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 continued

Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
  - When $d = 1$ (simple linear regression)
  - What can go wrong?
  - General case, any $d > 0$ (multiple variable linear regression)
    - Algorithm
    - Example
    - Vectorized Algorithm
- Global Optimization: Normal Equations
- Feature scaling
- Assessing Goodness of Fit
- Extensions
- Removing features
- Objective function revisited: Probabilistic interpretation

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
- Understand feature scaling/data normalization
- Understand how to find the objective function using MLE

Updated loss function

General ML problem

- Get data
- Pick a model with parameters
- Pick a loss function
  - Measures goodness of fit model to data
  - Function of the parameters
- Find parameters that minimizes loss

Linear regression

Data: $(\mathbf{x}^{(i)}, y^{(i)}), i = 1,2,\ldots,N$

Linear model:

$$\hat{y}^{(j)} = w_0 + w_1 x_1^{(j)} + w_2 x_2^{(j)} + \cdots + w_d x_d^{(j)}$$

Loss function: $\frac{\text{RSS}(\mathbf{w})}{2N} = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$

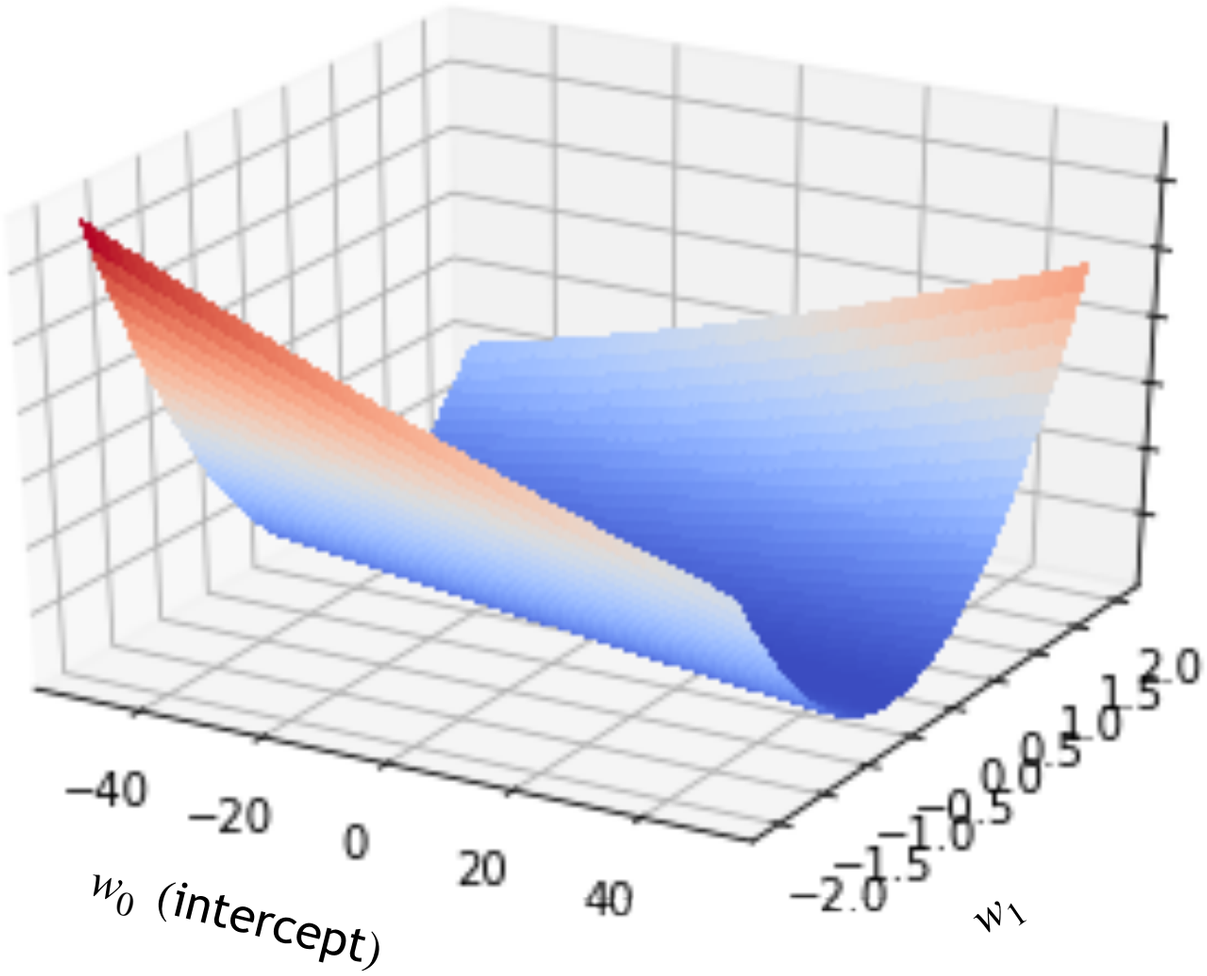
$$= \frac{1}{2N}\left[(y^{(1)} - \hat{y}^{(1)})^2 + (y^{(2)} - \hat{y}^{(2)})^2 + \cdots + (y^{(N)} - \hat{y}^{(N)})^2\right]$$

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

Global optimization methods

Discussion of ideas

Local optimization methods



Gradient Descent Optimization

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

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

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

α learning rate

for $i = 1$ to num_iter

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

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

Next w

Curent w

How big

Direction of increase

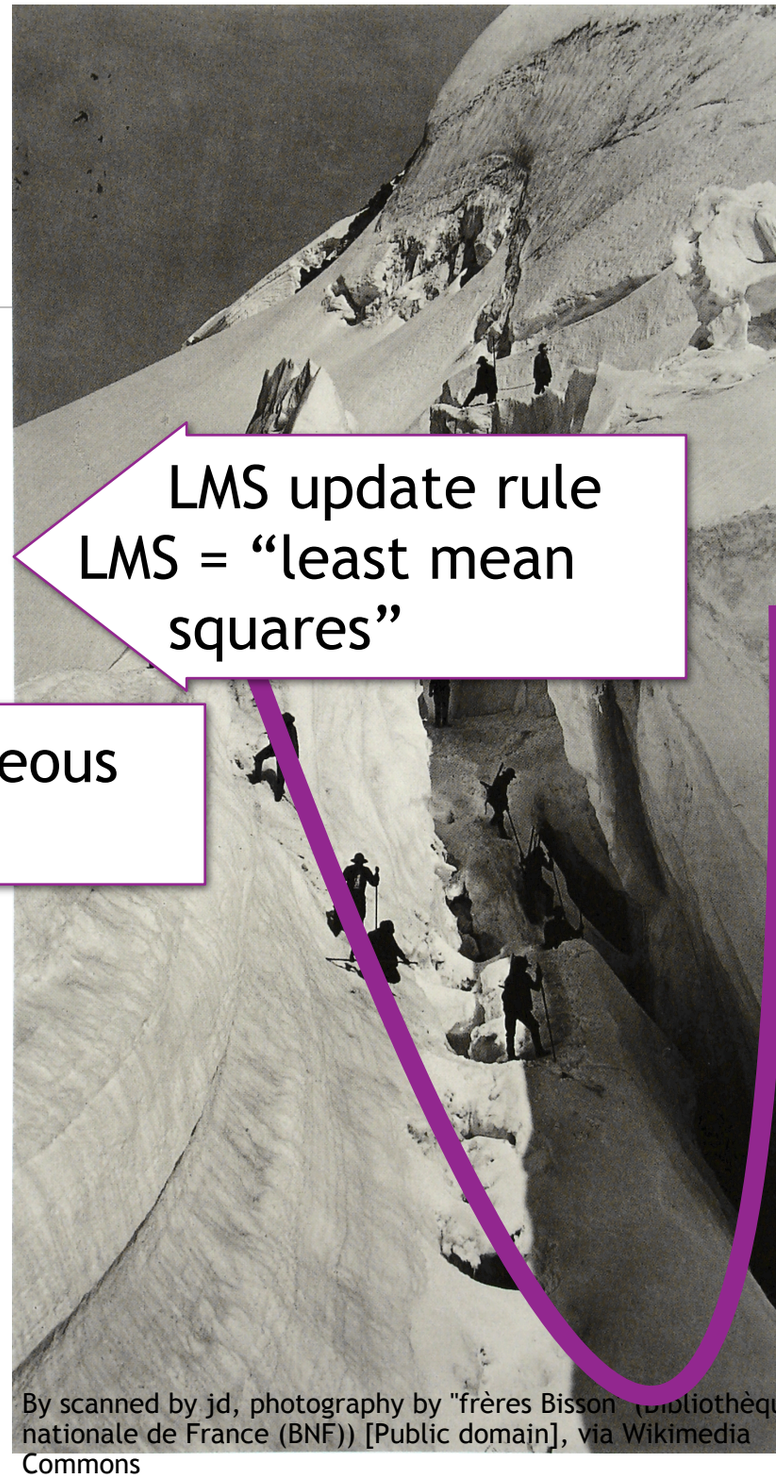
$$w_0 = temp0$$

$$w_1 = temp1$$

LMS update rule

LMS = "least mean squares"

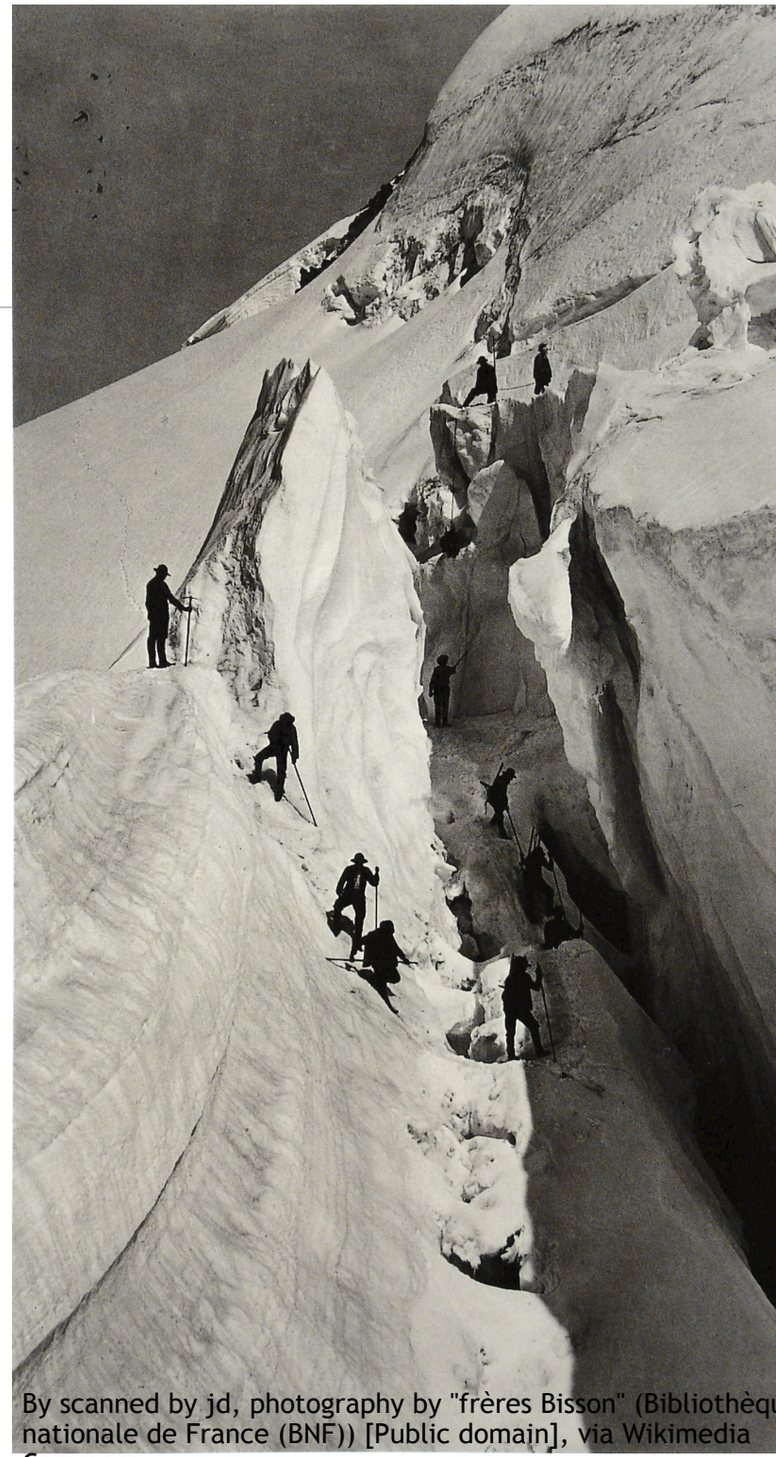
Simultaneous update



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

Batch Gradient Descent

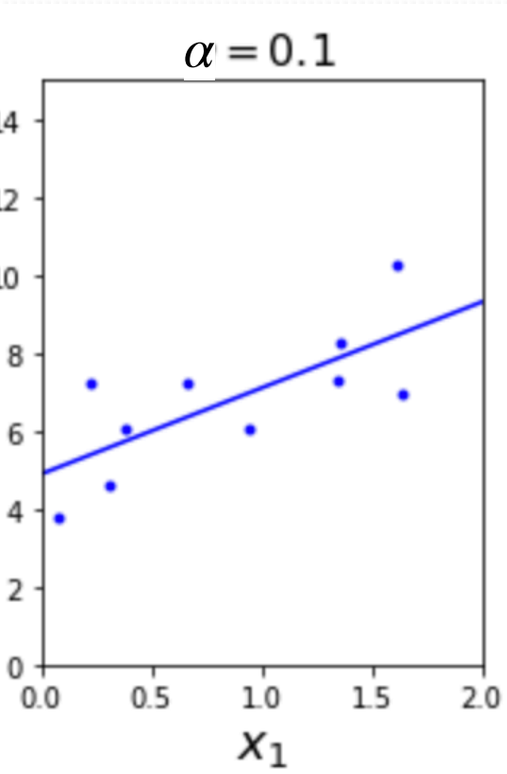
- We need some initial values for w_0, w_1. We can choose any initial values. We will choose $w_0 = 0, w_1 = 0$
- How do we choose our step size, α?
- The gradient descent applies to more general class of functions than RSS



