39.94 + (-0.16)\mathbf{x}$$

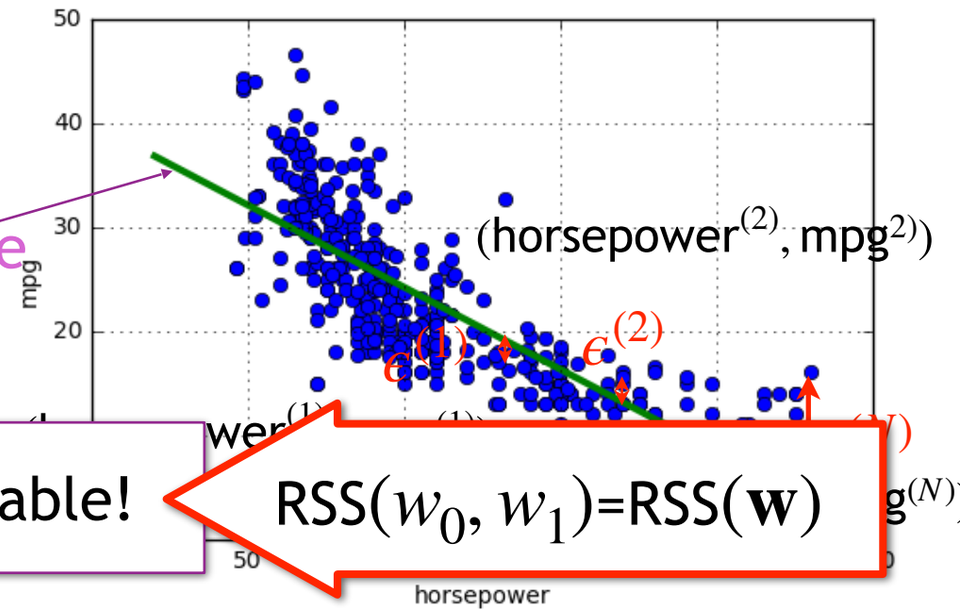
horsepower	mpg
130.0	18.0
165.0	15.0
150.0	18.0
150.0	16.0
…	…

- Absolute value of difference?

$$|y - \hat{y}| = |\text{MPG} - \text{predicted MPG}|$$

$$|18 - (w_0 + w_1 * 130)|$$
$$+|15 - (w_0 + w_1 * 165)|$$
$$+|18 - (w_0 + w_1 * 150)|$$
$$+|16 - (w_0 + w_1 * 150)|$$
$$\dots$$

Note that
$|y - \hat{y}| = |\hat{y} - y|$
E.g. $|w_0 + w_1 * 130 - 18|$
$= |18 - (w_0 + w_1 * 130)|$



- Squared value of difference?

$$(y - \hat{y})^2 = (\text{MPG} - \text{predicted MPG})^2$$

$$(18 - (w_0 + w_1 * 130))^2$$
$$+(15 - (w_0 + w_1 * 165))^2$$
$$+(18 - (w_0 + w_1 * 150))^2$$
$$+(16 - (w_0 + w_1 * 150))^2$$
$$\dots$$

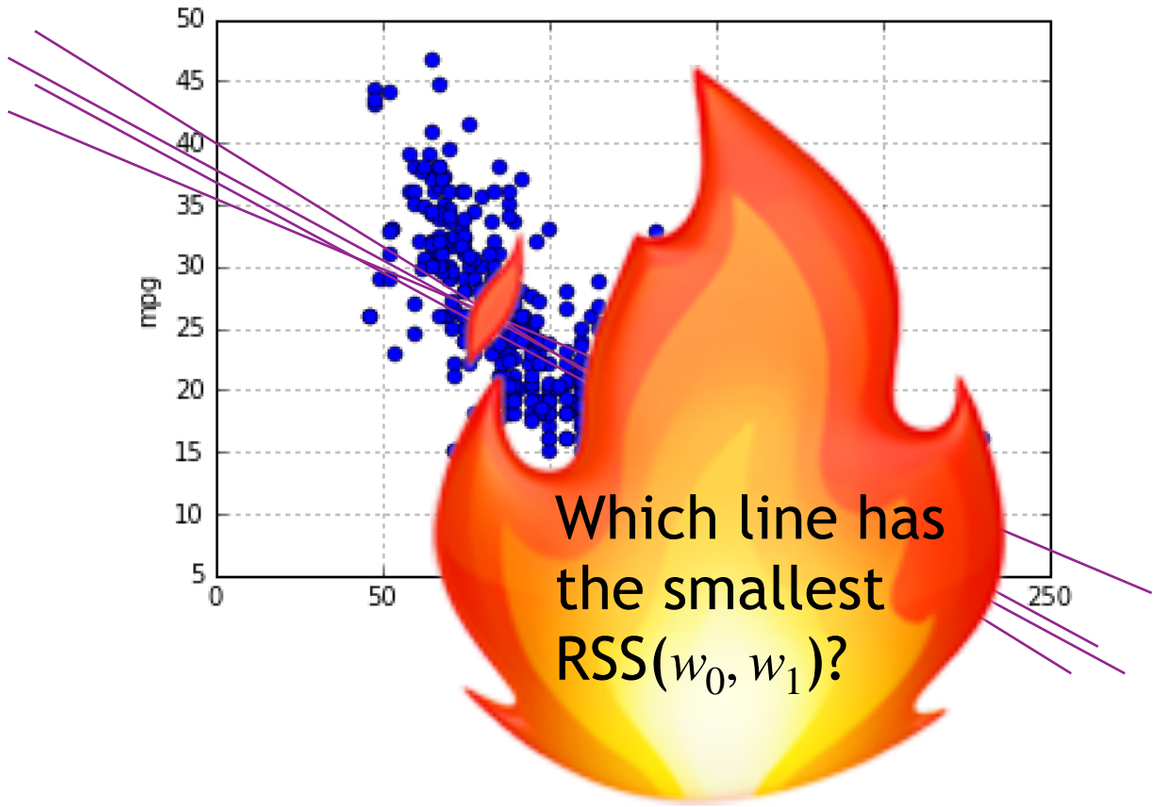
Note that $(y - \hat{y})^2 = (\hat{y} - y)^2$
E.g. $(w_0 + w_1 * 130 - 18)^2$
$= \left(18 - (w_0 + w_1 * 130 - 18)\right)^2$

