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.

Gradient Descent: http://theory.stanford.edu/~tim/s16/l/l5.pdf

Topic 2
Multiple Linear Regression continued

INTRODUCTION TO MACHINE LEARNING

Learning Objectives

- ❑ Formulate a machine learning model as a multiple linear regression model
  - Identify features and label/target for the problem
- ❑ Understand the data matrix/design matrix
- ❑ Find the solution using gradient descent
- ❑ Write the regression model in vectorized form
- ❑ Derive the closed form solution for multiple linear regression
- ❑ Compute the least-squares solution for the regression coefficients on training data
- ❑ Understand feature scaling/data normalization
- ❑ Added: Understand how to find objective function using MLE

Outline

- ❑Motivating Example: Understanding glucose levels in diabetes patients
- ❑Multiple variable linear models
- ❑Least squares solutions
  - Gradient descent
  - Normal Equations
  - Feature scaling
- ❑Evaluating our hypothesis/Computing the solutions in python
- ❑Special case: Simple linear regression
- ❑ Rethinking the objective function
- ❑Extensions
- ❑Removing features

Why do we transform features

This is a very brief introduction to the topic. We will transform before training.

Warning - all transformations needs to also be performed on any test/validation/new data.

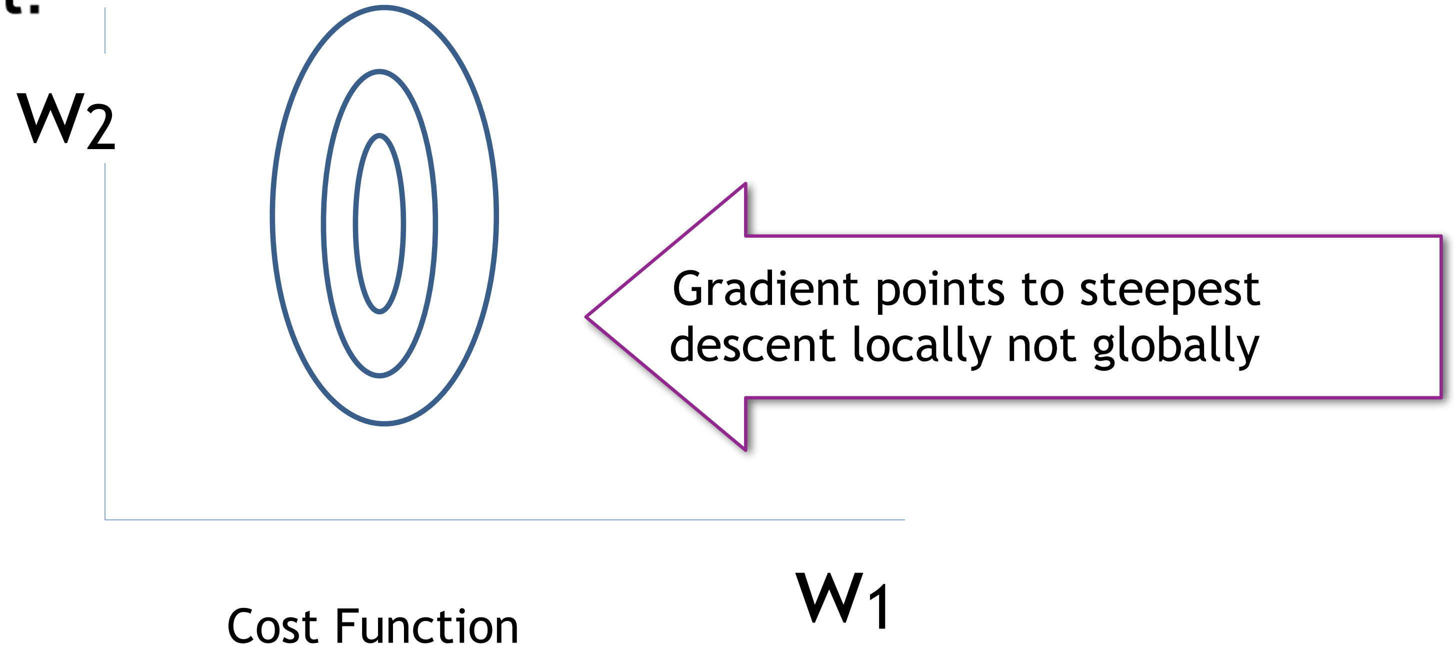
- Necessary transformations
  - Some algorithms require numeric data
  - Some algorithms expect the input to be of a specific size
- Optional transformations
  - Normalizing numeric features
  - Creating non-linearities in the feature space - thus allowing us to still use linear models (next topic)

Normalization

Goal: transform features to have a similar scale

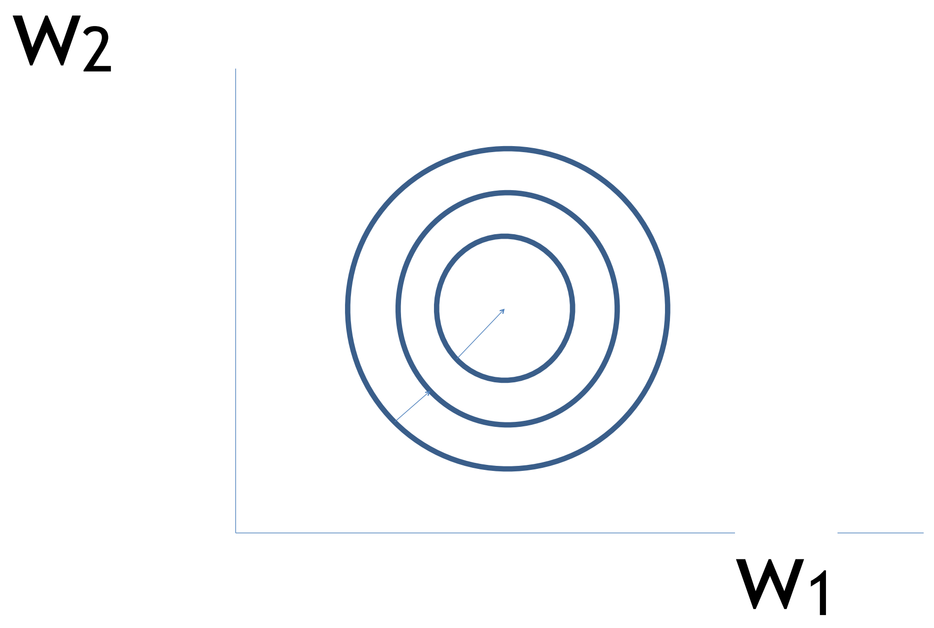
Cost Function

- Visualizing cost function using a contour plot.



