CST 407 ML – Lecture 2

Ch 2, Regression techniques cont’d

# NumPy and Pandas

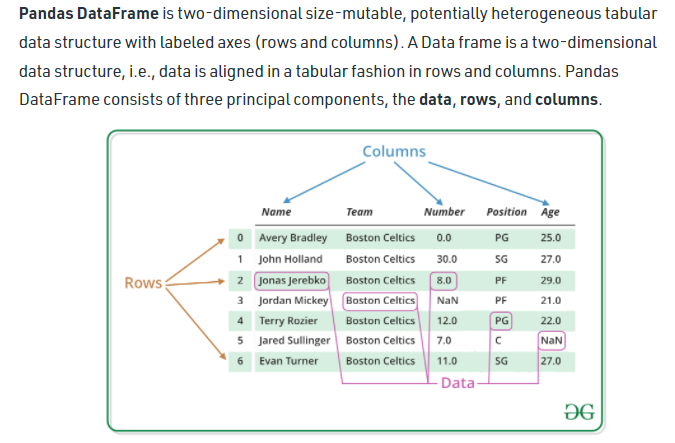
NumPy: “The fundamental package for scientific computing with Python”

Pandas: “…an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.”

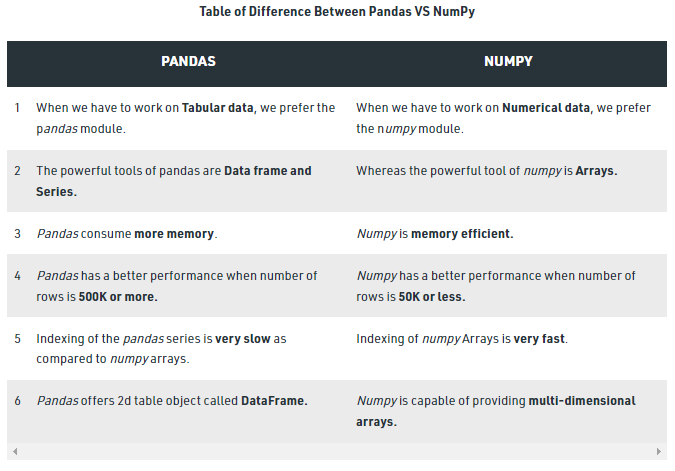
NumPy array…

“NumPy arrays are faster and more compact than Python lists. An array consumes less memory and is convenient to use. NumPy uses much less memory to store data and it provides a mechanism of specifying the data types. This allows the code to be optimized even further.”

Pandas DataFrame…



Why use NumPy or Pandas?



SciKit-Learn (sklearn) is built on both NumPy and Pandas, so we’re using them both!

But it’s good to know when to use one or the other :-)

References:

* <https://numpy.org/>
* <https://numpy.org/doc/stable/reference/arrays.html>
* <https://www.geeksforgeeks.org/difference-between-pandas-vs-numpy/>
* <https://pandas.pydata.org/docs/index.html>
* <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>
* <https://www.geeksforgeeks.org/python-pandas-dataframe/>

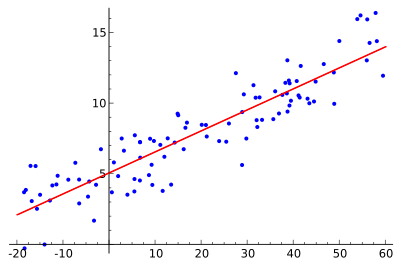
# Regression Algorithms

A few important regression algorithms:

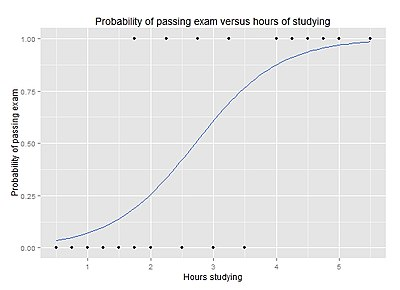
* Linear Regression
* Logistic Regression
* k-Nearest Neighbors

Thanks to wikipedia for the following diagrams…

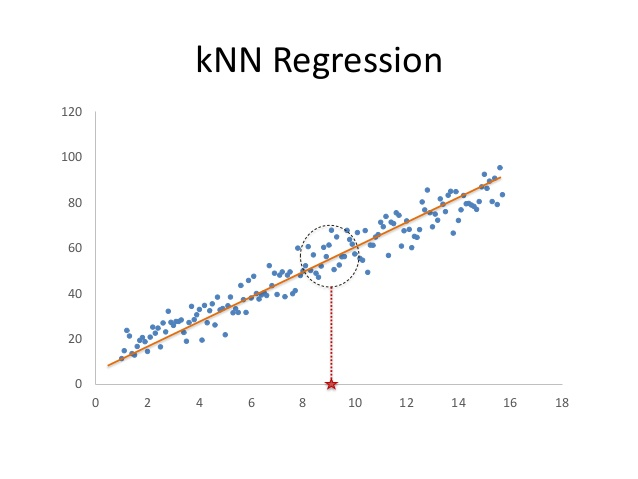
**Linear Regression** – models relationship between labels and features as a line y = mx+b