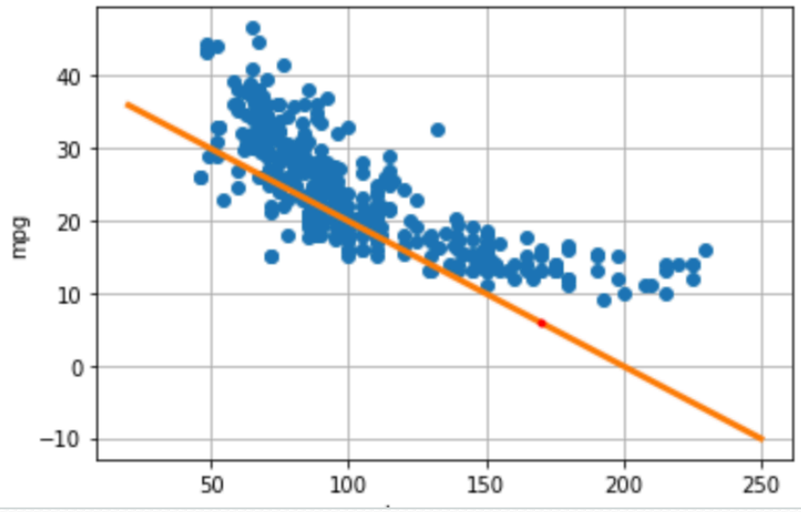
Differentiable!

RSS(w_0, w_1)=RSS$(\mathbf{w})$

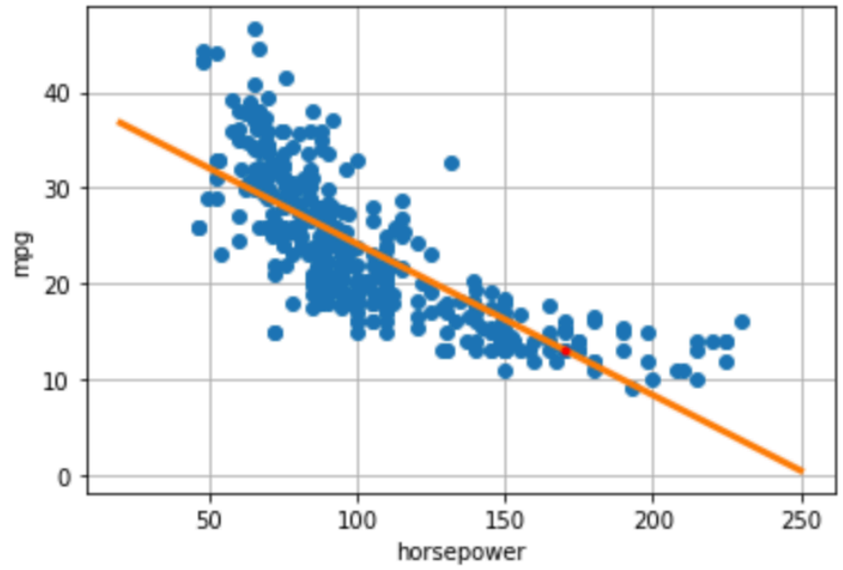
We chose to minimize the variation around the line