Cost Function

After Scaling



Feature Scaling
AKA Data Normalization

- Before applying many machine learning algorithms make sure that the features that are on a similar scale to prevent one feature from overly influencing the algorithm.
- Feature scaling is typically done before performing gradient descent to improve the rate the algorithm converges
- Eg: Say you are using 2 features for predicting housing price problem:
  - X_1 = size (0 – 4000 sq ft)
  - X_2 = No. of bedrooms (1 – 5)

Types of Feature Scaling
AKA Data Normalization

- Min/Max Normalization: scaling to a range
- Standardization:
- For other methods see https://en.wikipedia.org/wiki/Feature_scaling

Min/Max Normalization

```
from sklearn.preprocessing import MinMaxScaler
data = np.array([[10, -10], [20, 10], [30, 30]])
test=np.array([[21,11],[22,12]])
scaler = MinMaxScaler()
scaler.fit(data)
scaled_data = scaler.transform(data)
 scaled_test = scaler.transform(test)
```

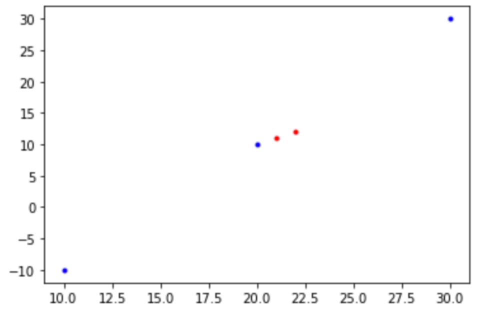
- Scale the range of features to become [0,1] (or [a,b])
- Steps:

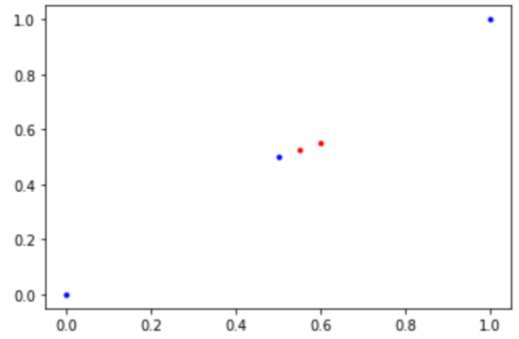
For each feature, j, in the data (except x_0)

- Find the range of values $[\min(x_j), \max(x_j)]$ for the jth feature
- Update every example's jth feature:

$$x_j^{(i)} = \frac{x_j^{(i)} - \min(x_j)}{\max(x_j) - \min(x_j)}$$

$$x_1^{(i)} = \frac{x_1^{(i)} - (10)}{30 - (10)} \qquad x_2^{(i)} = \frac{x_2^{(i)} - (-10)}{30 - (-10)}$$





Eg: If the range of feature 1 is [0, 4000]

update all training example such that $x^{(i)}_1 = (x^{(i)}_1 - 0)/4000$. Now the range of $x^{(i)}_1$ is [0,1]

If the range if the second feature is [1,5] update all training examples $x^{(i)}_2 = (x^{(i)}_2 - 1)/4$.
Now the range of the second feature is [0, 1]

Min/Max Normalization

- Scale the range of features to become [0,1] (or [a,b])
- Steps:

For each feature, j, in the data (except x_0)

- Find the range of values $[\min(x_j), \max(x_j)]$ for the jth feature
- Update every example's jth feature:

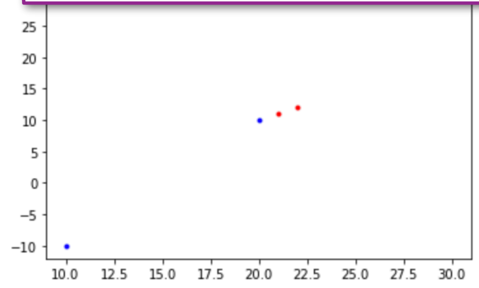
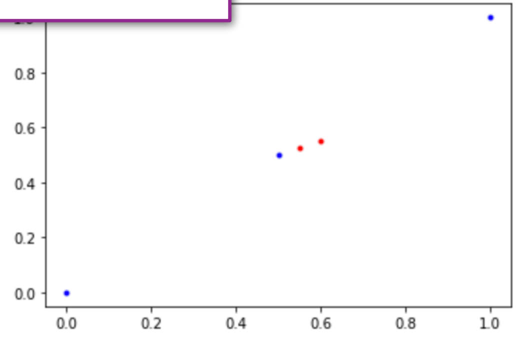
$$x_j^{(i)} = \frac{x_j^{(i)} - \min(x_j)}{\max(x_j) - \min(x_j)}$$

```
fr                                    MinMaxScaler
                                      , [30, 30]])
sc                                    a)
                                      )
```

$$\frac{x_2^{(i)} - (-10)}{30 - (-10)}$$

Pair share: What happens if there is an outlier?
i.e. if you have a few large value and many small values?

Not covered in class: Feature clipping: caps all features above (or below) a certain value to a fixed value.

Eg: If the range of feature 1 is [0, 4000]

update all training example such that $x^{(i)}_1 = (x^{(i)}_1 - 0)/4000$. Now the range of $x^{(i)}_1$ is [0,1]

If the range if the second feature is [1,5] update all training examples $x^{(i)}_2 = (x^{(i)}_2 - 1)/4$.
Now the range of the second feature is [0, 1]

Standardization (in social science this is call Z-score normalization)

```
from sklearn.preprocessing import StandardScaler
data = [[0, 0], [0, 0], [1, 1], [1, 1]]
scaler = StandardScaler()
print(scaler.fit(data))
print(scaler.mean_)
print(scaler.transform(data))
```

```
[0.5 0.5]
```

```
[[-1. -1.]
 [-1. -1.]
 [ 1.  1.]
 [ 1.  1.]]
```

```
print(scaler.transform([[2, 2]]))
```

```
[[3. 3.]]
```

$$x_1^{(i)} = \frac{x_1^{(i)} - 0.5}{0.5}$$

- Scale the range of features to become zero centered and have unit variance
- Steps:

  For each feature, j, in the data

  - Find the average for the jth feature: $\text{ave}(x_j) = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}$
  - Find the standard deviation for the jth feature: $\text{STD}(x_j) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_j^{(i)} - \text{ave}(x_j)\right)^2}$
  - Update the jth feature

$$x_j^{(i)} = \frac{x_j^{(i)} - \text{ave}(x_j)}{\text{STD}(x_j)}$$

Eg: If the average value of feature j is 70, and the standard deviation is 12 update all training example such that

$x_j^{(i)} = \frac{x_j^{(i)} - 70}{12}$ Now the average value of feature j is 0, and the standard deviation of feature j for the training examples is 1

Diabetes dataset
Mean Centering and Scaling to Unit Length example

"these data are first standardized to have zero mean and unit L2 norm before they are used in the examples."

Original Data

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

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

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

work area

$$\begin{bmatrix} \text{AGE} \\ 59 \\ 72 \\ 24 \\ 50 \\ 23 \\ 36 \end{bmatrix} - \begin{bmatrix} \text{mean} \\ 48.5 \\ 48.5 \\ 48.5 \\ 48.5 \\ 48.5 \\ 48.5 \end{bmatrix} = \begin{bmatrix} 10.5 \\ -0.5 \\ 23.5 \\ -24.5 \\ 1.5 \\ -25.5 \\ -12.5 \end{bmatrix}$$

l2 norm of the age feature is 275.3

$$\begin{bmatrix} 10.5/275.3 \\ -0.5/275.3 \\ 23.5/275.3 \\ -24.5/275.3 \\ 1.5/275.3 \\ -25.5/275.3 \\ -12.5/275.3 \end{bmatrix} = \begin{bmatrix} 0.038 \\ -0.002 \\ 0.085 \\ -0.089 \\ 0.0054 \\ -0.093 \\ -0.045 \end{bmatrix}$$

Learn more at https://en.wikipedia.org/wiki/Feature_scaling

I used the full dataset when I computed the mean and the l2 norm

How to handle categorical data?

Categorical data contains labels: {BMW, VW, Ford, GM} or {low, medium, high}, etc

Some categories contain a *natural ordering*: low, medium, high
Other categories *don't*: BMW, VW, Ford, GM

Many machine learning algorithms cannot work with categorical data.
How can we convert them to numerical data?