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

Toy example: two steps of gradient descent

*Numbers were rounded



If $\mathbf{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ then $\hat{y}^{(i)} = 0$ for all training e

The partial derivatives of J at the point $w_0 = 0,\ w_1 = 0$

$$\frac{\partial J(0,0)}{\partial w_0} = \frac{1}{10}\left((0-6.06)+(0-3.79)+\cdots+(0-6.96)\right) = -6.78$$

$$\frac{\partial J(0,0)}{\partial w_1} = \frac{1}{10}\left((0-6.06)(0.38)+(0-3.79)(0.08)+\cdots+(0-6.96)(1.64)\right) = -6.53$$

$$\mathbf{w} = \begin{bmatrix} 0.68 \\ 0.65 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.1 \begin{bmatrix} -6.78 \\ -6.53 \end{bmatrix}$$

The current $w_0 = w_1 = 0$, The learning rate is $\alpha = 0.1$

Next $\mathbf{w}$

Curent $\mathbf{w}$

How big

Direction of increase

$$X = \begin{bmatrix} 0.38 \\ 0.08 \\ 0.31 \\ 1.62 \\ 0.66 \\ 0.94 \\ 1.35 \\ 0.22 \\ 1.36 \\ 1.64 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 6.06 \\ 3.79 \\ 4.64 \\ 10.29 \\ 7.21 \\ 6.08 \\ 7.31 \\ 7.21 \\ 8.24 \\ 6.96 \end{bmatrix}$$

For $\mathbf{w} = \begin{bmatrix} 0.68 \\ 0.65 \end{bmatrix}$ then $\hat{\mathbf{y}} = \begin{bmatrix} 1 & 0.38 \\ 1 & 0.08 \\ \vdots & \\ 1 & 1.64 \end{bmatrix} \begin{bmatrix} 0.68 \\ 0.65 \end{bmatrix} = \begin{bmatrix} 0.93 \\ 0.73 \\ \vdots \\ 1.75 \end{bmatrix}$

The partial derivatives of J at the point $w_0 = 0.68,\ w_1 = 0.65$

$$\frac{\partial J(0.68,0.65)}{\partial w_0} = \frac{1}{10}\left((0.93-6.06)+(0.73-3.79)+\cdots+(1.75-6.96)\right) = -5.54$$

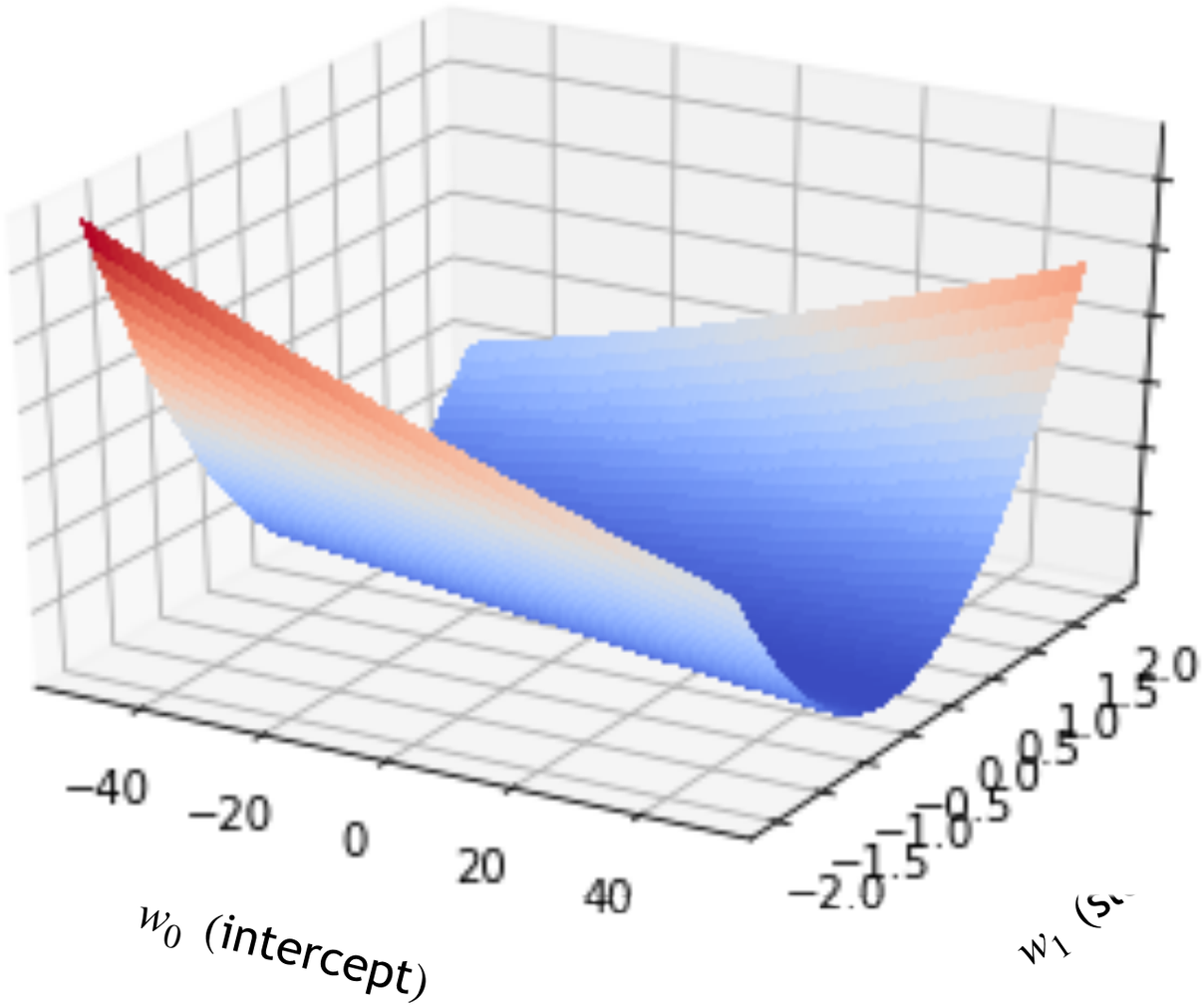
$$\frac{\partial J(0.68,0.65)}{\partial w_1} = \frac{1}{10}\left((0.93-6.06)(0.38)+(0.73-3.79)(0.08)+\cdots+(1.75-6.96)(1.64)\right) = -5.25$$

$$\mathbf{w} = \begin{bmatrix} 1.23 \\ 1.17 \end{bmatrix} = \begin{bmatrix} 0.68 \\ 0.65 \end{bmatrix} - 0.1 \begin{bmatrix} -5.54 \\ -5.25 \end{bmatrix}$$

Outline

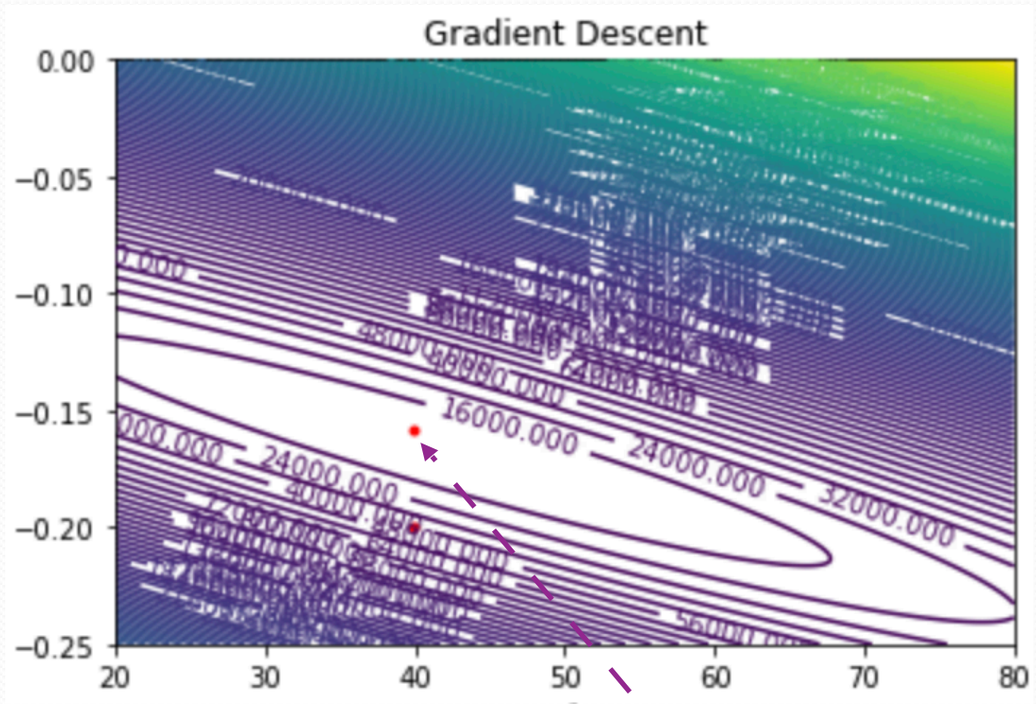
- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
  - When $d = 1$ (simple linear regression)
  - What can go wrong?
  - General case, any $d > 0$ (multiple variable linear regression)
    - Example
    - Algorithm
    - Vectorized Algorithm
- Global Optimization: Normal Equations
- Feature scaling
- Assessing Goodness of Fit
- Extensions
- Removing features
- Objective function revisited: Probabilistic interpretation

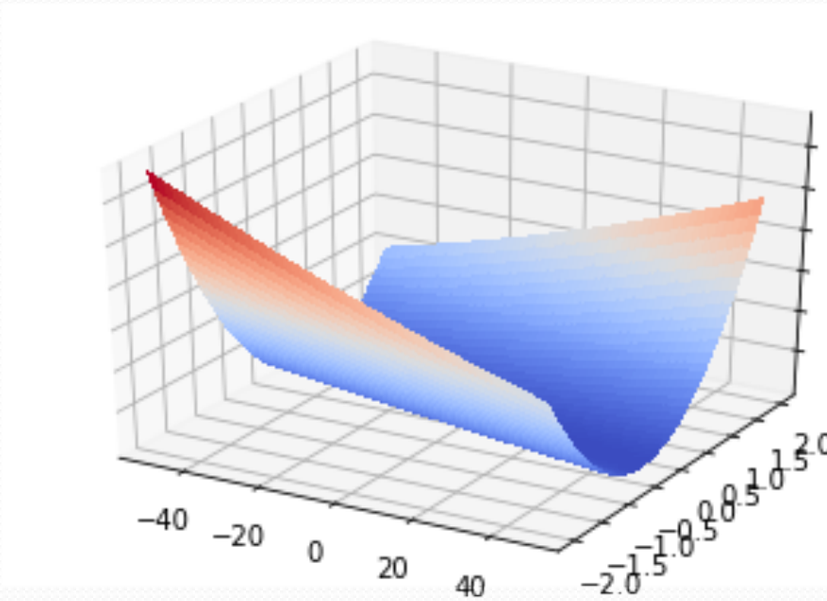
Pair share: what could go wrong with Local optimization methods?