Linear Model



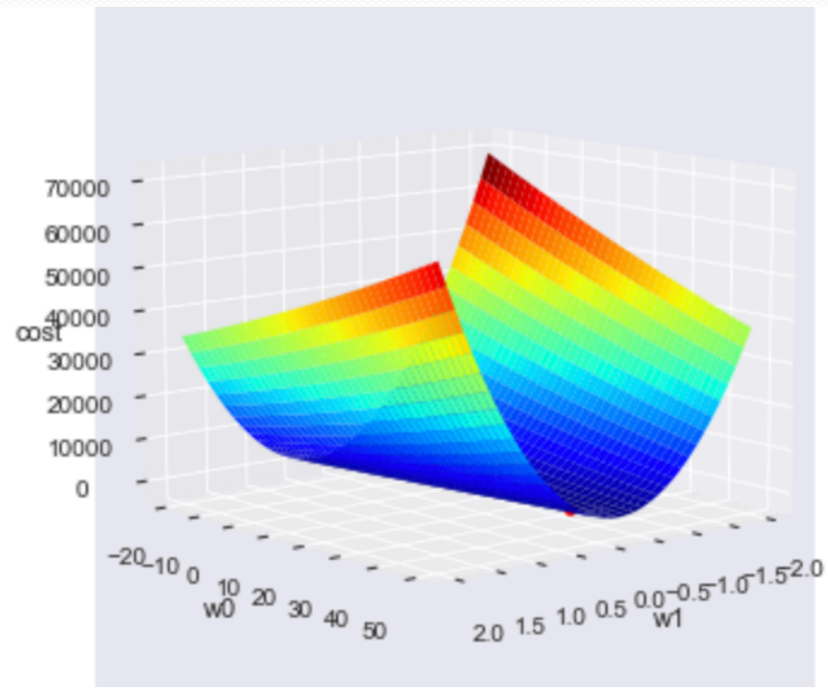


horsepower	mpg
130.0	18.0
165.0	15.0
150.0	18.0
150.0	16.0
…	…

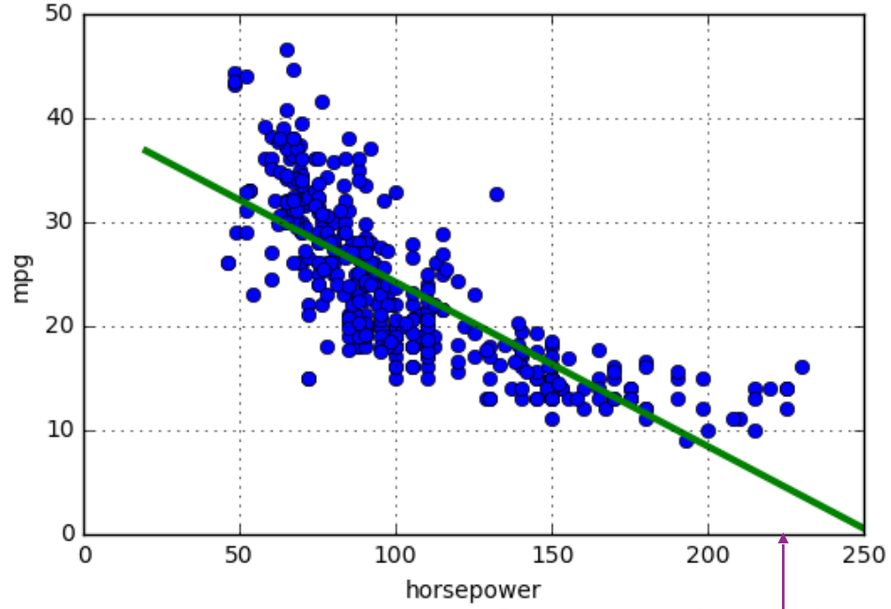


$$RSS(39.9,\ -0.2)$$
$$= (39.9 + (-0.2) * 130 - 18)^2 + (39.9 + (-0.2) * 165 - 15)^2 + \cdots$$
$$= 18142.38$$

$$RSS(39.94,\ -0.16)$$
$$= (39.94 + (-0.16) * 130 - 18)^2 + (39.94 + (-0.16) * 165 - 15)^2 + \cdots$$
$$= 9385.92$$



Thought experiment



mpg = w_0 + w_1 horsepower

- Would we choose a different set of parameters w_0, w_1 if our **Objective function was:**

$$J(w_0, w_1) = \frac{1}{2}\,\mathbf{MSE}(w_0, w_1) = \frac{1}{2N}\mathbf{RSS}(w_0, w_1)$$

MSE over the *training data* is called the "in sample" error:
$E_{\text{in}}(\mathbf{w})$

Updated loss function

General ML problem	Linear regression
☑ Get data	→ Data: $(\mathbf{x}^{(i)}, y^{(i)}), i = 1,2,\ldots,N$
☑ Pick a model with parameters	→ Linear model: $\hat{y}^{(j)} = w_0 + w_1 x_1^{(j)} + w_2 x_2^{(j)} + \cdots + w_d x_d^{(j)}$
☑ Pick a loss function ◦ Measures goodness of fit model to data ◦ Function of the parameters	→ Loss function: $\frac{\text{RSS}(\mathbf{w})}{2N} = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$ $= \frac{1}{2N}\left[(y^{(1)} - \hat{y}^{(1)})^2 + (y^{(2)} - \hat{y}^{(2)})^2 + \cdots + (y^{(N)} - \hat{y}^{(N)})^2\right]$
☐ Find parameters that minimizes loss	→ Select $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_d]^T$ to minimize $\frac{\text{RSS}(\mathbf{w})}{2N}$ This $\mathbf{w}$ also minimizes $\text{RSS}(\mathbf{w})$

Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
- …

How do we find the *global minimum*?

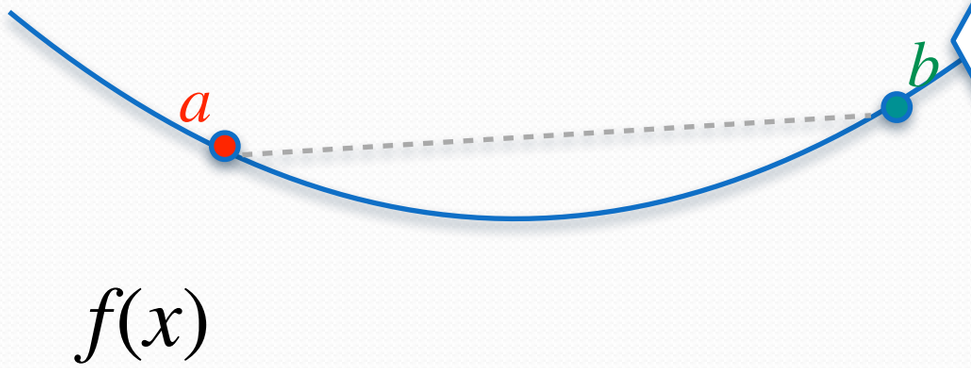
Remember $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$