Two common approaches:

- Ordinal encoding
- One-hot encoding

If the number of categories is 2, then create a new variable x_i that takes takes two values. E.g. if we have a gender variable create

$$x_i^{(j)} = \begin{cases} 0 \text{ if jth person is female} \\ 1 \text{ if jth person is male} \end{cases}$$

ordinal encoding

Suppose x_i is a categorical variable

- One of a finite number of choices
- Example: "place" (gold silver bronze), etc

"**Ordinal encoding** should be used for *ordinal variables* (where order matters, like `cold`, `warm`, `hot`)"

- Assigns an integer to encode each value

If the $x_j \in \{$ cold, warm, hot$\}$

Cold	0
Warm	1
Cold	0
Hot	2
Warm	1

in the OrdinalEncoder class it is possible to set the categories argument

Warning! Sklearn OrdinalEncoder class assigns integers to categories based on alphabetic ordering
If you want the "right" encoding - assign the order "manually" by using the *categories* argument.

https://feature-engine.readthedocs.io/en/latest/encoding/OrdinalEncoder.html

https://datascience.stackexchange.com/questions/39317/difference-between-ordinalencoder-and-labelencoder

```
<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'
```

One Hot Coding

- Suppose x_i is a categorical variable
  - One of a finite number of choices
  - Example: male or female, or model of a car, location of a house, etc

```
[['<1H OCEAN']
 ['<1H OCEAN']
 ['NEAR OCEAN']
 ...
 ['INLAND']
 ['<1H OCEAN']
 ['NEAR BAY’]]
```

```
<1H OCEAN' 'INLAND' 'ISLAND' 'NEAR BAY' 'NEAR OCEAN'
[1., 0., 0., 0., 0.],
[1., 0., 0., 0., 0.],
[0., 0., 0., 0., 1.],
...,
[0., 1., 0., 0., 0.],
[1., 0., 0., 0., 0.],
[0., 0., 0., 1., 0.]
```

Dummy variable encoding

Dummy variable encoding is the preferred method for linear regression. This method avoids creating collinearity

If the $x_i \in \{$ Ford, BMW, GM, VW$\}$

Model	u_1	u_2	u_3
Ford	0	0	0
BMW	1	0	0
GM	0	1	0
VW	0	0	1

```
<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'
```

One Hot Coding

- Suppose x_i is a categorical variable
  - One of a finite number of choices
  - Example: male or female, or model of a car, location of a house, etc

```
[['
 ['
 ['
 ...
 ['INLAND']
 ['<1H OCEAN']
 ['NEAR BAY’]]
```

1 - if value is equal
0 - otherwise

```
'<1H OCEAN'  'INLAND'  'ISLAND'  'NEAR BAY'  'NEAR OCEAN'

[1., 0., 0., 0., 0.],
[1., 0., 0., 0., 0.],
[0., 0., 0., 0., 1.],
...,
[0., 1., 0., 0., 0.],
[1., 0., 0., 0., 0.],
[0., 0., 0., 1., 0.]
```

Dummy variable encoding

Dummy variable encoding is the preferred method for linear regression. This method avoids creating collinearity

If the $x_i \in \{$ Ford, BMW, GM, VW$\}$

Model	u_1	u_2	u_3
Ford	0	0	0
BMW	1	0	0
GM	0	1	0
VW	0	0	1

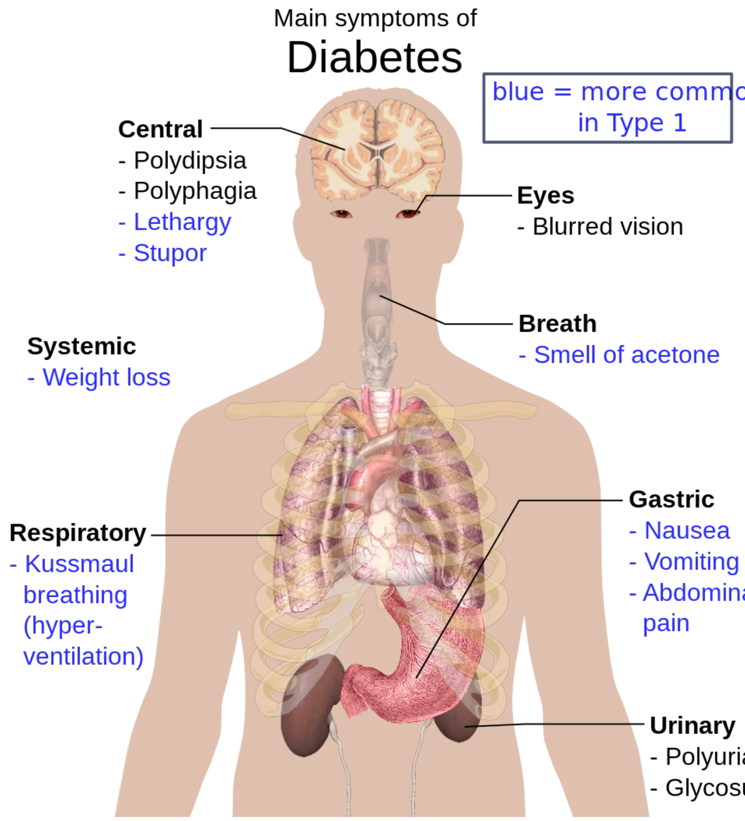
Outline

- ❑Motivating Example: Understanding glucose levels in diabetes patients
- ❑Multiple variable linear models
- ❑Least squares solutions
  - Gradient descent
  - Normal Equations
  - Feature scaling
- ❑Evaluating our hypothesis/Computing the solutions in python
- ❑Special case: Simple linear regression
- ❑ Rethinking the objective function
- ❑Extensions
- ❑Removing features

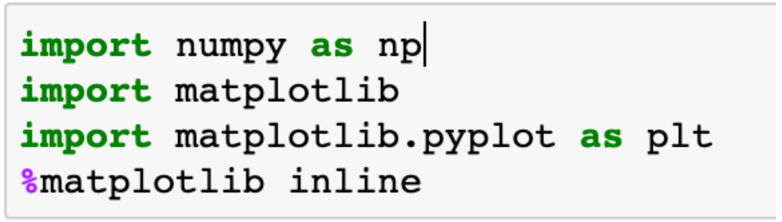
Linear Regression using Scikit-learn (Sklearn)

Built on NumPy

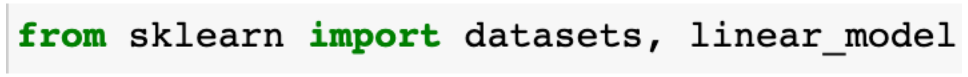
- Can we predict diabetes patients' condition a year after taking 10 baseline measurements?
- The 10 baseline measurements are: age, sex, body mass index. average blood pressure, and 6 blood serum measurements
- Steps for using Scikit-Learn to make predictions
  1. Import packages
  2. Get the data and preprocess the data
  3. Create a model, fit model with data
  4. Evaluate how well your model performs
  5. Use model to predict