Contour Plots

Graphical technique for representing a 3-D surface in 2-D plane



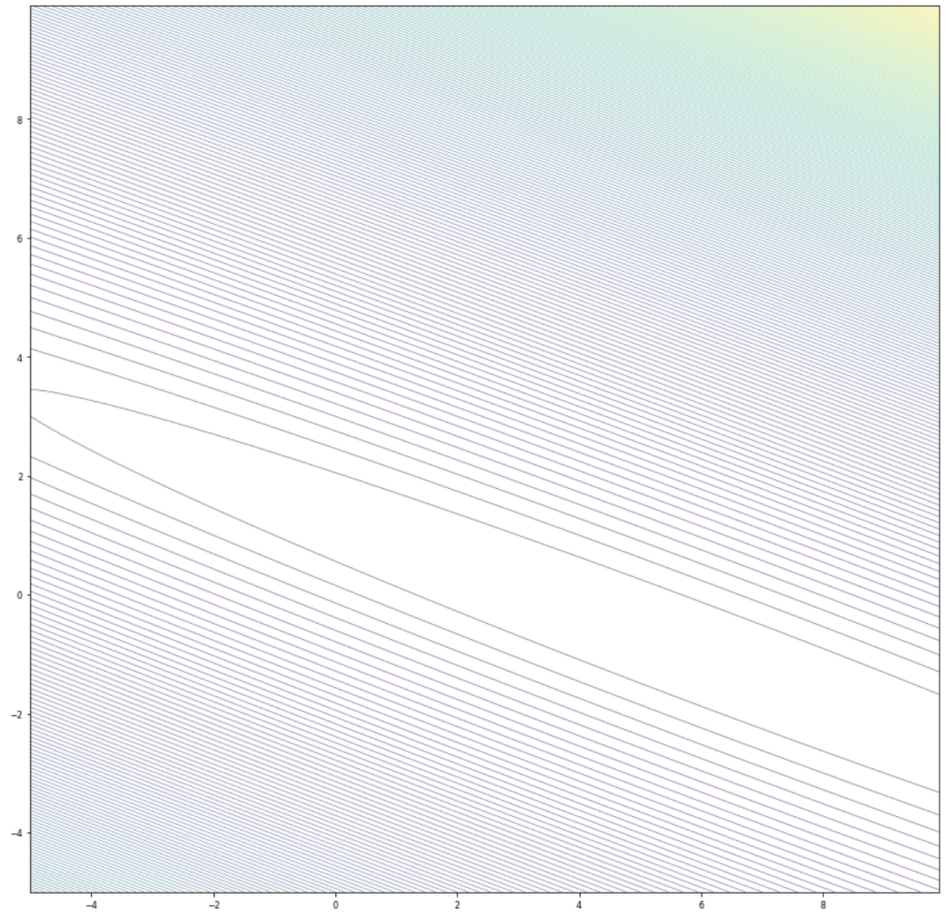


- **Contour curve**: a 3-D contour curve can be formed on the 3-D surface by finding all the points which have the same Z value (e.g. $\text{RSS}(w_0, w_1)$=k, where k is a constant value)
- **Level curve**: are in the the 2-D plane and can be created by projecting the contour curve onto 2-D plane (e.g.the w_0, w_1 axis)
- **Contour plot**: a 2-D plot containing level curves of a function

Read more at: https://ocw.mit.edu/courses/mathematics/18-02sc-multivariable-calculus-fall-2010/2.-partial-derivatives/part-a-functions-of-two-variables-tangent-approximation-and-optimization/session-25-level-curves-and-contour-plots/MIT18_02SC_notes_15.pdf

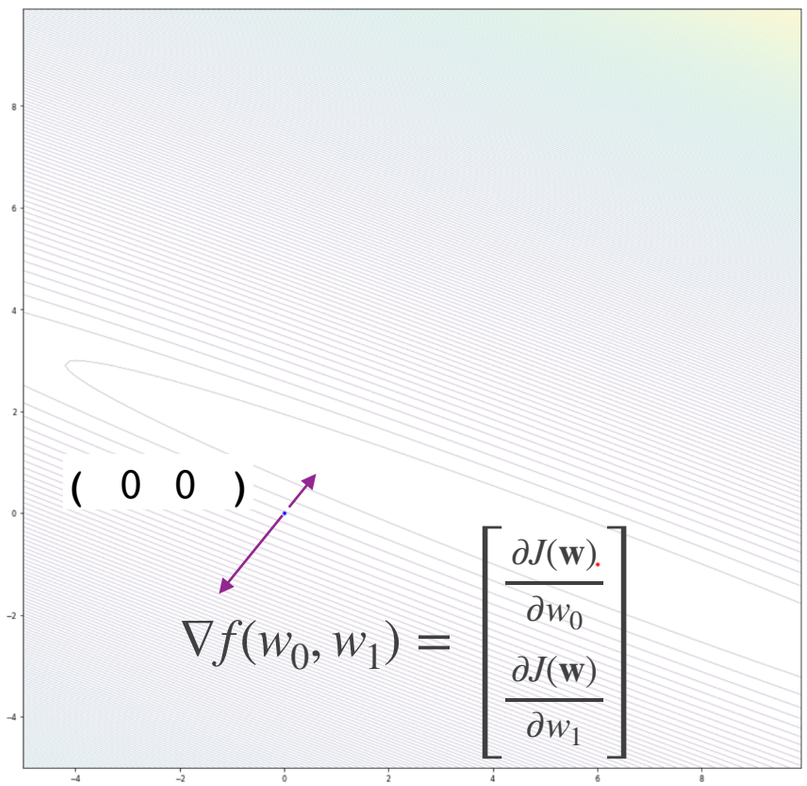
Lets look at minimizing a simpler function

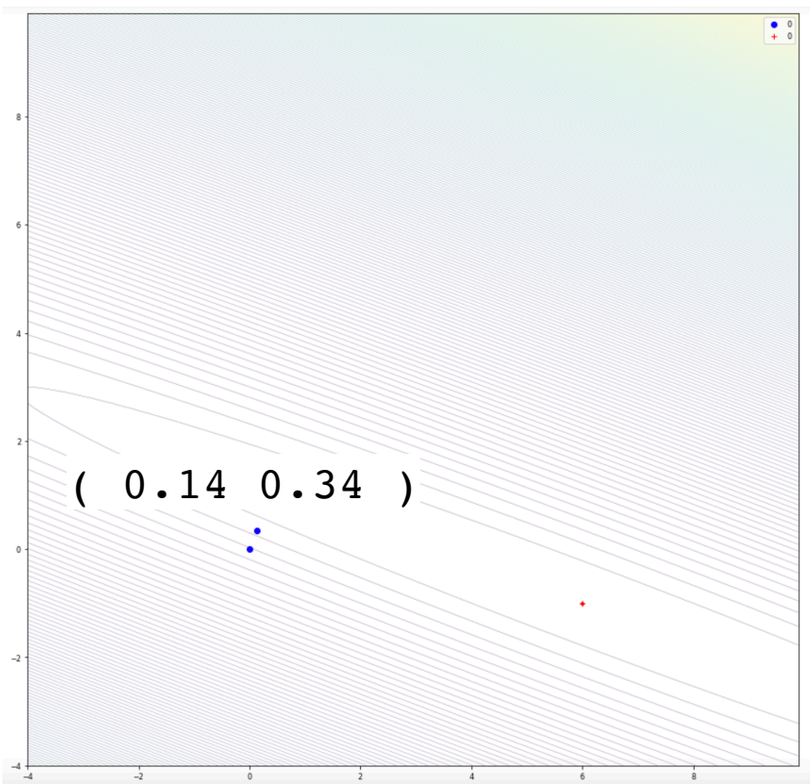
$$f(w_0, w_1) = (w_0 + 2 * w_1 - 4)^2 + (w_0 + 3 * w_1 - 3)^2$$

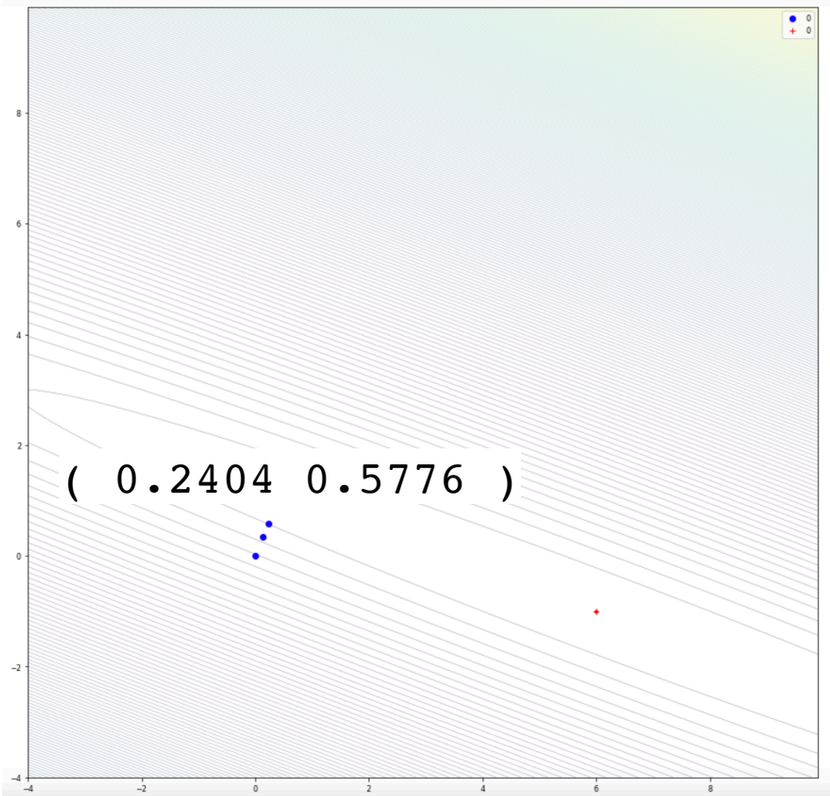


Local optimization to minimize a function

$$f(w_0, w_1) = (w_0 + 2 * w_1 - 4)^2 + (w_0 + 3 * w_1 - 3)^2$$

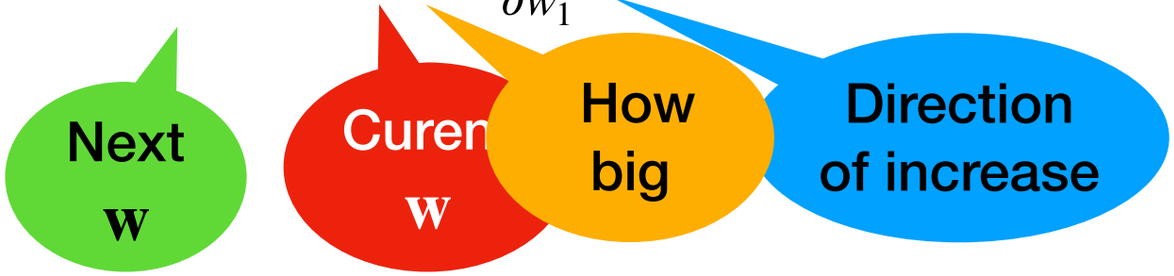




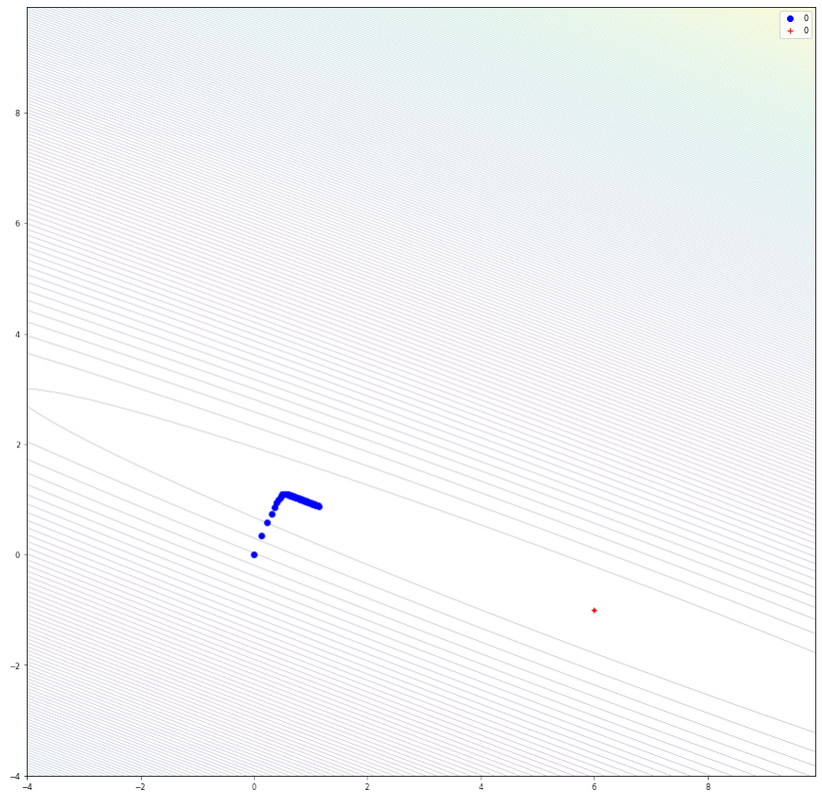


$$\text{temp0} = w_0 - \alpha \frac{\partial J(\mathbf{w})}{\partial w_0}$$

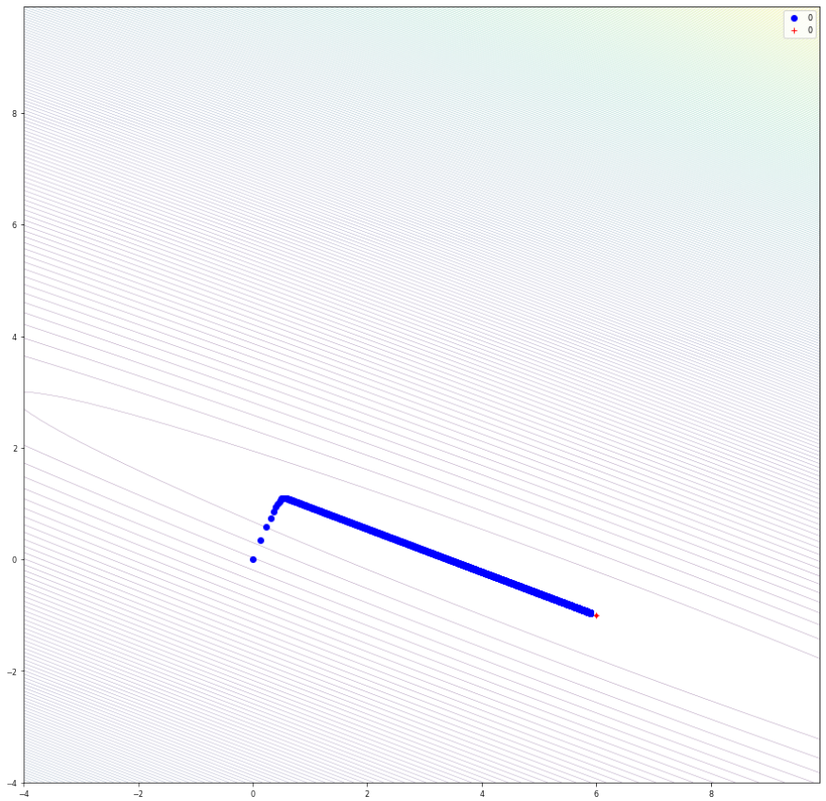
$$\text{temp1} = w_1 - \alpha \frac{\partial J(\mathbf{w})}{\partial w_1}$$