I.e. how do we find the line that ***minimizes*** the ***cost function?***

$$\min_{\mathbf{w}} \text{RSS}(\mathbf{w})$$

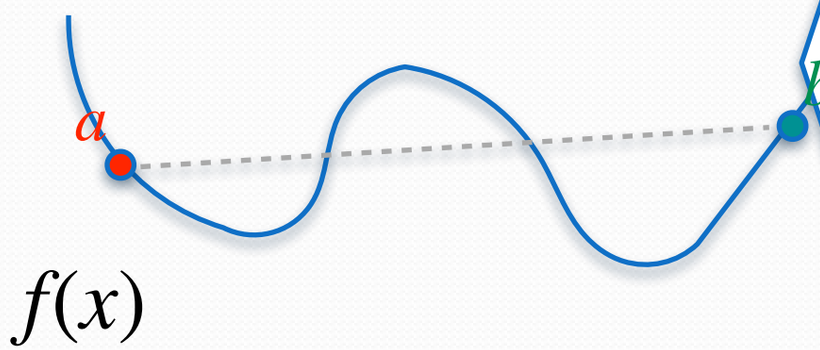
Which values for w_0, w_1 minimizes $\text{RSS}(\mathbf{w})$?

Convex function

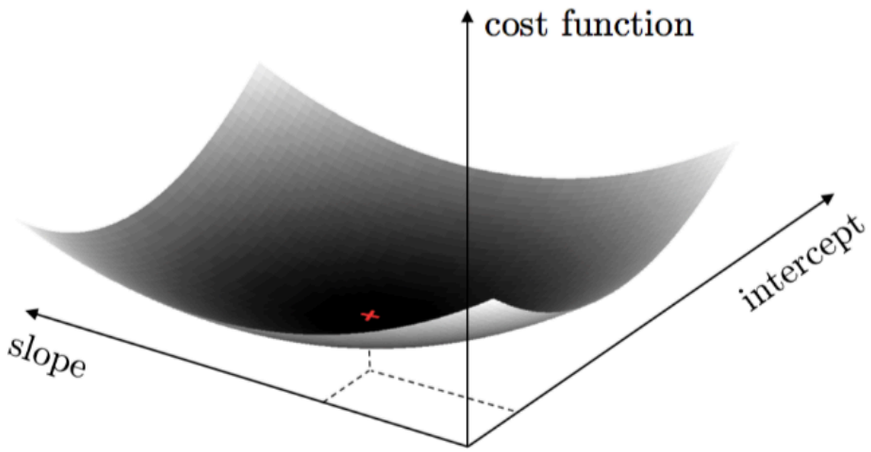


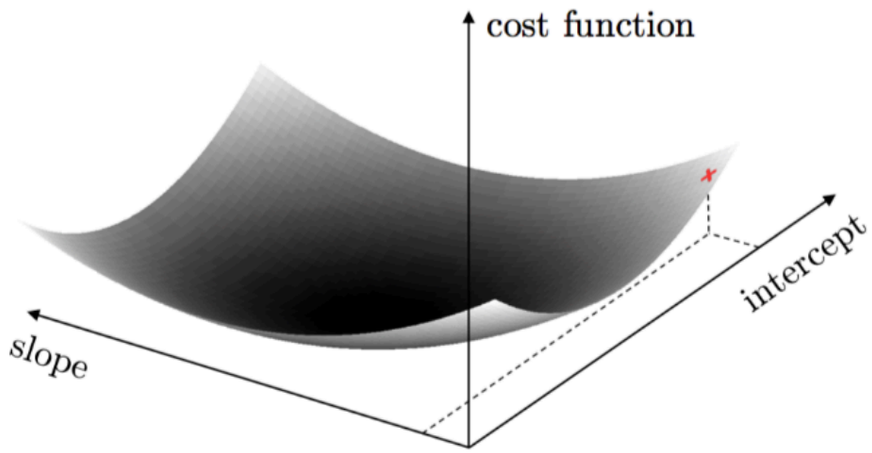
Only one global minimum value

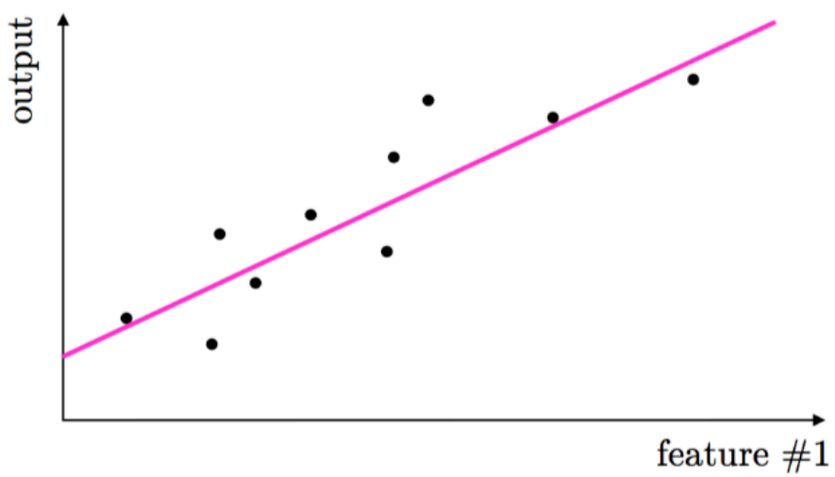
Not a convex function

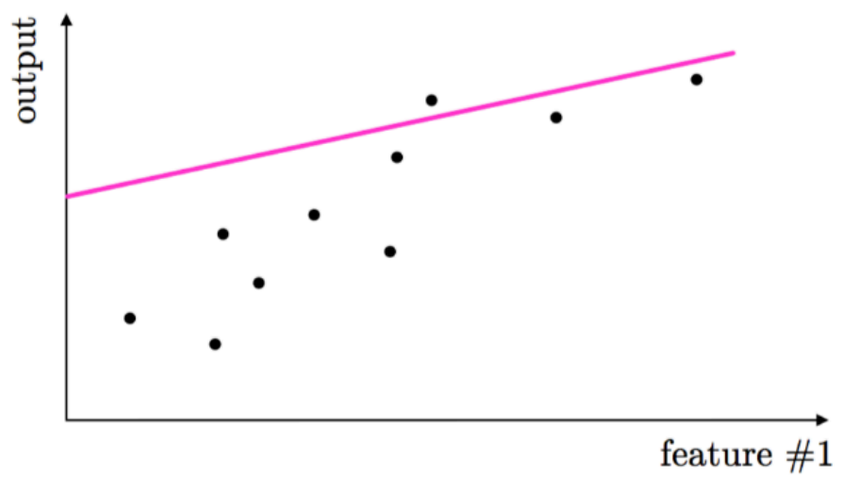