Step 1) Import packages



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



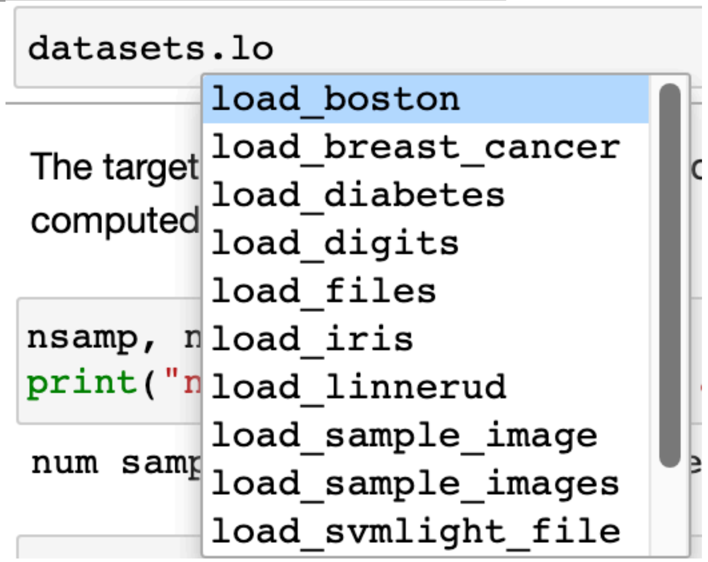
```
from sklearn import datasets, linear_model
```

Step 2) Loading the Data

```
# 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
```



```
datasets.lo
load_boston
load_breast_cancer
load_diabetes
load_digits
load_files
load_iris
load_linnerud
load_sample_image
load_sample_images
load_svmlight_file
```

The target
computed

```
nsamp, n
print("n
num samp
```

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

Step 2) dividing the dataset

We should randomly permute the data first!

```
ns_train = 300
ns_test = nsamp - ns_train
X_tr = X[:ns_train,:]       # Gets the first ns_train rows of X
y_tr = y[:ns_train]         # Gets the corresponding rows of y
```

- Divide data into two portions:
  - Training data: First 300 samples
  - Test data: Remaining 142 samples
- Train model on training data.
- Test model (i.e. measure RSS) on test data
- Reason for splitting data will be discussed in the next topic.

```
[[ 0.038  0.051  0.062  0.022 -0.044 -0.035 -0.043 -0.003  0.02   -0.018]
 [-0.002 -0.045 -0.051 -0.026 -0.008 -0.019  0.074 -0.039 -0.068 -0.092]
 [ 0.085  0.051  0.044 -0.006 -0.046 -0.034 -0.032 -0.003  0.003 -0.026]
 [-0.089 -0.045 -0.012 -0.037  0.012  0.025 -0.036  0.034  0.023 -0.009]
 [ 0.005 -0.045 -0.036  0.022  0.004  0.016  0.008 -0.003 -0.032 -0.047]
 [-0.093 -0.045 -0.041 -0.019 -0.069 -0.079  0.041 -0.076 -0.041 -0.096]
 [-0.045  0.051 -0.047 -0.016 -0.04  -0.025  0.001 -0.039 -0.063 -0.038]
 [ 0.064  0.051 -0.002  0.067  0.091  0.109  0.023  0.018 -0.036  0.003]
 [ 0.042  0.051  0.062 -0.04  -0.014  0.006 -0.029 -0.003 -0.015  0.011]
 [-0.071 -0.045  0.039 -0.033 -0.013 -0.035 -0.025 -0.003  0.068 -0.014]
  …
 ]
```

(300, 10)

```
[151.   75. 141. 206. 135.   97. 138.   63. 110. 310., …,   83]
```

(300,)

Step 3) Calling the sklearn method

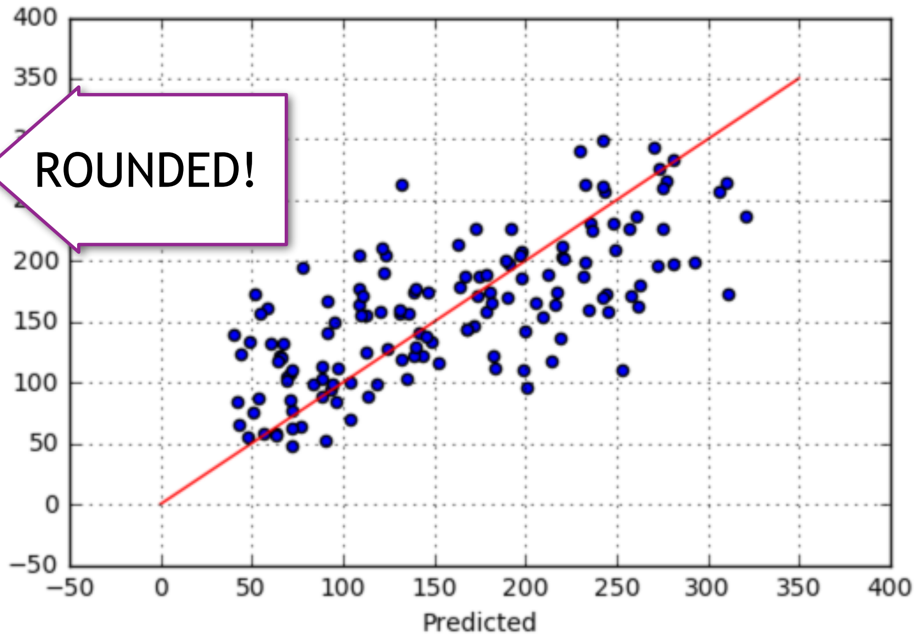
```
regr = linear_model.LinearRegression()
regr.fit(X_tr,y_tr)
```

```
print('The intercept w0 = ', regr.intercept_)
print('The coefficients w[1..d]=', regr.coef_)
```

```
The intercept w0 =  152.35
The coefficients w[1..d]= [-16.58 -254.67 560.99 278.92 -393.41
97.05 -19. 169.46 632.95 114.22]
```

ROUNDED!

- ❑Construct a linear regression object
- ❑Run it on the training data
- ❑Find the parameters of the model



$$\hat{y}^{(i)} = 152.35 - 16.58x_1^{(i)} - 254.67x_2^{(i)} + 560.99x_3^{(i)} + 278.92x_4^{(i)} - 393.41x_4^{(i)} + 97.05x_6^{(i)} - 19x_7^{(i)} + 169.46x_8^{(i)} + 632.95x_9^{(i)} + 114.22x_{10}^{(i)}$$

Learn more at https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
https://machine-learning-apps.github.io/hands-on-ml2/04_training_linear_models

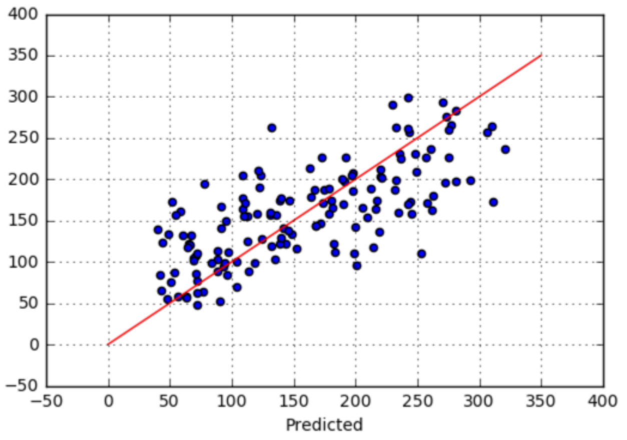
$$\hat{y}^{(i)} = 152.35 - 16.58x_1^{(i)} - 254.67x_2^{(i)} + 560.99x_3^{(i)} + 278.92x_4^{(i)} - 393.41x_4^{(i)} + 97.05x_6^{(i)} - 19x_7^{(i)} + 169.46x_8^{(i)} + 632.95x_9^{(i)} + 114.22x_{10}^{(i)}$$