$$f(w_0, w_1) = (w_0 + 2 * w_1 - 4)^2 + (w_0 + 3 * w_1 - 3)^2$$



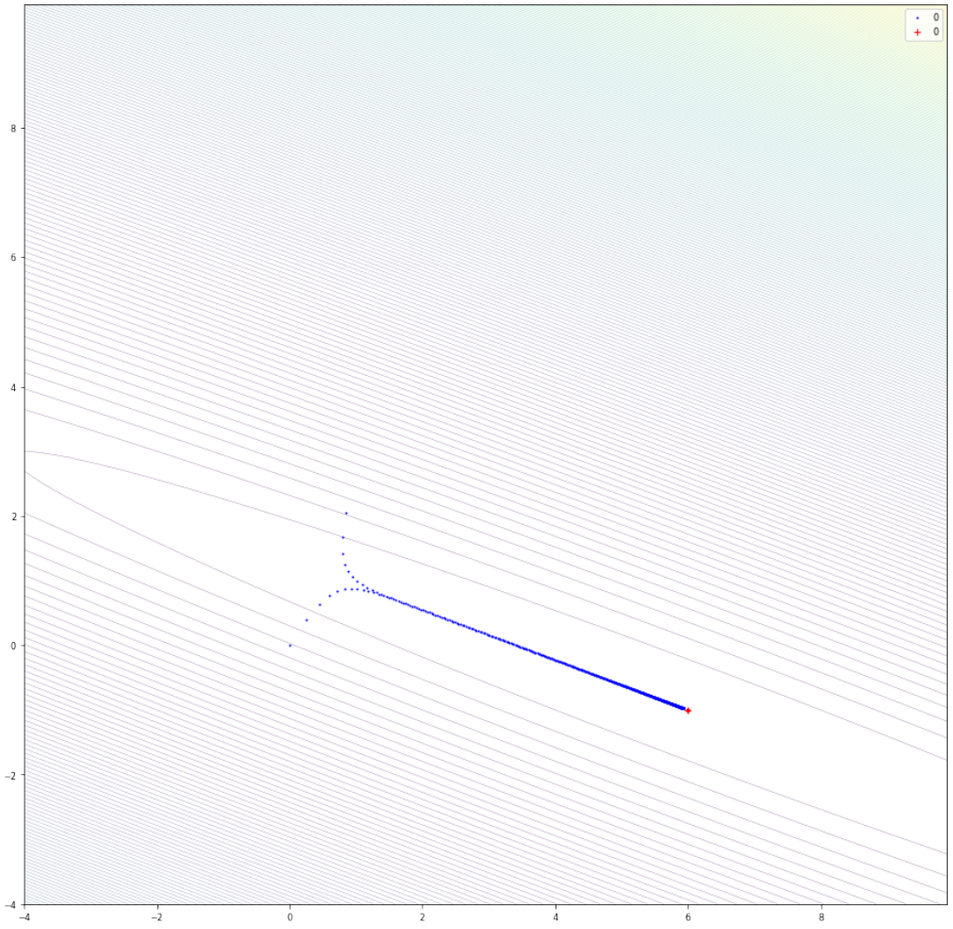
After 100 iterations



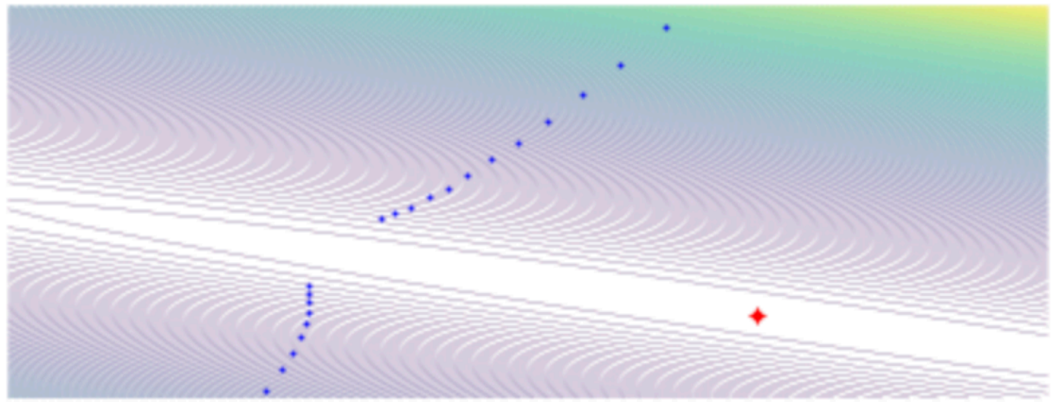
After 3000 iterations

What if we used a larger learning rate?

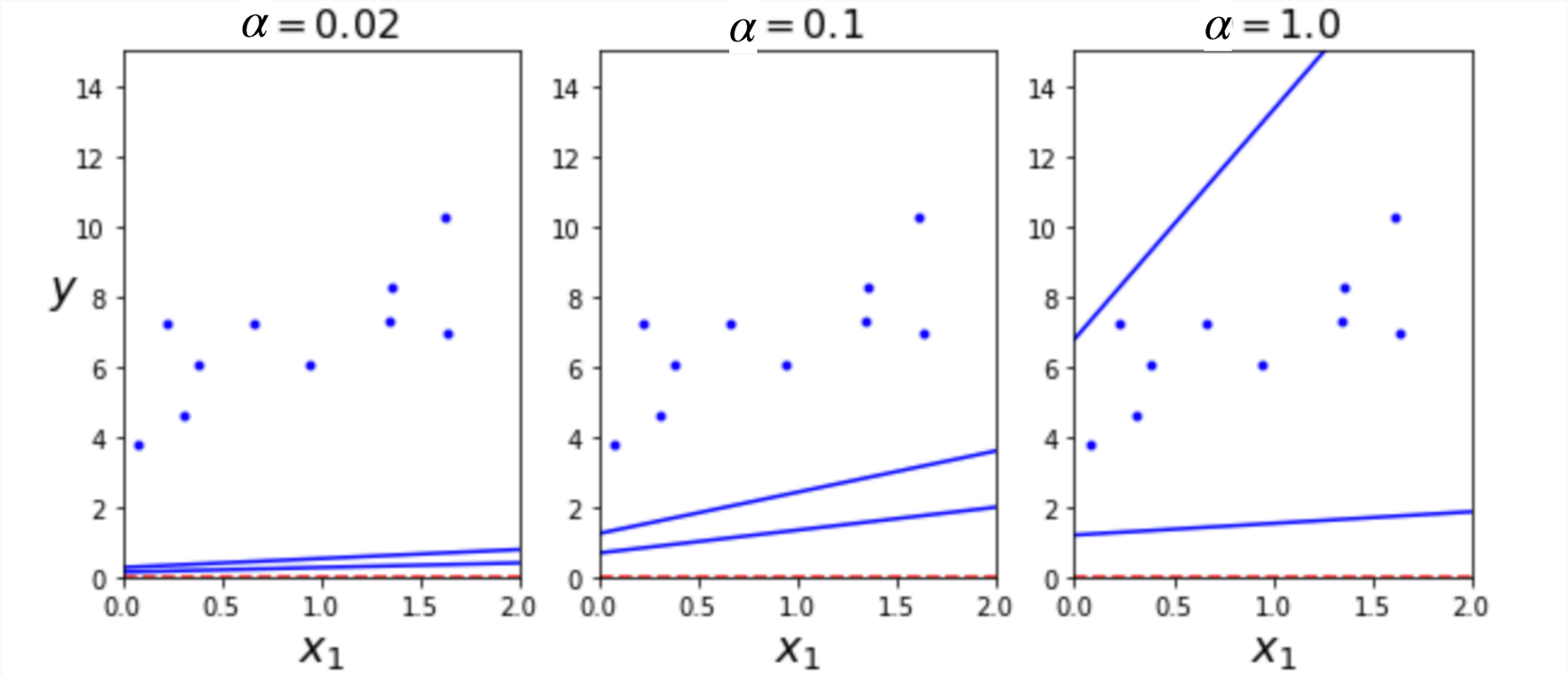
$$\begin{bmatrix}0\\0\end{bmatrix} \to \begin{bmatrix}0.84\\2.04\end{bmatrix} \to \begin{bmatrix}0.25\\0.39\end{bmatrix} \to \begin{bmatrix}0.780\\1.67\end{bmatrix} \to \begin{bmatrix}0.45\\0.623\end{bmatrix} \to \dots$$



$$\begin{bmatrix}0\\0\end{bmatrix} \to \begin{bmatrix}0.98\\2.38\end{bmatrix} \to \begin{bmatrix}0.02\\-0.26\end{bmatrix} \to \begin{bmatrix}1.17\\2.58\end{bmatrix} \to \begin{bmatrix}0.02\\-0.56\end{bmatrix} \to \dots$$



Different learning rates



Not covered in this class for linear regression are heuristics for adaptively changing the learning rate.

You can read on your own the choices available in Sklearn's implementation of SGD for linear regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html#sklearn.linear_model.SGDRegressor

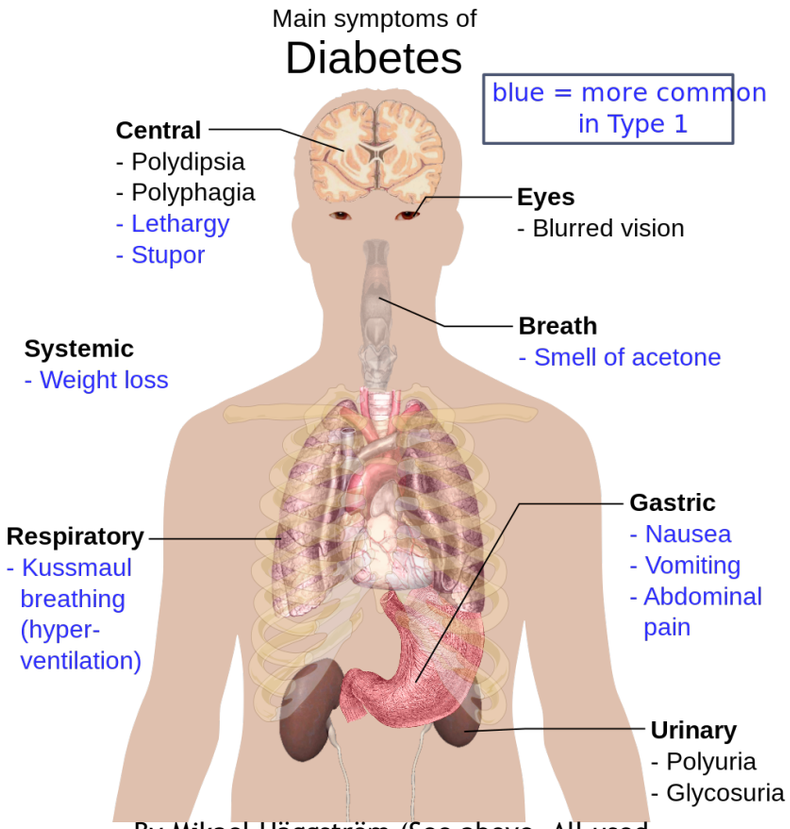
Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
  - When $d = 1$ (simple linear regression)
  - What can go wrong?
  - General case, any $d > 0$ (multiple variable linear regression)
    - Example
    - Algorithm
    - Vectorized Algorithm
- Global Optimization: Normal Equations
- Feature scaling
- Assessing Goodness of Fit
- Extensions
- Removing features
- Objective function revisited: Probabilistic interpretation

How would we modify our algorithm if we had more features? ($d \geq 1$)

Example: Diabetes Patients Progression

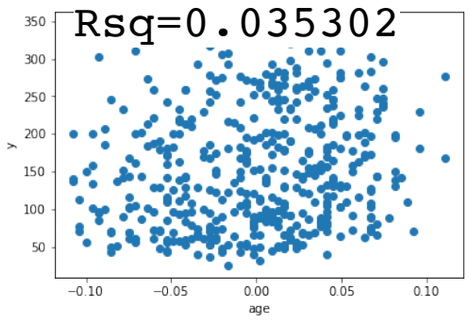
- Can we predict diabetes patients' condition a year after taking 10 baseline measurements?
- The 10 baseline measurements are: age, sex, body mass index, average blood pressure, and 6 blood serum measurements
- Many factors affect diabetes
  - Hard to derive from first principles
  - Difficult to model physiological processes precisely
- Can machine learning help?