More than one local minimum value









Slide from Machine Learning Refined: Foundations, Algorithms, and Applications

...this would be even more difficult if we had more dimensions

I think we need an **algorithmic approach** to find the **optimal value**

Mathematical optimization

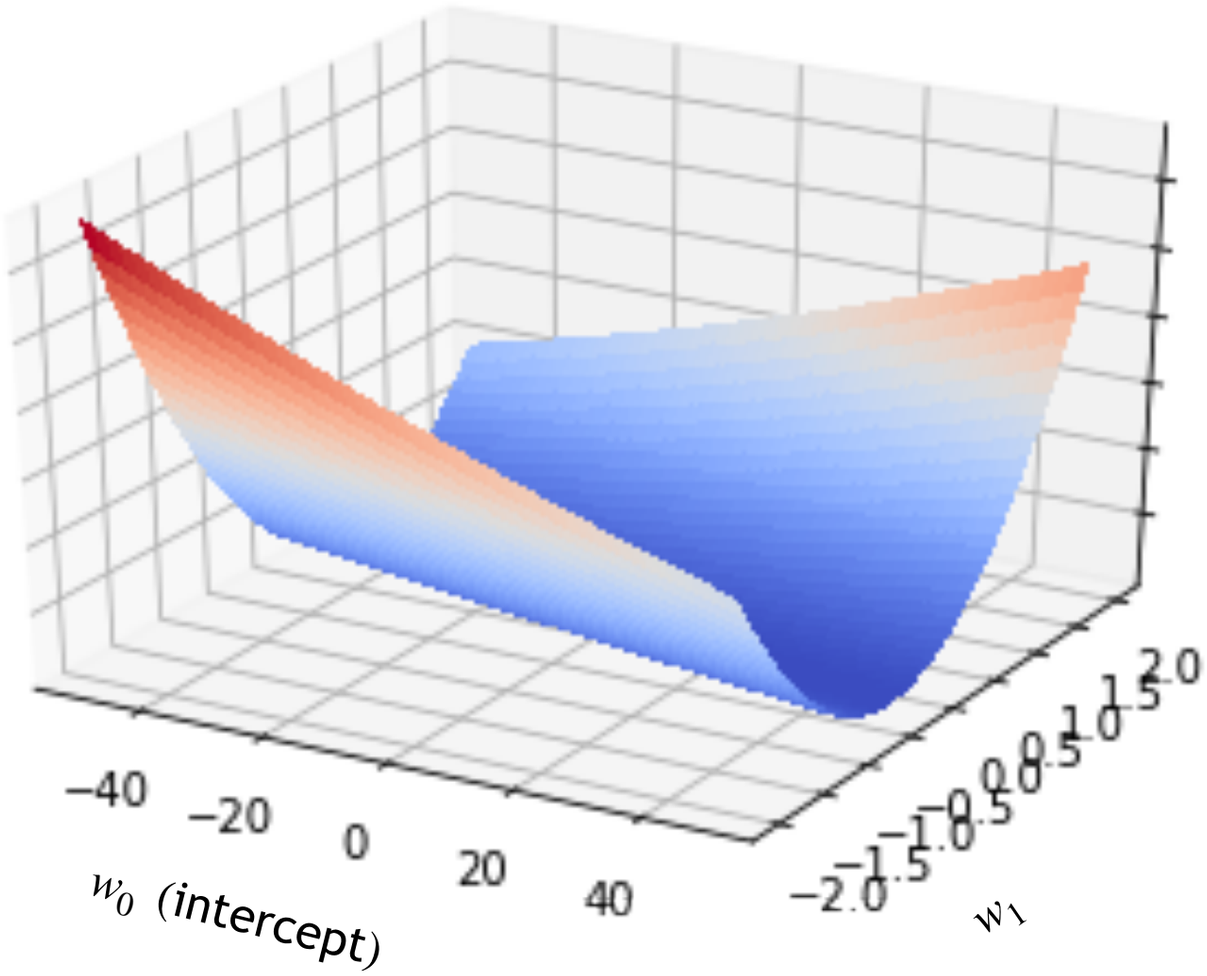
Pair share

Discuss various methods for finding the optimal parameters, i.e. values for $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$