Step 4 & 5) Evaluating the model and predicting

```
y_tr_pred = regr.predict(X_tr)
RSS = np.sum((y_tr_pred-y_tr)**2)
TSS = np.sum((y_tr - np.mean(y_tr))**2)
print("RSS = {0:f}".format(RSS))
print("Ein = {0:f}".format(RSS/ns_train))
print("RMSE = {0:f}".format(np.sqrt(RSS/ns_train)))
print("R^2 = {0:f}".format(1-RSS/TSS))
```

```
RSS = 876900.060150
Ein = 2923.000201
RMSE = 54.064778
R^2 = 0.514719
```

- ❑Predict values on the training data
- ❑Compute the R² score
- ❑Predict values on the test data

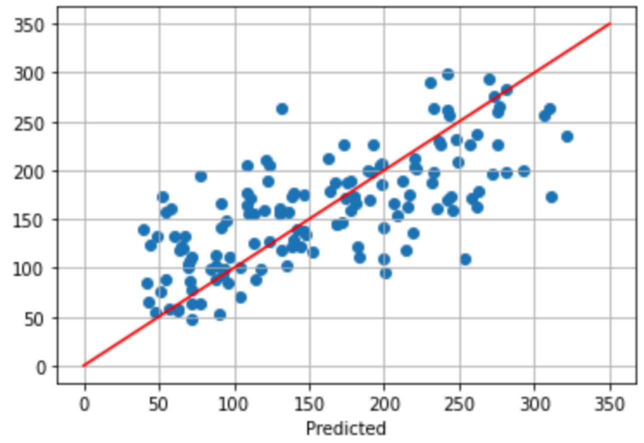


```
X_test = X[ns_train:,:]
y_test = y[ns_train:]
y_test_pred = regr.predict(X_test)

RSS = np.sum((y_test_pred-y_test)**2)

print("RSS = {0:f}".format(RSS))
print("Ein = {0:f}".format(RSS/ns_test))
print("RMSE = {0:f}".format(np.sqrt(RSS/ns_test)))
```

```
RSS = 396828.800059
Etest = 2794.569015
RMSE = 52.863683
```



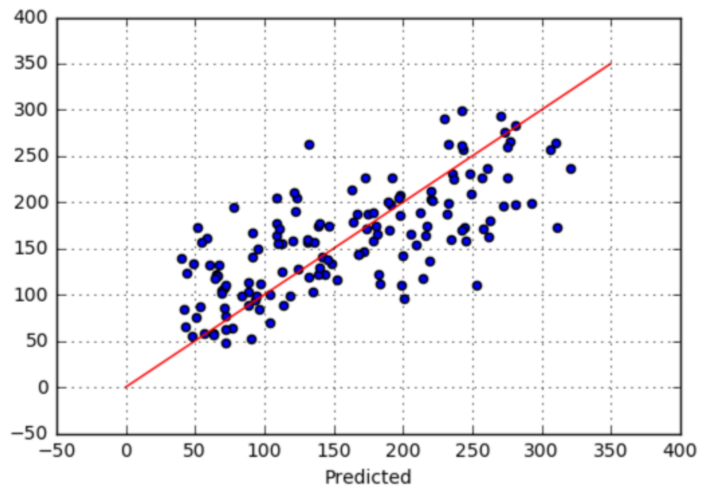
$$\hat{y}^{(i)} = 152.35 - 16.58x_1^{(i)} - 254.67x_2^{(i)} + 560.99x_3^{(i)} + 278.92x_4^{(i)} - 393.41x_4^{(i)} + 97.05x_6^{(i)} - 19x_7^{(i)} + 169.46x_8^{(i)} + 632.95x_9^{(i)} + 114.22x_{10}^{(i)}$$

Step 4) Evaluating the model

❑Compute the R² score

```
# We can also use the built in score function
regr.score(X_tr,y_tr)
```

```
R^2 = 0.514719
```



$$\hat{y}^{(i)} = 152.35 - 16.58x_1^{(i)} - 254.67x_2^{(i)} + 560.99x_3^{(i)} + 278.92x_4^{(i)} - 393.41x_4^{(i)} + 97.05x_6^{(i)} - 19x_7^{(i)} + 169.46x_8^{(i)} + 632.95x_9^{(i)} + 114.22x_{10}^{(i)}$$

Step 5) predicting

```
[[36    2           22      90   160     99.6      50      3    3.9512    82  ]]
```

Scale any new data using the mean and std of the training dataset

y_hat = rear.predict(x)

```
[[-0.046  0.051 -0.047  -1.600e-02  -0.040  -2.480e-02  -0.001  -0.039 -0.063  -0.038  ]]
```

```
71.81
```

Outline

❑Motivating Example: Understanding glucose levels in diabetes patients

❑Multiple variable linear models

❑Least squares solutions

- Gradient descent
- Normal Equations
- Feature scaling

❑Evaluating our hypothesis/Computing the solutions in python

❑Special case: Simple linear regression

❑ Rethinking the objective function

❑Extensions

❑Removing features

Comparison to Single Variable Models

- We could compute models for each variable separately:

$$y = a_1 + b_1 x_1$$
$$y = a_2 + b_2 x_2$$
$$\vdots$$

- But, doesn't provide a way to account for joint effects
- Example: Consider three linear models to predicting longevity:
  - A: Longevity vs. some factor in diet (e.g. amount of fiber consumed)
  - B: Longevity vs. exercise
  - C: Longevity vs. diet AND exercise
  - What does C tell you that A and B do not?

Simple Linear Regression for Diabetes Data

```
0  Rsq=0.035302
1  Rsq=0.001854
2  Rsq=0.343924
3  Rsq=0.194908
4  Rsq=0.044954
5  Rsq=0.030295
6  Rsq=0.155859
7  Rsq=0.185290
8  Rsq=0.320224
9  Rsq=0.146294
```

Best individual variable

- Try a fit of each variable individually
- Compute R_k^2 coefficient for each variable
- Use formula on previous slide
- "Best" individual variable is a poor fit
  - $R_k^2 \approx 0.34$

Simple vs. Multiple Regression

- Simple linear regression: One predictor (feature)
  - One feature
  - Linear model: $\hat{y}^{(i)} = w_0 + w_1 \mathbf{x}^{(i)}$
  - Can only account for one variable

- Multiple linear regression: Multiple predictors (features)
  - Many features (use a column vector to store features)
  - Linear model: $\hat{y}^{(i)} = w_0 + w_1 \mathbf{x}_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_d x_d^{(i)}$
  - Can account for multiple predictors
  - Turns into simple linear regression when $d = 1$

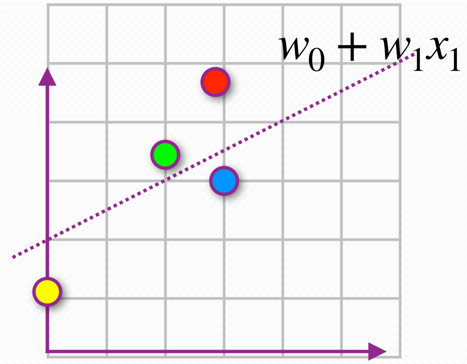
Outline

- ❑Motivating Example: Understanding glucose levels in diabetes patients
- ❑Multiple variable linear models
- ❑Least squares solutions
  - Gradient descent
  - Normal Equations
  - Feature scaling
- ❑Evaluating our hypothesis/Computing the solutions in python
- ❑Special case: Simple linear regression
- ❑ Rethinking the objective function
- ❑Extensions
- ❑Removing features

Another approach for determining which $\mathbf{w}$ is best

…it ends of reducing to the objective function we already chose

Probabilistic Interpretation



- **Errors due to** missing features, or noise in our measurements:

$$y^{(i)} = \mathbf{w}^T\mathbf{x}^{(1)} + \epsilon^{(i)}$$

- $p(\epsilon^{(1)}) \cdot p(\epsilon^{(2)}) \cdot p(\epsilon^{(3)}) \cdot p(\epsilon^{(4)})$

- What is $p(\epsilon^{(i)})$?