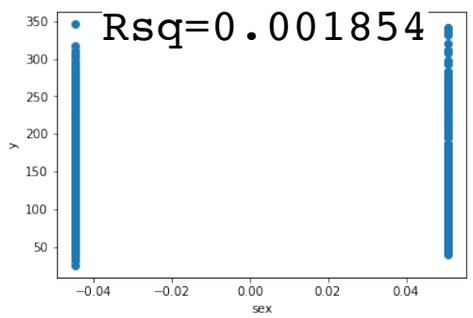


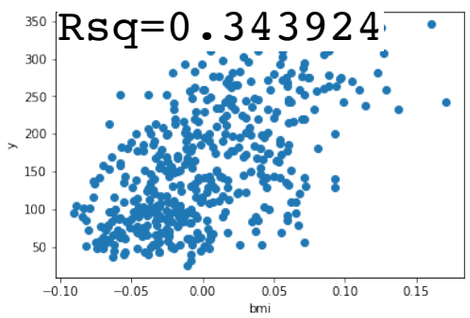
By Mikael Häggström (See above. All used images are in public domain.) [Public domain], via Wikimedia Commons

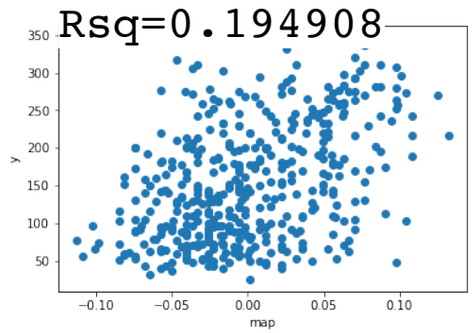
age	sex	bmi	map	tc	ldl	hdl	tch	ltg	glu	age	sex	bmi	map	tc	ldl	hdl	tch	ltg	glu
59	2	32.1	101	157	93.2	38	4	4.8598	87	0.038	0.051	0.062	2.187e-02	-0.044	-3.482e-02	0.043	-0.003	0.020	-0.018
48	1	21.6	87	183	103.2	70	3	3.8918	69	-0.002	-0.045	-0.051	-2.633e-02	-0.008	-1.916e-02	-0.074	-0.039	-0.068	-0.092
72	2	30.5	93	156	93.6	41	4	4.6728	85	0.085	0.051	0.044	-5.670e-03	-0.046	-3.419e-02	0.032	-0.003	0.003	-0.026
24	1	25.3	84	198	131.4	40	5	4.8903	89	-0.089	-0.045	-0.012	-3.666e-02	0.012	2.499e-02	0.036	0.034	0.023	-0.009
50	1	23	101	192	125.4	52	4	4.2905	80	0.005	-0.045	-0.036	2.187e-02	0.004	1.560e-02	-0.008	-0.003	-0.032	-0.047
23	1	22.6	89	139	64.8	61	2	4.1897	68	-0.093	-0.045	-0.041	-1.944e-02	-0.069	-7.929e-02	-0.041	-0.076	-0.041	-0.096
36	2	22	90	160	99.6	50	3	3.9512	82	-0.046	0.051	-0.047	-1.600e-02	-0.040	-2.480e-02	-0.001	-0.039	-0.063	-0.038

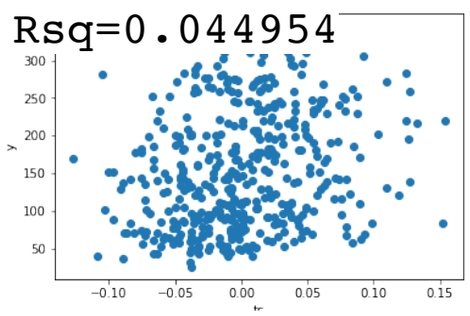
Lets plot all the features

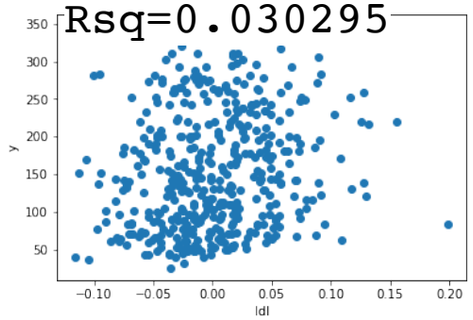


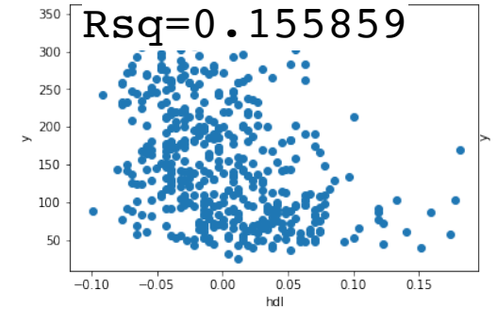


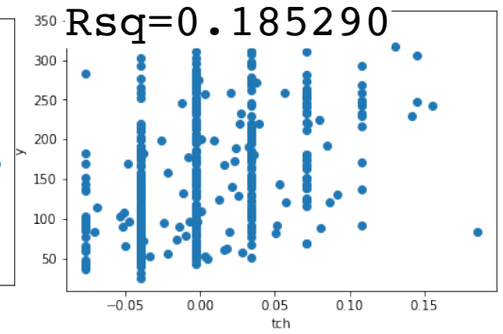


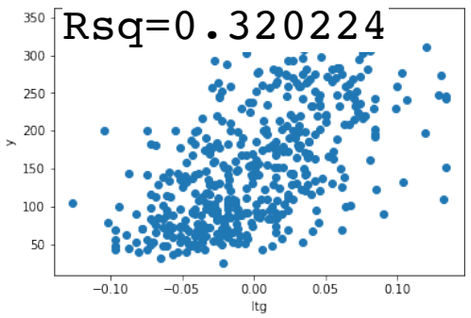


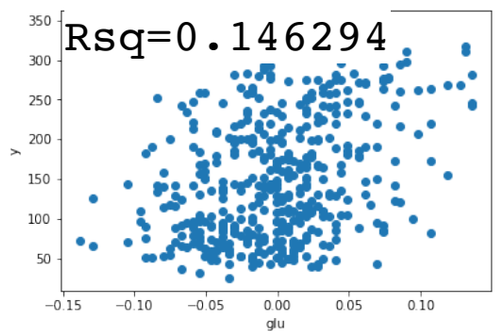












Could we do a better job of predicting if we used more than one feature to make the prediction?

Demo on GitHub

- Code will be posted on Brightspace

Demo: Predicting Glucose Levels using Mulitple Linear Regression

In this demo, you will learn how to:

- Fit multiple linear regression models using python's `sklearn` pachage.
- Split data into training and test.
- Manipulating and visualizing multivariable arrays.

We first load the packages as usual.

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

Diabetes Data Example

To illustrate the concepts, we load the well-known diabetes data set. This dataset is included in the `sklearn.da` can be loaded as follows.

```
from sklearn import datasets, linear_model, preprocessing
```

Loading the Data

```
from sklearn import datasets, linear_model, preprocessing

# Load the diabetes dataset
diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target
```

```
nsamp, natt = X.shape
print("num samples={0:d}  num attributes={1:d}".format(nsamp,natt))
```

```
num samples=442  num attributes=10
```

- Sklearn package:
  - Many methods for machine learning
  - Datasets
  - Will use throughout this class
- Diabetes dataset is one example

Design matrix: Diabetes Patients Progression

$$X = \begin{bmatrix}
\texttt{age} & \texttt{sex} & \texttt{bmi} & \texttt{map} & \texttt{tc} & \texttt{ldl} & \texttt{hdl} & \texttt{tch} & \texttt{ltg} & \texttt{glu} \\
59 & 2 & 32.1 & 101 & 157 & 93.2 & 38 & 4 & 4.8598 & 87 \\
48 & 1 & 21.6 & 87 & 183 & 103.2 & 70 & 3 & 3.8918 & 69 \\
72 & 2 & 30.5 & 93 & 156 & 93.6 & 41 & 4 & 4.6728 & 85 \\
24 & 1 & 25.3 & 84 & 198 & 131.4 & 40 & 5 & 4.8903 & 89 \\
50 & 1 & 23 & 101 & 192 & 125.4 & 52 & 4 & 4.2905 & 80 \\
23 & 1 & 22.6 & 89 & 139 & 64.8 & 61 & 2 & 4.1897 & 68 \\
36 & 2 & 22 & 90 & 160 & 99.6 & 50 & 3 & 3.9512 & 82
\end{bmatrix} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \mathbf{x}^{(3)T} \\ \mathbf{x}^{(4)T} \\ \mathbf{x}^{(5)T} \\ \mathbf{x}^{(6)T} \\ \mathbf{x}^{(7)T} \end{bmatrix}$$

$$\mathbf{x}^{(1)} = \begin{bmatrix} 59 \\ 2 \\ 32.1 \\ 101 \\ 157 \\ 93.2 \\ 38 \\ 4 \\ 4.8598 \\ 87 \end{bmatrix}$$

$$\mathbf{x}^{(1)T} = [59 \quad 2 \quad 32.1 \quad 101 \quad 157 \quad 93.2 \quad 38 \quad 4 \quad 4.8598 \quad 87]$$

Example (using only some of the 442 training examples)

7 x 10 matrix

$$X = \begin{bmatrix} \text{age} & \text{sex} & \text{bmi} & \text{map} & \text{tc} & \text{ldl} & \text{hdl} & \text{tch} & \text{ltg} & \text{glu} \\ 59 & 2 & 32.1 & 101 & 157 & 93.2 & 38 & 4 & 4.8598 & 87 \\ 48 & 1 & 21.6 & 87 & 183 & 103.2 & 70 & 3 & 3.8918 & 69 \\ 72 & 2 & 30.5 & 93 & 156 & 93.6 & 41 & 4 & 4.6728 & 85 \\ 24 & 1 & 25.3 & 84 & 198 & 131.4 & 40 & 5 & 4.8903 & 89 \\ 50 & 1 & 23 & 101 & 192 & 125.4 & 52 & 4 & 4.2905 & 80 \\ 23 & 1 & 22.6 & 89 & 139 & 64.8 & 61 & 2 & 4.1897 & 68 \\ 36 & 2 & 22 & 90 & 160 & 99.6 & 50 & 3 & 3.9512 & 82 \end{bmatrix} \quad y = \begin{bmatrix} 151 \\ 75 \\ 141 \\ 206 \\ 135 \\ 97 \\ 138 \end{bmatrix}$$

$y^{(4)}$

Where is the 4th example?

Where is the 7th feature for all the examples?

What is the 7th feature of the 4th example $x_7^{(4)} = 40$?

What is the measure of disease progression one year after baseline of the 4th example $y^{(4)}$?

$$y^{(4)} = 206$$

Data from https://web.stanford.edu/~hastie/Papers/LARS/diabetes.data

1-indexed example in book and notes (common in math)
0-indexed in python

e (using onl e of the 442 training examples)

7th feature is in the 7th column

7 x 10 matrix

4th example is in the 4th row

$y^{(4)}$

$$X = \begin{bmatrix} \text{age} & \text{sex} & \text{bmi} & \text{map} & \text{tc} & \text{ldl} & \text{hdl} & \text{tch} & \text{ltg} & \text{glu} \\ 59 & 2 & 32.1 & 101 & 157 & 93.2 & 38 & 4 & 4.8598 & 87 \\ 48 & 1 & 21.6 & 87 & 183 & 103.2 & 70 & 3 & 3.8918 & 69 \\ 72 & 2 & 30.5 & 93 & 156 & 93.6 & 41 & 4 & 4.6728 & 85 \\ 24 & 1 & 25.3 & 84 & 198 & 131.4 & 40 & 5 & 4.8903 & 89 \\ 50 & 1 & 23 & 101 & 192 & 125.4 & 52 & 4 & 4.2905 & 80 \\ 23 & 1 & 22.6 & 89 & 139 & 64.8 & 61 & 2 & 4.1897 & 68 \\ 36 & 2 & 22 & 90 & 160 & 99.6 & 50 & 3 & 3.9512 & 82 \end{bmatrix} \quad y = \begin{bmatrix} 151 \\ 75 \\ 141 \\ 206 \\ 135 \\ 97 \\ 138 \end{bmatrix}$$

y, a column vector is an N x 1 matrix. E.g. Y is a 7 x 1 matrix. It is also called a 7 dimensional vector

Where is the 4th example?

Where is the 7th feature for all the examples?

What is the 7th feature of the 4th example $x_7^{(4)} = 40$?