Global optimization methods

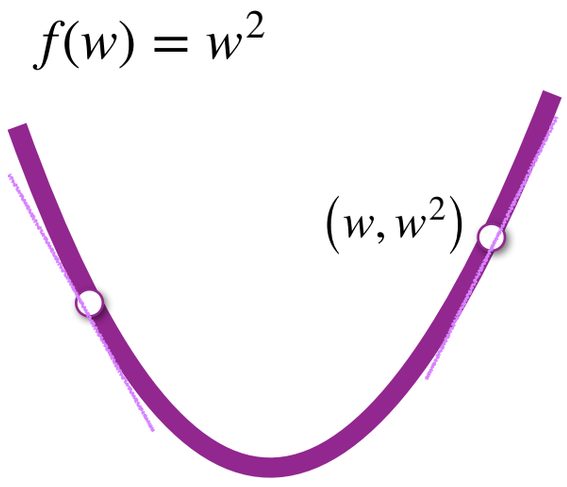
Discussion of ideas

Local optimization methods



Before finding how to use a local optimization technique on RSS, lets develop some intuition on a simpler function

Local optimization



Lets use a simpler function: $f(w) = w^2$
For any w, which $w' = w + \epsilon$ would most likely cause $f(w') < f(w)$?
(You may assume $f(w) \neq 0$)

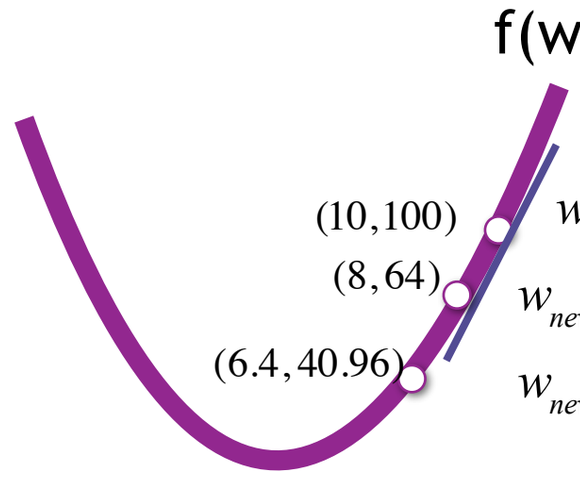
Notation alert!
w' is just another variable, not the derivative.

Local optimization

- Notice that the amount we move decreases as we move toward the minimum
  - If we are at w = 10 the amount we move is 2
  - If we are at 8 the amount we move is 1.6
- If we started at x = -10, the derivative is negative, so

$$w_{new} = (-10) - \alpha \cdot 2 \cdot (-10)$$
$$= (-10) - (0.1) \cdot 2 \cdot (-10)$$
$$= (-10) + 2 = -8$$

The derivate gives the direction to move



Algorithm:
For i = 1 to num_iters:
  if $\frac{df(w)}{dw} > 0$ then f is increasing,
  move w a little to the left

  if $\frac{df(w)}{dw} < 0$ then f is decreasing,
  move w a little to the right

A nice intuitive explanation is at: https://theory.stanford.edu/~tim/s16/l/l5.pdf

Slide NOT part of lecture. I am only including this slide for those students who are interested.

Justification for gradient descent using the Taylor Expansion

Taylor expansion for a univariate real-valued function:

$$f(w+\Delta w) = f(w) + f'(w)\Delta w + f''(w)\frac{(\Delta w)^2}{2!} + f'''(w)\frac{(\Delta w)^3}{3!} + \cdots$$

We can approximate this function for ***small*** Δw by the first-order Taylor approximation

$$f(w+\Delta w) \approx f(w) + f'(w)\Delta w$$

First-order Taylor expansion for a Multivariate real-valued function:

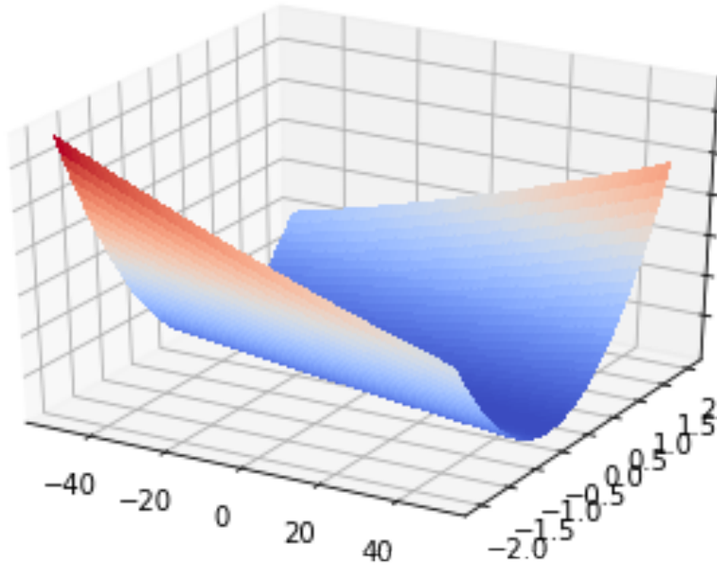
$$f(\mathbf{w}+\Delta\mathbf{w}) \approx f(\mathbf{w}) + \Delta\mathbf{w}^T \nabla f(\mathbf{w})$$

Now if we choose $\Delta\mathbf{w} = -\alpha\nabla f(\mathbf{w})$ then $f(\mathbf{w}+\Delta\mathbf{w}) \approx f(\mathbf{w}) - \alpha\nabla f(\mathbf{w})^T \nabla f(\mathbf{w}) < f(\mathbf{w})$

Positive if the gradient is not zero

Now we just need to make sure that we don't move to far so the approximation is reasonable!

Local Optimization



WHAT IS THE PSEUDOCODE?