- One approach is to formally model $\epsilon^{(i)}$

$$(\mathbf{x}^{(1)}, y^{(1)}) = (2, 3 + \epsilon^{(1)})$$
$$(\mathbf{x}^{(2)}, y^{(2)}) = (3, 3.5 + \epsilon^{(2)})$$
$$(\mathbf{x}^{(3)}, y^{(3)}) = (0, 2 + \epsilon^{(3)})$$
$$(\mathbf{x}^{(4)}, y^{(4)}) = (3, 3.5 + \epsilon^{(4)})$$

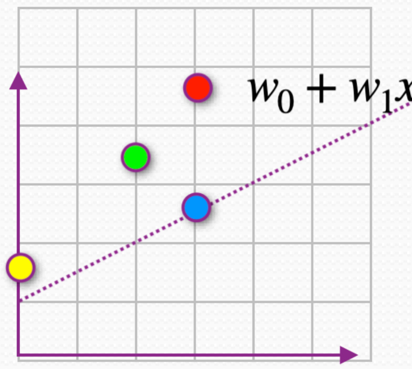
Also assume the noise is independently and identically distributed (IID)

$$\frac{1}{2^{(0.5)^2}} \quad \frac{1}{2^{(-1)^2}} \quad \frac{1}{2^{(-0.5)^2}} \quad \frac{1}{2^{(0.75)^2}} = 0.24$$

Intuition - it doesn't work since it is not a distribution....

$$\frac{1}{\sqrt{2\pi}}\frac{1}{e^{(0.5)^2/2}} \frac{1}{\sqrt{2\pi}}\frac{1}{e^{(1)^2/2}} \frac{1}{\sqrt{2\pi}}\frac{1}{e^{(-0.5)^2/2}} \frac{1}{\sqrt{2\pi}}\frac{1}{e^{(0.75)^2/2}}$$

Probabilistic Interpretation



- **Errors due to** missing features, or noise in our measurements:

$$y^{(i)} = \mathbf{w}^T\mathbf{x}^{(1)}+\epsilon^{(i)}$$

- $p(\epsilon^{(1)}) \cdot p(\epsilon^{(2)}) \cdot p(\epsilon^{(3)}) \cdot p(\epsilon^{(4)})$

- What is $p(\epsilon^{(i)})$?

- One approach is to formally model $\epsilon^{(i)}$

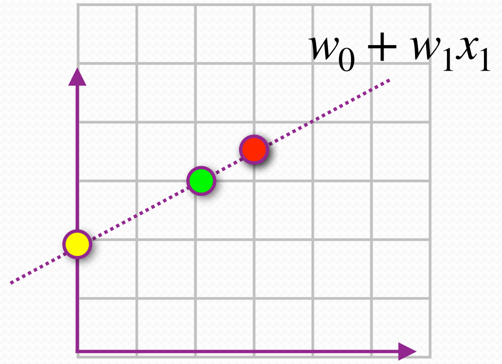
$$(\mathbf{x}^{(1)}, y^{(1)}) = (2, 2 + \epsilon^{(1)})$$
$$(\mathbf{x}^{(2)}, y^{(2)}) = (3, 2.5 + \epsilon^{(2)})$$
$$(\mathbf{x}^{(3)}, y^{(3)}) = (0, 1 + \epsilon^{(3)})$$
$$(\mathbf{x}^{(4)}, y^{(4)}) = (3, 2.5 + \epsilon^{(4)})$$

Also assume the noise is independently and identically distributed (IID)

$$\frac{1}{2^{(1.5)^2}}\ \frac{1}{2^{(0)^2}}\ \frac{1}{2^{(0.5)^2}}\ \frac{1}{2^{(1.75)^2}} = 0.03$$

$$\frac{1}{\sqrt{2\pi}}\ \frac{1}{e^{(1.5)^2/2}}\ \frac{1}{\sqrt{2\pi}}\ \frac{1}{e^{(0)^2/2}}\ \frac{1}{\sqrt{2\pi}}\ \frac{1}{e^{(0.5)^2/2}}\ \frac{1}{\sqrt{2\pi}}\ \frac{1}{e^{(1.75)^2}}$$

Probabilistic Interpretation



- **Errors due to** missing features, or noise in our measurements:

$$y^{(i)} = \mathbf{w}^T \mathbf{x}^{(1)} + \epsilon^{(i)}$$

- One choice is to assume $\epsilon^{(i)}$ is normally distributed (i.e. drawn as if it came from the univariate Gaussian with mean 0)

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{\left(\epsilon^{(i)}\right)^2}{2\sigma^2}\right)$$

- Now we can model the probability of seeing the value y given a fixed value of $\mathbf{x}$

Also assume the noise is independently and identically distributed (IID)

The measurements we have do not have infinite precision (i.e. $\mathbf{x} \in [\mathbf{x}_0 - \Delta/2, \mathbf{x}_0 + \Delta/2]$) . Thus we can use the probability density function compute the probability per unit area

$$\int_{\mathbf{x}_0-\Delta/2}^{\mathbf{x}_0+\Delta/2} f(\mathbf{x}_0; \theta)\, dx \approx f(\mathbf{x}_0; \theta)\Delta$$

Maximum Likelihood estimation (cont.)

$$\left(\epsilon^{(i)}\right)^2 = (y^{(i)} - \hat{y}^{(i)})^2$$
$$\left(y^{(i)} - (w_0 + w_1\mathbf{x}^{(i)})\right)^2$$

Which parameter w is best?

The one that is most likely is the one that maximizes $L(\mathbf{w}) = L(\mathbf{w}; X, \mathbf{y})$

$$L(\mathbf{w}) = \prod_{i=1}^{N} p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = \prod_{i=1}^{N} P(\epsilon^{(i)}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} exp \frac{-\left(y^{(i)} - (\mathbf{w}^T\mathbf{x}^{(i)})\right)^2}{2\sigma^2}$$

By performing a series of algebraic simplifications can be see to be the same as minimizing

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

https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/06/lecture-06.pdf
nttps://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/lec5.pdf

The next slide was not presented during the 11am class

From: http://cs229.stanford.edu/notes/cs229-notes1.pdf

Remember:

$\log(a \cdot b \cdot c) = \log a + \log b + \log c$

Thus

$\log \prod p(\epsilon^{(i)}) = \sum \log p(\epsilon^{(i)})$

What cost makes sense? (cont.)

Which parameter w is best?