What is the measure of disease progression one year after baseline of the 4th example $y^{(4)}$?

$$y^{(4)} = 206$$

Data from https://web.stanford.edu/~hastie/Papers/LARS/diabetes.data

Standardized Example

X =

age	sex	bmi	map	tc	ldl	hdl	tch	ltg	glu
0.038	0.051	0.062	2.187e-02	-0.044	-3.482e-02	0.043	-0.003	0.020	-0.018
-0.002	-0.045	-0.051	-2.633e-02	-0.008	-1.916e-02	-0.074	-0.039	-0.068	-0.092
0.085	0.051	0.044	-5.670e-03	-0.046	-3.419e-02	0.032	-0.003	0.003	-0.026
-0.089	-0.045	-0.012	-3.666e-02	0.012	2.499e-02	0.036	0.034	0.023	-0.009
0.005	-0.045	-0.036	2.187e-02	0.004	1.560e-02	-0.008	-0.003	-0.032	-0.047
-0.093	-0.045	-0.041	-1.944e-02	-0.069	-7.929e-02	-0.041	-0.076	-0.041	-0.096
-0.046	0.051	-0.047	-1.600e-02	-0.040	-2.480e-02	-0.001	-0.039	-0.063	-0.038

y =

-1.133
-77.133
-11.133
53.866
-17.133
-55.133
-14.133

“data are first standardized to have zero mean and unit L2 norm”

Example $\hat{\mathbf{y}} = X\mathbf{w}$ (using only some of the 442 training examples)

$Xw =$

	age	sex	bmi	map	tc	ldl	hdl	tch	ltg	glu
1	0.038	0.051	0.062	2.187e-02	-0.044	-3.482e-02	0.043	-0.003	0.020	-0.018
1	-0.002	-0.045	-0.051	-2.633e-02	-0.008	-1.916e-02	-0.074	-0.039	-0.068	-0.092
1	0.085	0.051	0.044	-5.670e-03	-0.046	-3.419e-02	0.032	-0.003	0.003	-0.026
1	-0.089	-0.045	-0.012	-3.666e-02	0.012	2.499e-02	0.036	0.034	0.023	-0.009
1	0.005	-0.045	-0.036	2.187e-02	0.004	1.560e-02	-0.008	-0.003	-0.032	-0.047
1	-0.093	-0.045	-0.041	-1.944e-02	-0.069	-7.929e-02	-0.041	-0.076	-0.041	-0.096
1	-0.046	0.051	-0.047	-1.600e-02	-0.040	-2.480e-02	-0.001	-0.039	-0.063	-0.038

$$= \begin{bmatrix} 204.51116637 \\ 67.10485972 \\ 175.02956894 \\ 165.88615565 \\ 123.11207835 \\ 105.64709238 \\ 71.73293158 \end{bmatrix} = \hat{y}$$

$$\begin{bmatrix} 152.34786452 \\ -16.57607993 \\ -254.66532396 \\ 560.98630022 \\ 278.91811152 \\ -393.41357305 \\ 97.05460405 \\ -19.0023093 \\ 169.46450327 \\ 632.95050374 \\ 114.21638941 \end{bmatrix}$$

Think of this as points in d+1 dimensional space

$\hat{\mathbf{y}}$ is a N-dimensional vector

w is a (d+1)-dimensional vector

How do we find this vector?

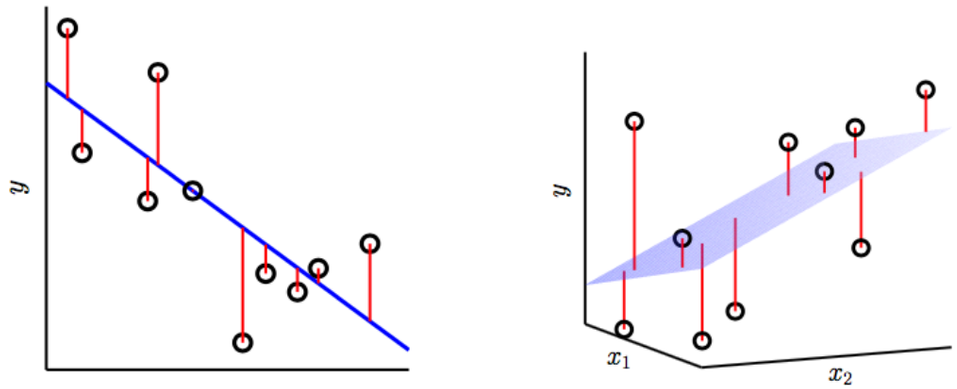
In numpy this is X.dot(w)

Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
  - When $d = 1$ (simple linear regression)
  - What can go wrong?
  - General case, any $d > 0$ (multiple variable linear regression)
    - Example
    - Algorithm
    - Vectorized Algorithm
- Global Optimization: Normal Equations
- Feature scaling
- Assessing Goodness of Fit
- Extensions
- Removing features
- Objective function revisited: Probabilistic interpretation

Least mean squares when $\mathbf{x}$ is d-dimensional

Finding the best line/hyperplane with the smallest squared residuals



$$y = f(\mathbf{x}) + \epsilon \qquad \longleftarrow \text{ noisy target } P(y|\mathbf{x})$$

$$y^{(i)} \approx \hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_d x_d^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)}$$

$$RSS(\mathbf{w}) = \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2 = \sum_{i=1}^{N} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2$$

We imagine that

$y = w_0 + w_1 x_1 + w_2 x_2 + \epsilon$

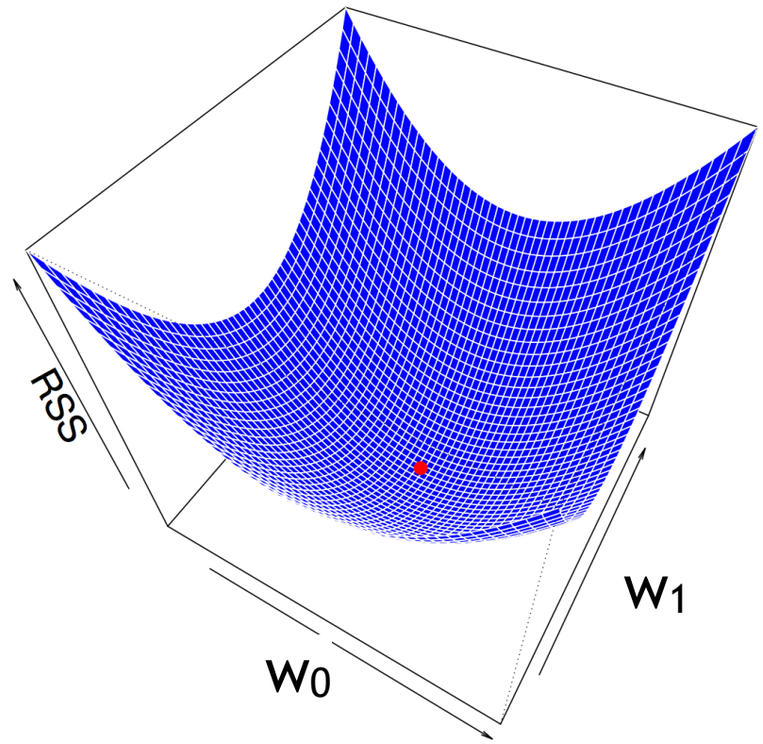
The pictures are from the Malik Magdon-Ismail and Hsuan-Tien Lin's lecture slides

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

Gradient

$$J(\mathbf{w}) = \frac{1}{2N}\text{RSS}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2$$



From calculus we know that the direction for the maximum rate of change for a function J(w) is

$$\nabla J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial w_d} \end{bmatrix}$$

Each coefficient (parameter/weight) should be updated as follows:

$$w_0 - \alpha\frac{\partial J(\mathbf{w})}{\partial w_0}$$

$$w_1 - \alpha\frac{\partial J(\mathbf{w})}{\partial w_1}$$

$$\vdots$$

$$w_d - \alpha\frac{\partial J(\mathbf{w})}{\partial w_d}$$

"Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani "

The partial derivative of J(w)

$$\mathrm{J}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}((\mathbf{w}^T x^{(i)}) - y^{(i)})^2 = \frac{1}{2N}\sum_{i=1}^{N}(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots w_d x_d^{(i)} - y^{(i)})^2$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{N}\cdot\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})x_j^{(i)}$$

Verify at home 😊

$$= \frac{1}{N}\sum_{i=1}^{N}(w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots + w_d x_d^{(i)} - y^{(i)})x_j^{(i)}$$

Slide not presented in class

$$\mathrm{J}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}((\mathbf{w}^T x^{(i)}) - y^{(i)})^2 = \frac{1}{2N}\sum_{i=1}^{N}(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots w_d x_d^{(i)} - y^{(i)})^2$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{2N}\cdot\frac{\partial}{\partial w_j}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2 = \frac{1}{2N}\cdot\sum_{i=1}^{N}\frac{\partial}{\partial w_j}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2N}\cdot\sum_{i=1}^{N}2(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})\frac{\partial}{\partial w_j}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)}) \qquad = \frac{1}{N}\cdot\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})x_j^{(i)}$$

$$= \frac{1}{N}\sum_{i=1}^{N}(w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots + w_d x_d^{(i)} - y^{(i)})x_j^{(i)}$$

Cost function

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

$$\mathrm{J}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}((\mathbf{w}^T x^{(i)}) - y^{(i)})^2 = \frac{1}{2N}\mathrm{np.sum(np.square}(X\mathbf{w} - y)) = \frac{1}{2N}RSS(\mathbf{w})$$

- Both RSS and J are convex function (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 J function wrt the parameters:

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{1}{N}\sum_{i=1}^{N}(w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots + w_d x_d^{(i)} - y_i)x_{i0} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T x^{(i)} - y^{(i)})x_0^{(i)}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots + w_d x_d^{(i)} - y^{(i)})x_j^{(i)} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T x^{(i)} - y^{(i)})x_j^{(i)}$$

individual slope

average slope

updating by α·(averaging the slope over all training

$x_0^{(i)} = 1$

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

Gradient Descent Optimization

- Loss/cost function (objective) $\frac{1}{2N}\text{RSS}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2$

- The gradient is: $\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})x_j^{(i)}$

- To decrease the cost, we update the parameters, $\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$, using the update rule: $w_j - \alpha\frac{\partial J(\mathbf{w})}{\partial w_j}$

$x_0^{(i)} = 1$

for $i = 1$ to num_iter

$$temp0 = w_0 - \alpha\frac{\partial J(\mathbf{w})}{\partial w_0} = w_0 - \frac{\alpha}{N}\sum_{i=1}^{N}(w^T\mathbf{x}^{(i)} - y^{(i)})x_0^{(i)}$$

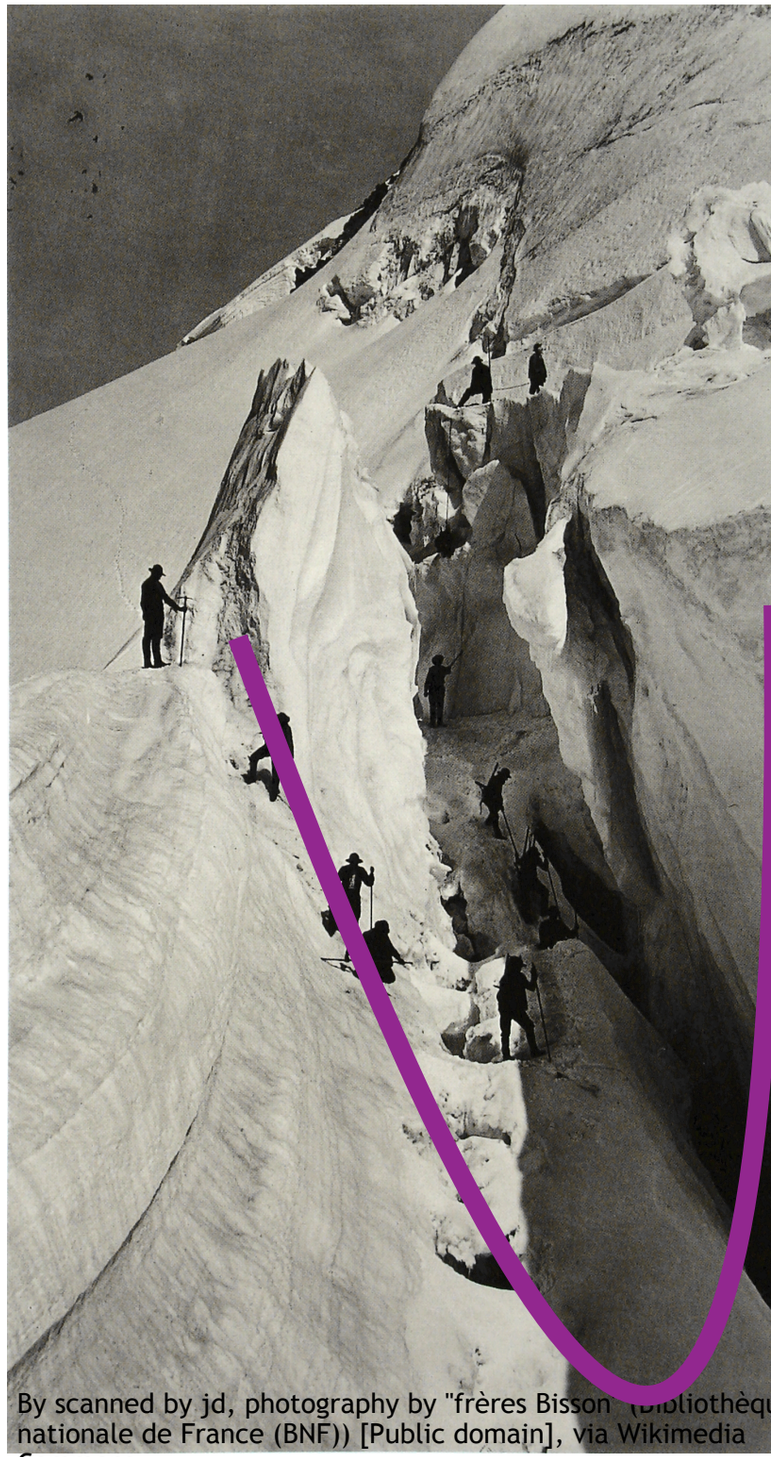
$$temp1 = w_1 - \alpha\frac{\partial J(\mathbf{w})}{\partial w_1} = w_1 - \frac{\alpha}{N}\sum_{i=1}^{N}(w^T\mathbf{x}^{(i)} - y^{(i)})x_1^{(i)}$$

$$\vdots$$

$$tempd = w_d - \alpha\frac{\partial J(\mathbf{w})}{\partial w_d} = w_d - \frac{\alpha}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})x_d^{(i)}$$

$$w_0 = temp0$$
$$w_1 = temp1$$
$$\vdots$$
$$w_d = tempd$$

Simultaneous update



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

Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
  - When $d = 1$ (simple linear regression)
  - What can go wrong?
  - General case, any $d > 0$ (multiple variable linear regression)
    - Example
    - Algorithm
    - Vectorized Algorithm
- Global Optimization: Normal Equations
- Feature scaling
- Assessing Goodness of Fit
- Extensions
- Removing features
- Objective function revisited: Probabilistic interpretation

Vectorized Implementation

$$J(\mathbf{w}) = \frac{1}{2N}\text{RSS}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2$$

Small example to show $\nabla J(\mathbf{w}) = \frac{1}{N}X^T(X\mathbf{w} - \mathbf{y})$ if we have one feature: $\hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)}$.