Open discussion - design an algorithm using the ideas we just discussed.

$$\mathbf{x}^{(i)} = \left[x_1^{(i)}\right]$$

Objective function:

$$RSS(w_0, w_1) = \sum_{i=1}^{N} ((w_0 + w_1 x_1^{(i)}) - y^{(i)})^2$$

Simplification: New Objective function

$$J(w_0, w_1) = \frac{1}{2N} RSS(w_0, w_1) = \frac{1}{2} MSE(w_0, w_1) = \frac{1}{2N} \sum_{i=1}^{N} ((w_0 + w_1 x_1^{(i)}) - y^{(i)})^2$$

Partial derivatives

$$\frac{\partial J(w_0, w_1)}{\partial w_0} =$$

$$\frac{\partial J(w_0, w_1)}{\partial w_1} =$$

Cost function

- Minimizing RSS(**w**) $= \sum_{i=1}^{N} (y^{(i)} - w_0 - w_1 x^{(i)})^2$ (our cost function for linear regression) is the same as minimizing:

$$J(w_0, w_1) = \frac{1}{2} \mathrm{E}_{\text{in}}(w_0, w_1) = \frac{1}{2N} \sum_{i=1}^{N} (w_0 + w_1 x^{(i)} - y^{(i)})^2 \quad = \frac{1}{2N} RSS$$

notice we switch the sign inside the bracket

- J, RSS and E_{in} are convex functions (as was x^2), so we can use the gradient to find the minimum by taking a sequence of steps. Here is the derivative for the E_{in} function with respect to the parameters:

$$\frac{\partial J(w_0, w_1)}{\partial w_0} = \frac{1}{N} \sum_{i=1}^{N} (w_0 + w_1 x^{(i)} - y^{(i)})$$

$$\frac{\partial \mathrm{J}(w_0, w_1)}{\partial w_1} = \frac{1}{N} \sum_{i=1}^{N} (w_0 + w_1 x^{(i)} - y^{(i)}) x^{(i)}$$

By scanned by jd, photography by "frères Bisson (… nationale de France (BNF)) [Public domain], via Wikim… Commons

Gradient Descent Optimization

$$\frac{\partial J(w_0,w_1)}{\partial w_0} = \frac{1}{N}\sum_{i=1}^{N}(w_0 + w_1 x^{(i)} - y^{(i)}) \qquad \frac{\partial J(w_0,w_1)}{\partial w_1} = \frac{1}{N}\sum_{i=1}^{N}(w_0 + w_1 x^{(i)} - y^{(i)})x^{(i)}$$

- To decrease the cost, we update the parameters, w_0, w_1, using the update rule:

for $i = 1$ to num_iter

$$temp0 = w_0 - \alpha\frac{\partial J(w_0,w_1)}{\partial w_0} = w_0 - \frac{\alpha}{N}\sum_{i=1}^{N}(w_0 + w_1 x^{(i)} - y^{(i)})$$

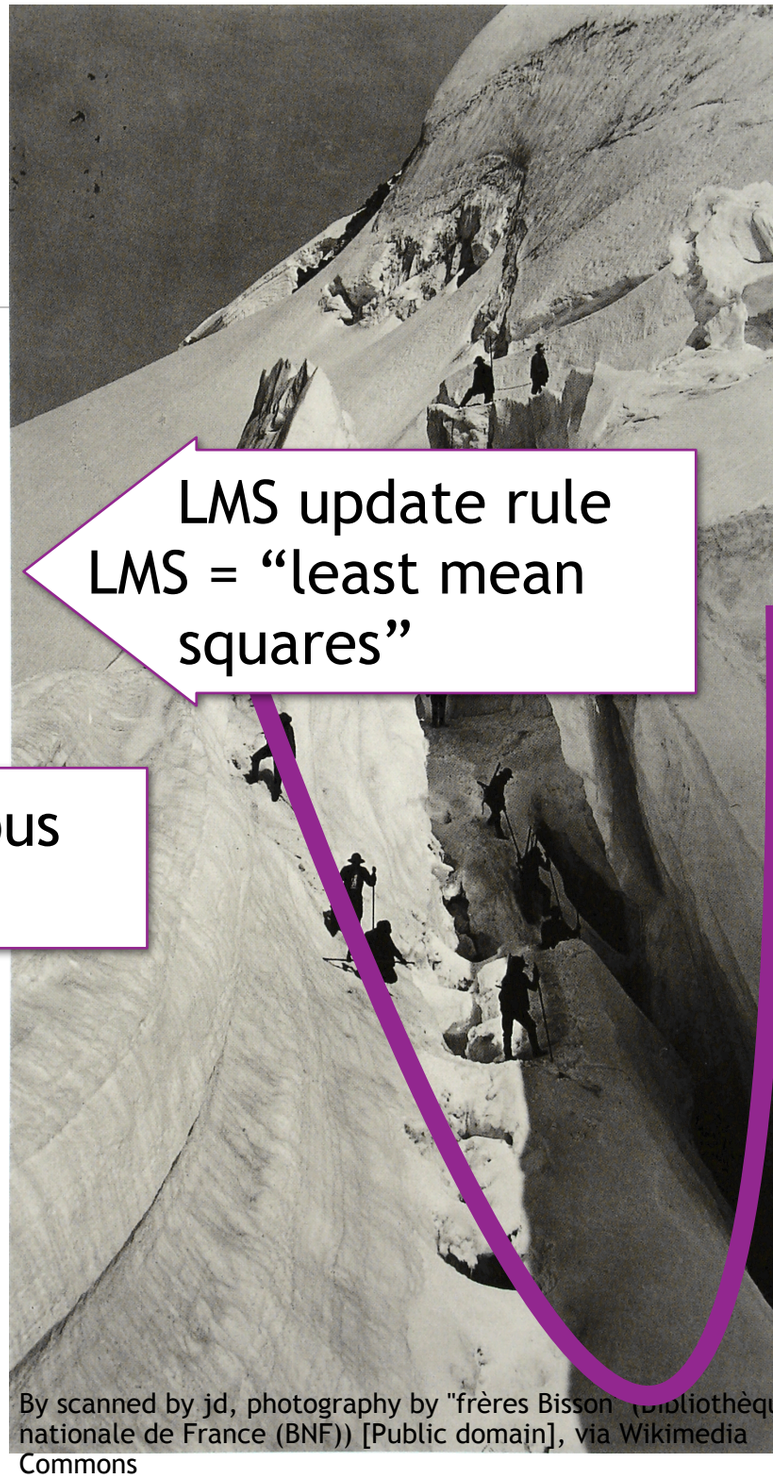
$$temp1 = w_1 - \alpha\frac{\partial J(w_0,w_1)}{\partial w_1} = w_1 - \frac{\alpha}{N}\sum_{i=1}^{N}(w_0 + w_1 x^{(i)} - y^{(i)})x^{(i)}$$