The one that is most likely is the one that maximizes $L(\mathbf{w}) = L(\mathbf{w}; X, \mathbf{y})$

$$L(\mathbf{w}) = \prod_{i=1}^{N} p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = \prod_{i=1}^{N} P(\epsilon^{(i)}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} exp \frac{-\left(y^{(i)} - (w_0 + w_1\mathbf{x}^{(i)})\right)^2}{2\sigma^2}$$

Just algebraic manipulation

Note that maximizing this value is the same as maximizing $\ell(\mathbf{w}) = \log L(\mathbf{w})$

We are using ℓ for the log likelihood function

$$\ell(\mathbf{w}) = \log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} exp \frac{-\left(y^{(i)} - (w_0 + w_1\mathbf{x}^{(i)})\right)^2}{2\sigma^2}$$

$$= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} exp \frac{-\left(y^{(i)} - (w_0 + w_1\mathbf{x}^{(i)})\right)^2}{2\sigma^2}$$

$$= N \log \frac{1}{\sqrt{2\pi\sigma^2}} + \frac{1}{2\sigma^2} \sum_{i=1}^{N} -\left(y^{(i)} - (w_0 + w_1\mathbf{x}^{(i)})\right)^2$$

This … is the same as minimizing $\sum_{i=1}^{N} \left(y^{(i)} - (w_0 + w_1\mathbf{x}^{(i)})\right)^2$

Additional information can be found at: https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/06/lecture-06.pdf
https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/lec5.pdf

Outline

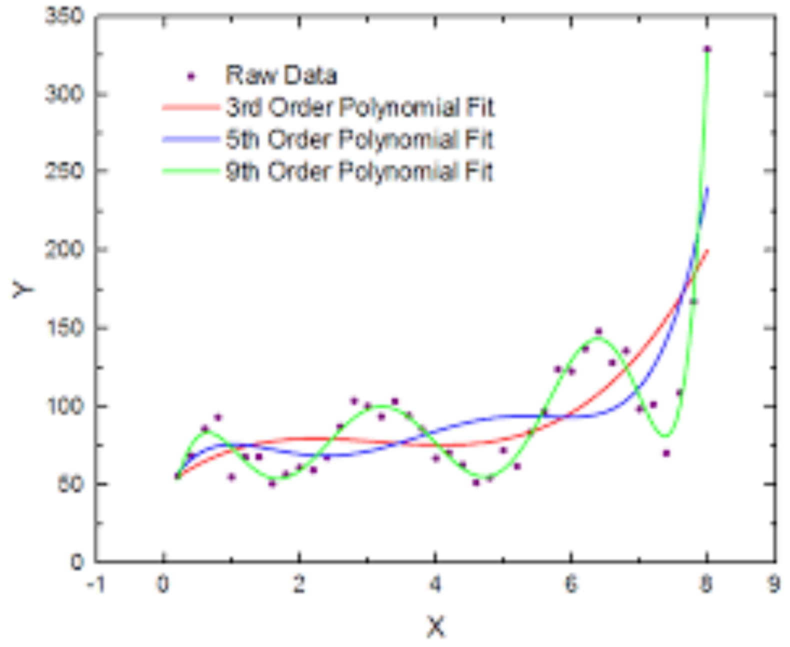
- ❑Motivating Example: Understanding glucose levels in diabetes patients
- ❑Multiple variable linear models
- ❑Least squares solutions
  - Gradient descent
  - Normal Equations
  - Feature scaling
- ❑Evaluating our hypothesis/Computing the solutions in python
- ❑Special case: Simple linear regression
- ❑ Rethinking the objective function
- ❑Extensions
- ❑Removing features

Polynomial Fitting

- Learn a polynomial model $\hat{y}^{(i)} = w_0 + w_1 \cdot x_1^{(i)} + w_2 \cdot \left(x_1^{(i)}\right)^2 + \cdots + w_d \cdot (x_1^{(i)})^d$
- Given data $(\mathbf{x}^{(i)}, y^{(i)}),\ i = 1, \dots, n$
- Form feature matrix and coefficient vector

$$X = \begin{bmatrix} 1 & \left(x_1^{(1)}\right)^1 & \dots & \left(x_1^{(1)}\right)^d \\ 1 & \left(x_1^{(2)}\right)^1 & \dots & \left(x_1^{(2)}\right)^d \\ \vdots & \vdots & & \vdots \\ 1 & \left(x_1^{(N)}\right)^1 & \dots & \left(x_1^{(N)}\right)^d \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

  - $p = d + 1$ transformed features from 1 original feature
- Will discuss model order selection in next topic
- Extensions to other nonlinear transforms



Outline

- ❑Motivating Example: Understanding glucose levels in diabetes patients
- ❑Multiple variable linear models
- ❑Least squares solutions
  - Gradient descent
  - Normal Equations
  - Feature scaling
- ❑Evaluating our hypothesis/Computing the solutions in python
- ❑Special case: Simple linear regression
- ❑Extensions
- ❑Removing features

The material on the following slides will not be asked on an exam /quiz

In the next topic, we will talk about how to create **new** features from our current set of features.

How do we “prune” away non useful features?

We will have to train more than one model to find what works best

Which features to select

Here we are using k instead d for the number of features

- Subset selection: Identify a subset of the k predictors we believe are associated with the response. Then the least squares solution can be fit on the reduced set of variables
- Try all 2^k-1 subsets of the features and select the best one
- We will use adjusted R^2 statistics to compare models when the number of features varies. (R^2 can increase when the number of features increases even if the features do not help predict the outcome!)

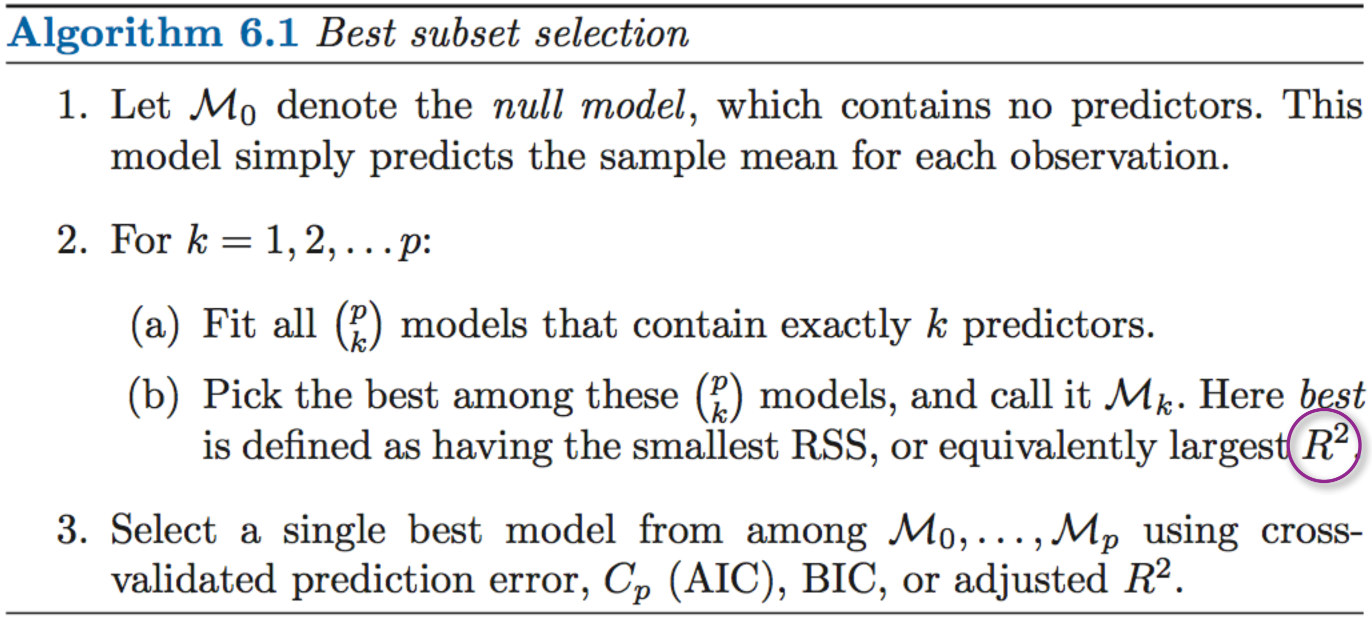
$$R^2_{adj} = 1 - \frac{RSS/(n-k-1)}{TSS/(n-1)}$$