The partial derivative of our cost function is: $\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})x_j^{(i)}$

If we have one feature and three examples ($N = 3$) the partial derivatives are:

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{1}{3}\Big(\underbrace{(\mathbf{w}^T\mathbf{x}^{(1)})}_{\hat{y}^{(1)}} - y^{(1)}) + \underbrace{(\mathbf{w}^T\mathbf{x}^{(2)})}_{\hat{y}^{(2)}} - y^{(2)}) + \underbrace{(\mathbf{w}^T\mathbf{x}^{(3)})}_{\hat{y}^{(3)}} - y^{(3)}) \Big) = \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}\hat{y}^{(1)} - y^{(1)} \\ \hat{y}^{(2)} - y^{(2)} \\ \hat{y}^{(3)} - y^{(3)}\end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{1}{3}\Big(\underbrace{(\mathbf{w}^T\mathbf{x}^{(1)})}_{\hat{y}^{(1)}} - y^{(1)})x_1^{(1)} + \underbrace{(\mathbf{w}^T\mathbf{x}^{(2)})}_{\hat{y}^{(2)}} - y^{(2)})x_1^{(2)} + \underbrace{(\mathbf{w}^T\mathbf{x}^{(3)})}_{\hat{y}^{(3)}} - y^{(3)})x_1^{(3)} \Big) = \frac{1}{3}\begin{bmatrix}x_1^{(1)} & x_1^{(2)} & x_1^{(3)}\end{bmatrix}\begin{bmatrix}\hat{y}^{(1)} - y^{(1)} \\ \hat{y}^{(2)} - y^{(2)} \\ \hat{y}^{(3)} - y^{(3)}\end{bmatrix}$$

Thus we can write the gradient for these three examples as:

$$\nabla J(\mathbf{w}) = \begin{bmatrix}\frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1}\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1 & 1 & 1 \\ x_1^{(1)} & x_1^{(2)} & x_1^{(3)}\end{bmatrix}\begin{bmatrix}\hat{y}^{(1)} - y^{(1)} \\ \hat{y}^{(2)} - y^{(2)} \\ \hat{y}^{(3)} - y^{(3)}\end{bmatrix}$$

Calculations continued on the next slide

$$J(\mathbf{w}) = \frac{1}{2N}\text{RSS}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2$$

Small example continued:

Using the fact that $X\mathbf{w} = \hat{\mathbf{y}}$, $d = 1$, and $N = 3$ we can show $\nabla J(\mathbf{w}) = \frac{1}{N}X^T(X\mathbf{w} - \mathbf{y})$

$$\nabla J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1} \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ x_1^{(1)} & x_1^{(2)} & x_1^{(3)} \end{bmatrix}\begin{bmatrix} \hat{y}^{(1)} - y^{(1)} \\ \hat{y}^{(2)} - y^{(2)} \\ \hat{y}^{(3)} - y^{(3)} \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ x_1^{(1)} & x_1^{(2)} & x_1^{(3)} \end{bmatrix}\left(\begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ 1 & x_1^{(3)} \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \end{bmatrix}\right) = \frac{1}{N}X^T(X\mathbf{w} - \mathbf{y})$$

$$J(\mathbf{w}) = \frac{1}{2N}\text{RSS}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2$$

2nd small example:

The partial derivative of our objective function $\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})x_j^{(i)}$

If we have two features and three examples ($N = 3$) the partial derivatives are:

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{1}{3}\Big((\mathbf{w}^T\mathbf{x}^{(1)} - y^{(1)}) + (\mathbf{w}^T\mathbf{x}^{(2)} - y^{(2)}) + (\mathbf{w}^T\mathbf{x}^{(3)} - y^{(3)}) \Big) = \frac{1}{3}[1 \quad 1 \quad 1]\begin{bmatrix}\hat{y}^{(1)} - y^{(1)} \\ \hat{y}^{(2)} - y^{(2)} \\ \hat{y}^{(3)} - y^{(3)}\end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{1}{3}\Big((\mathbf{w}^T\mathbf{x}^{(1)} - y^{(1)})x_1^{(1)} + (\mathbf{w}^T\mathbf{x}^{(2)} - y^{(2)})x_1^{(2)} + (\mathbf{w}^T\mathbf{x}^{(3)} - y^{(3)})x_1^{(3)} \Big) = \frac{1}{3}\begin{bmatrix}x_1^{(1)} & x_1^{(2)} & x_1^{(3)}\end{bmatrix}\begin{bmatrix}\hat{y}^{(1)} - y^{(1)} \\ \hat{y}^{(2)} - y^{(2)} \\ \hat{y}^{(3)} - y^{(3)}\end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_2} = \frac{1}{3}\Big((\mathbf{w}^T\mathbf{x}^{(1)} - y^{(1)})x_2^{(1)} + (\mathbf{w}^T\mathbf{x}^{(2)} - y^{(2)})x_2^{(2)} + (\mathbf{w}^T\mathbf{x}^{(3)} - y^{(3)})x_2^{(3)} \Big) = \frac{1}{3}\begin{bmatrix}x_2^{(1)} & x_2^{(2)} & x_2^{(3)}\end{bmatrix}\begin{bmatrix}\hat{y}^{(1)} - y^{(1)} \\ \hat{y}^{(2)} - y^{(2)} \\ \hat{y}^{(3)} - y^{(3)}\end{bmatrix}$$

Thus we can write the gradient for these three examples as:

$$\nabla J(\mathbf{w}) = \begin{bmatrix}\frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1} \\ \frac{\partial J(\mathbf{w})}{\partial w_2}\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1 & 1 & 1 \\ x_1^{(1)} & x_1^{(2)} & x_1^{(3)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)}\end{bmatrix}\begin{bmatrix}\hat{y}^{(1)} - y^{(1)} \\ \hat{y}^{(2)} - y^{(2)} \\ \hat{y}^{(3)} - y^{(3)}\end{bmatrix} = \frac{1}{N}X^T(X\mathbf{w} - \mathbf{y})$$

Example

$$J(\mathbf{w}) = \frac{1}{2N}\text{RSS}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})x_j^{(i)} = \frac{1}{N}\sum_{i=1}^{N}(-\epsilon^{(i)})x_j^{(i)}$$

$$\nabla J(\mathbf{w}) = \frac{1}{N}X^T(X\mathbf{w} - \mathbf{y}) = \frac{1}{N}X^T(\hat{\mathbf{y}} - \mathbf{y})$$

$$\nabla J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial w_d} \end{bmatrix} = \frac{1}{N}$$

1	1	1	1	1	1	1	
0.038	−0.002	0.085	−0.089	0.005	−0.093	−0.045	⋮
0.051	−0.045	0.051	−0.045	−0.045	−0.045	0.051	
0.062	−0.051	0.044	−0.012	−0.036	−0.041	−0.047	
0.022	−0.026	−0.006	−0.037	0.022	−0.019	−0.016	
−0.044	−0.008	−0.046	0.012	0.004	−0.069	−0.04	
−0.035	−0.019	−0.034	0.025	0.016	−0.079	−0.025	
−0.043	0.074	−0.032	−0.036	0.008	0.041	0.001	
−0.003	−0.039	−0.003	0.034	−0.003	−0.076	−0.039	
0.02	−0.068	0.003	0.023	−0.032	−0.041	−0.063	
−0.018	−0.092	−0.026	−0.009	−0.047	−0.096	−0.038	

age	sex	bmi	map	tc	ldl	hdl	tch	ltg	glu
1	0.038	0.051	0.062	$2.187e-02$	−0.044	$-3.482e-02$	0.043	−0.003	0.020	−0.018
1	−0.002	−0.045	−0.051	$-2.633e-02$	−0.008	$-1.916e-02$	−0.074	−0.039	−0.068	−0.092
1	0.085	0.051	0.044	$-5.670e-03$	−0.046	$-3.419e-02$	0.032	−0.003	0.003	−0.026
1	−0.089	−0.045	−0.012	$-3.666e-02$	0.012	$2.499e-02$	0.036	0.034	0.023	−0.009
1	0.005	−0.045	−0.036	$2.187e-02$	0.004	$1.560e-02$	−0.008	−0.003	−0.032	−0.047
1	−0.093	−0.045	−0.041	$-1.944e-02$	−0.069	$-7.929e-02$	−0.041	−0.076	−0.041	−0.096
1	−0.046	0.051	−0.047	$-1.600e-02$	−0.040	$-2.480e-02$	−0.001	−0.039	−0.063	−0.038
⋮										

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \end{bmatrix} - \begin{bmatrix} 204.51116637 \\ 67.10485972 \\ 175.02956894 \\ 165.88615565 \\ 123.11207835 \\ 105.64709238 \\ 71.73293158 \end{bmatrix}$$

$$= \frac{1}{N}$$

1	1	1	1	1	1	1	
0.038	−0.002	0.085	−0.089	0.005	−0.093	−0.045	⋮
0.051	−0.045	0.051	−0.045	−0.045	−0.045	0.051	
0.062	−0.051	0.044	−0.012	−0.036	−0.041	−0.047	
0.022	−0.026	−0.006	−0.037	0.022	−0.019	−0.016	
−0.044	−0.008	−0.046	0.012	0.004	−0.069	−0.04	
−0.035	−0.019	−0.034	0.025	0.016	−0.079	−0.025	
−0.043	0.074	−0.032	−0.036	0.008	0.041	0.001	
−0.003	−0.039	−0.003	0.034	−0.003	−0.076	−0.039	
0.02	−0.068	0.003	0.023	−0.032	−0.041	−0.063	
−0.018	−0.092	−0.026	−0.009	−0.047	−0.096	−0.038	

$$\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(3)} \\ \hat{y}^{(4)} \\ \hat{y}^{(5)} \\ \hat{y}^{(6)} \\ \hat{y}^{(7)} \end{bmatrix} - \begin{bmatrix} 204.51116637 \\ 67.10485972 \\ 175.02956894 \\ 165.88615565 \\ 123.11207835 \\ 105.64709238 \\ 71.73293158 \end{bmatrix}$$

$$= \frac{1}{N}$$

1	1	1	1	1	1	1	
0.038	−0.002	0.085	−0.089	0.005	−0.093	−0.045	⋮
0.051	−0.045	0.051	−0.045	−0.045	−0.045	0.051	
0.062	−0.051	0.044	−0.012	−0.036	−0.041	−0.047	
0.022	−0.026	−0.006	−0.037	0.022	−0.019	−0.016	
−0.044	−0.008	−0.046	0.012	0.004	−0.069	−0.04	
−0.035	−0.019	−0.034	0.025	0.016	−0.079	−0.025	
−0.043	0.074	−0.032	−0.036	0.008	0.041	0.001	
−0.003	−0.039	−0.003	0.034	−0.003	−0.076	−0.039	
0.02	−0.068	0.003	0.023	−0.032	−0.041	−0.063	
−0.018	−0.092	−0.026	−0.009	−0.047	−0.096	−0.038	

$$\begin{bmatrix} -\epsilon^{(1)} \\ -\epsilon^{(2)} \\ -\epsilon^{(3)} \\ -\epsilon^{(4)} \\ -\epsilon^{(5)} \\ -\epsilon^{(6)} \\ -\epsilon^{(7)} \end{bmatrix} = \nabla J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial w_d} \end{bmatrix}$$

Gradient Descent Optimization Using Vectorization

$$J(\mathbf{w}) = \frac{1}{2N}\text{RSS}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2 \qquad \nabla J(\mathbf{w}) = \frac{1}{N}X^T(X\mathbf{w} - \mathbf{y})$$

- We can perform gradient updates for all parameters $\mathbf{w} = [w_0, w_1, \ldots, w_d]$ in a single line of vectorized code

for $i = 1$ to num_iter

$$\mathbf{w} = \mathbf{w} - \alpha\nabla J(\mathbf{w}) = \mathbf{w} - \alpha\frac{1}{N}X^T(X\mathbf{w} - \mathbf{y})$$