$$w_0 = temp0$$

$$w_1 = temp1$$

α learning rate

Simultaneous update

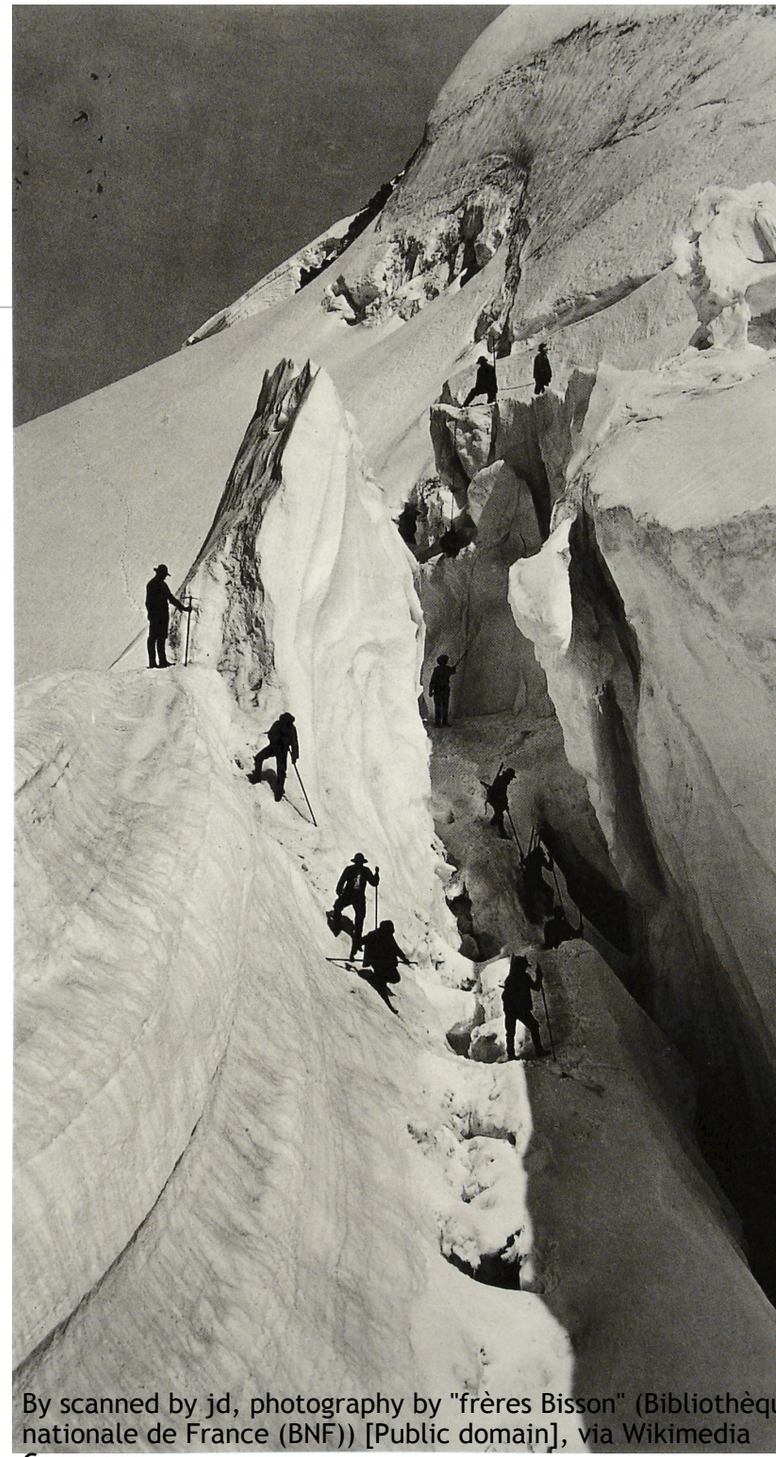


LMS update rule
LMS = "least mean squares"

By scanned by jd, photography by "frères Bisson" (Bibliothèque nationale de France (BNF)) [Public domain], via Wikimedia Commons

Batch Gradient Descent

- We need some initial values for w_0, w_1. We can choose any initial values. We will choose $w_0 = 0, w_1 = 0$
- How do we choose our step size, α?
- The gradient descent applies to more general class of functions than RSS



By scanned by jd, photography by "frères Bisson" (Bibliothèque nationale de France (BNF)) [Public domain], via Wikimedia Commons