- Where k equals the number of predictors in the model

Subset Selection

$$R^2_{adj} = 1 - \frac{RSS/(n-q-1)}{TSS/(n-1)}$$

- Subset selection: Identify a subset of the k predictors we believe are associated with the response. Then the least squares solution can be fit on the reduced set of variables



Algorithm 6.1 *Best subset selection*

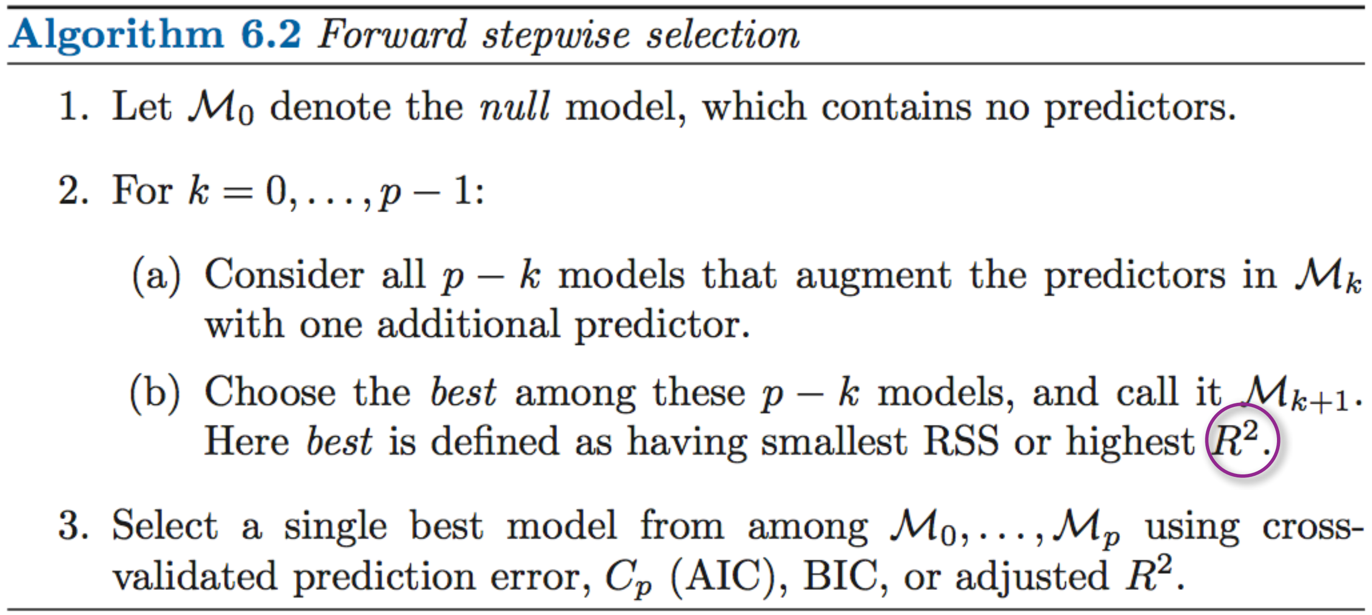
1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

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

Forward Selection

$$R_{adj}^2 = 1 - \frac{RSS/(n-q-1)}{TSS/(n-1)}$$

- Forward selections starts with no predictors predictors in the model
- Then repeatedly adds the most significant predictor



Algorithm 6.2 *Forward stepwise selection*

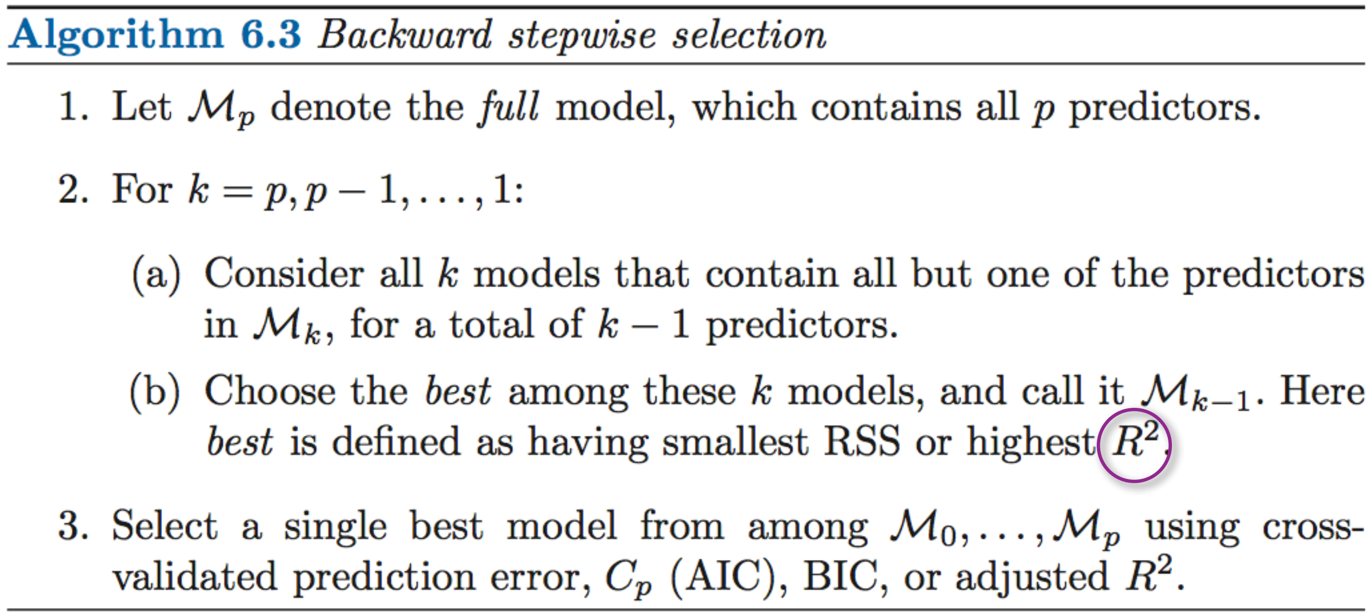
```
1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
    (a) Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$
        with one additional predictor.
    (b) Choose the *best* among these $p - k$ models, and call it $\mathcal{M}_{k+1}$.
        Here *best* is defined as having smallest RSS or highest $R^2$.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-
   validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.
```

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

Backward Selection

$$R^2_{adj} = 1 - \frac{RSS/(n-q-1)}{TSS/(n-1)}$$

- Backward elimination starts with all k predictors in the model
- Then repeatedly deletes the least significant predictor



Algorithm 6.3 *Backward stepwise selection*

```
1. Let $\mathcal{M}_p$ denote the *full* model, which contains all $p$ predictors.

2. For $k = p, p-1, \ldots, 1$:

    (a) Consider all $k$ models that contain all but one of the predictors
        in $\mathcal{M}_k$, for a total of $k-1$ predictors.
    (b) Choose the *best* among these $k$ models, and call it $\mathcal{M}_{k-1}$. Here
        *best* is defined as having smallest RSS or highest $R^2$.

3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-
   validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.
```

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

There are many ways to clean up the data

WE WILL NOT FOCUS ON MANY OF THE ISSUES

THIS IS AN INTRODUCTION TO THE TOPIC