Simultaneous update

- Note that:
  - $\mathbf{w}$ and $\nabla J(\mathbf{w})$ are $(d+1)\times 1$ column vectors
  - all $\mathbf{w}$ are getting updated simultaneously, and we do not need to store them in a temporary variable

By scanned by jd, photography by "frères Bi[illegible]n" (Bibli[illegible]thèqu nationale de France (BNF)) [Public domain], vi[illegible]edia Commons

Running time:
$O(Nd$ #iter$)$

Outline

- Motivating Example: Predicting the mpg of a car
- Model: Linear Model
- Objective function: Least Squares Fit Problem
- Local Optimizer: Gradient Descent
  - When $d = 1$ (simple linear regression)
  - What can go wrong?
  - General case, any $d > 0$ (multiple variable linear regression)
    - Example
    - Algorithm
    - Vectorized Algorithm
- Global Optimization: Normal Equations
- Feature scaling
- Assessing Goodness of Fit
- Extensions
- Removing features
- Objective function revisited: Probabilistic interpretation

Global Optimization

Our second method to finding the parameters $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$

Updated loss function

General ML problem

- Get data
- Pick a model with parameters
- Pick a loss function
  - Measures goodness of fit model to data
  - Function of the parameters
- Find parameters that minimizes loss

Multiple linear regression

Data: $(\mathbf{x}^{(i)}, y^{(i)}), i = 1,2,\ldots,N$

Linear model:

$$\hat{y}^{(j)} = w_0 + w_1 x_1^{(j)} + w_2 x_2^{(j)} + \cdots + w_d x_d^{(j)}$$

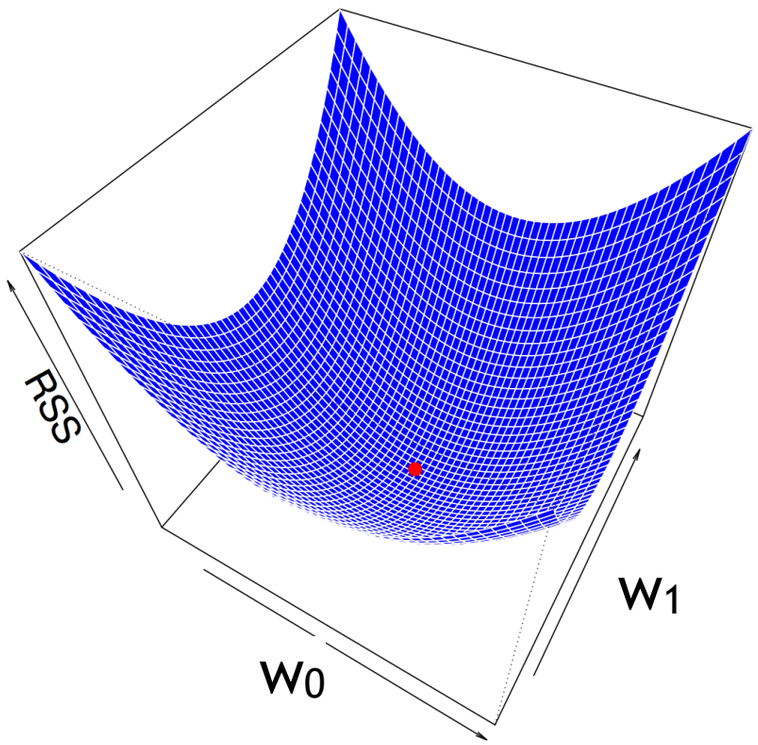
Loss function: $\frac{\text{RSS}(\mathbf{w})}{2N} = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$

$$= \frac{1}{2N}\left[(y^{(1)} - \hat{y}^{(1)})^2 + (y^{(2)} - \hat{y}^{(2)})^2 + \cdots + (y^{(N)} - \hat{y}^{(N)})^2\right]$$

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

Global optimization for finding **w** to minimize

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$



$$\nabla J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial w_d} \end{bmatrix} = \frac{1}{N} X^T (X\mathbf{w} - \mathbf{y}) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Goal find $\mathbf{w}$ such that $\nabla J(\mathbf{w}) = \mathbf{0}$

Finding **w** to minimize J(w)

Goal find $\mathbf{w}$ such that $\nabla J(\mathbf{w}) = \mathbf{0}$

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

Setting $\nabla J(\mathbf{w}) = \frac{1}{N}(X^T X\mathbf{w} - X^T\mathbf{y}) = \mathbf{0}$

Results in: $X^T X\mathbf{w} = X^T\mathbf{y}$

Thus $\mathbf{w}_{\text{lin}} = \underbrace{\left(X^T X\right)^{-1} X^T}\,\mathbf{y}$

pseudoinverse
left inverse

We added 'lin' to $\mathbf{w}$ to specify it was linear regression

Running time:
$O(Nd^2)$ for $X^T X$
$O(d^3)$ for inverse $(X^T X)^{-1}$
$O(dN)$ for $X^T\mathbf{y}$
$O(d^2)$ for $(X^T X)^{-1}$ times $X^T\mathbf{y}$
————————————————————
——
Total $O(Nd^2) + O(d^3)$

*finding the inverse (or pseudoinverse) of a $d \times d$ matrix can be found in $O(d^{2.373})$ time

If the columns of the matrix X are linearly independent then $X^T X$ is invertible and

$$X^+ = (X^T X)^{-1} X^T$$

is the pseudo inverse, i.e. the left inverse of X

$$X^+ X = (X^T X)^{-1} X^T X = I$$

http://sci.utah.edu/~gerig/CS6640-F2012/Materials/pseudoinverse-cis61009sl10.pdf

Finding **w** to minimize E_{in} (and RSS)

Linear Regression Algorithm:

1. Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$, where each $\mathbf{x}$ includes the $x_0 = 1$ coordinate,

$$\underbrace{\mathrm{X} = \begin{bmatrix} —\mathbf{x}_1— \\ —\mathbf{x}_2— \\ \vdots \\ —\mathbf{x}_N— \end{bmatrix}}_{\text{data matrix}}, \qquad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

2. Compute the pseudo inverse $\mathrm{X}^\dagger$ of the matrix X. If $\mathrm{X}^\mathrm{T}\mathrm{X}$ is invertible,

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

3. Return $\mathbf{w}_{\text{lin}} = \mathrm{X}^\dagger\mathbf{y}$.

Algorithm from book Learning form Data

Example

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

$$\mathbf{w}_{\text{lin}} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \end{bmatrix} = \left[\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 0.038 & -0.002 & 0.085 & -0.089 & 0.005 & -0.093 & -0.045 & \\ 0.051 & -0.045 & 0.051 & -0.045 & -0.045 & -0.045 & 0.051 & \\ 0.062 & -0.051 & 0.044 & -0.012 & -0.036 & -0.041 & -0.047 & \\ 0.022 & -0.026 & -0.006 & -0.037 & 0.022 & -0.019 & -0.016 & \\ -0.044 & -0.008 & -0.046 & 0.012 & 0.004 & -0.069 & -0.04 & \\ -0.035 & -0.019 & -0.034 & 0.025 & 0.016 & -0.079 & -0.025 & \\ -0.043 & 0.074 & -0.032 & -0.036 & 0.008 & 0.041 & 0.001 & \\ -0.003 & -0.039 & -0.003 & 0.034 & -0.003 & -0.076 & -0.039 & \\ 0.02 & -0.068 & 0.003 & 0.023 & -0.032 & -0.041 & -0.063 & \\ -0.018 & -0.092 & -0.026 & -0.009 & -0.047 & -0.096 & -0.038 & \end{bmatrix} \begin{bmatrix} & \texttt{age} & \texttt{sex} & \texttt{bmi} & \texttt{map} & \texttt{tc} & \texttt{ldl} & \texttt{hdl} & \texttt{tch} & \texttt{ltg} & \texttt{glu} \\ 1 & 0.038 & 0.051 & 0.062 & 2.187e-02 & -0.044 & -3.482e-02 & 0.043 & -0.003 & 0.020 & -0.018 \\ 1 & -0.002 & -0.045 & -0.051 & -2.633e-02 & -0.008 & -1.916e-02 & -0.074 & -0.039 & -0.068 & -0.092 \\ 1 & 0.085 & 0.051 & 0.044 & -5.670e-03 & -0.046 & -3.419e-02 & 0.032 & -0.003 & 0.003 & -0.026 \\ 1 & -0.089 & -0.045 & -0.012 & -3.666e-02 & 0.012 & 2.499e-02 & 0.036 & 0.034 & 0.023 & -0.009 \\ 1 & 0.005 & -0.045 & -0.036 & 2.187e-02 & 0.004 & 1.560e-02 & -0.008 & -0.003 & -0.032 & -0.047 \\ 1 & -0.093 & -0.045 & -0.041 & -1.944e-02 & -0.069 & -7.929e-02 & -0.041 & -0.076 & -0.041 & -0.096 \\ 1 & -0.046 & 0.051 & -0.047 & -1.600e-02 & -0.040 & -2.480e-02 & -0.001 & -0.039 & -0.063 & -0.038 \\ \vdots & & & & & & & & & & \end{bmatrix} \right]^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 0.038 & -0.002 & 0.085 & -0.089 & 0.005 & -0.093 & -0.045 & \\ 0.051 & -0.045 & 0.051 & -0.045 & -0.045 & -0.045 & 0.051 & \\ 0.062 & -0.051 & 0.044 & -0.012 & -0.036 & -0.041 & -0.047 & \\ 0.022 & -0.026 & -0.006 & -0.037 & 0.022 & -0.019 & -0.016 & \\ -0.044 & -0.008 & -0.046 & 0.012 & 0.004 & -0.069 & -0.04 & \\ -0.035 & -0.019 & -0.034 & 0.025 & 0.016 & -0.079 & -0.025 & \\ -0.043 & 0.074 & -0.032 & -0.036 & 0.008 & 0.041 & 0.001 & \\ -0.003 & -0.039 & -0.003 & 0.034 & -0.003 & -0.076 & -0.039 & \\ 0.02 & -0.068 & 0.003 & 0.023 & -0.032 & -0.041 & -0.063 & \\ -0.018 & -0.092 & -0.026 & -0.009 & -0.047 & -0.096 & -0.038 & \end{bmatrix} \begin{bmatrix} 204.51116637 \\ 67.10485972 \\ 175.02956894 \\ 165.88615565 \\ 123.11207835 \\ 105.64709238 \\ 71.73293158 \end{bmatrix}$$

$$\mathbf{w}_{\text{lin}} = \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \end{bmatrix} = \begin{bmatrix} 152.34786452 \\ -16.57607993 \\ -254.66532396 \\ 560.98630022 \\ 278.91811152 \\ -393.41357305 \\ 97.05460405 \\ -19.0023093 \\ 169.46450327 \\ 632.95050374 \\ 114.21638941 \end{bmatrix}$$

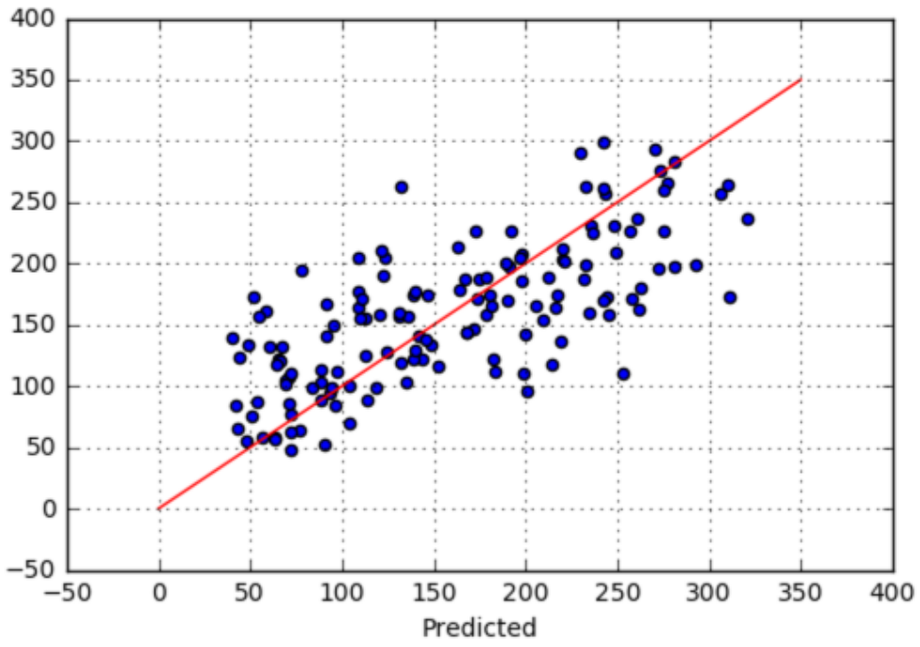
https://numpy.org/doc/stable/reference/generated/numpy.linalg.pinv.html

Python code: `w_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)`

Python code using the pseudoinverse: `w_best = np.linalg.pinv(X_b).dot(y)`

Best fitting coefficients/weights

- The difference between the predicted and the true values



$$w = \begin{bmatrix} 152.34786452 \\ -16.57607993 \\ -254.66532396 \\ 560.98630022 \\ 278.91811152 \\ -393.41357305 \\ 97.05460405 \\ -19.0023093 \\ 169.46450327 \\ 632.95050374 \\ 114.21638941 \end{bmatrix}$$

$R^2 = 0.514719$