**Logistic Regression** – models probability of binary labels using a logistic function over features



**k-Nearest Neighbors** – estimates a label using weighted average labels of k nearest samples



<https://bookdown.org/f100441618/bookdown-regresion/ml-tools.html>

# Linear Regression Explained

From Ch 4 Training Models, section “Linear Regression”

Remember the example from Lecture 1…

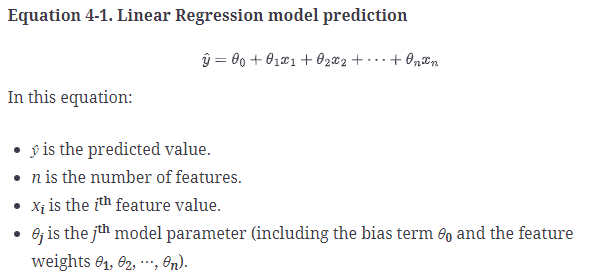


life\_satisfaction is the label we want to predict

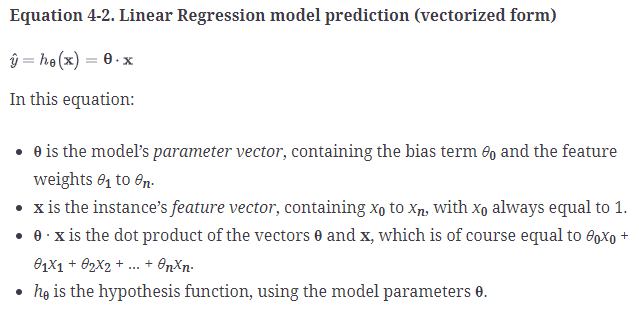
GDP\_per\_capita is a feature

θ0 and θ1 are the model’s parameters

In general…



Or even more generally, using vectors…



Training requires finding the ***best*** parameter values

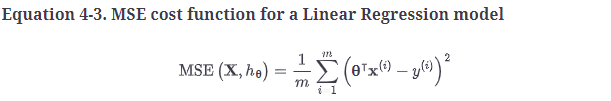
What do we mean by “best”?

We need a way to measure how good the fit is between the model and the data

We actually measure the opposite, error: how bad the fit is between the model and the data

The error can then be used as “cost” function for training

**MSE** – Means Squared Error

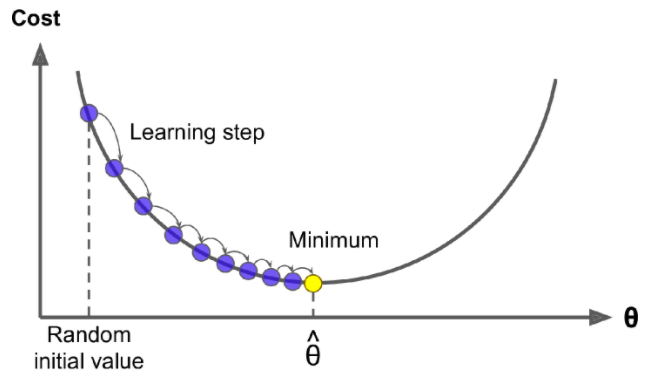


hθ is the hypothesis, or model, given current parameters θ

Think of this as the average distance between actual labels and predicted labels

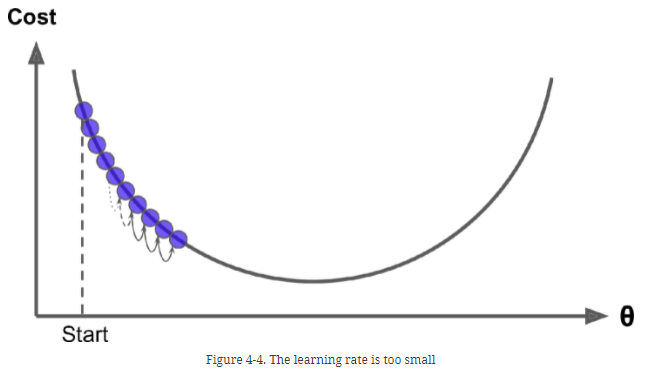
When we added up all those distances, we get the total error in the model

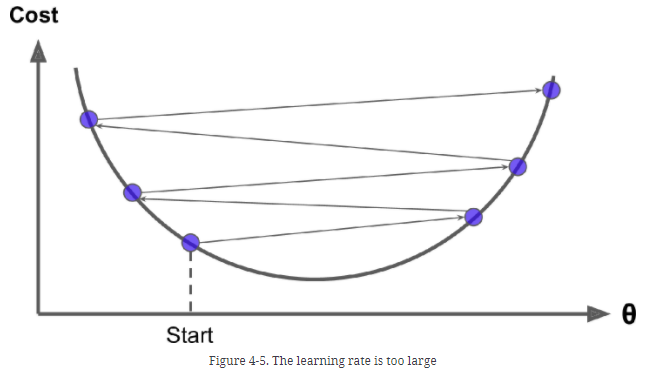
Now, to minimize the cost, we can use a technique like Gradient Descent



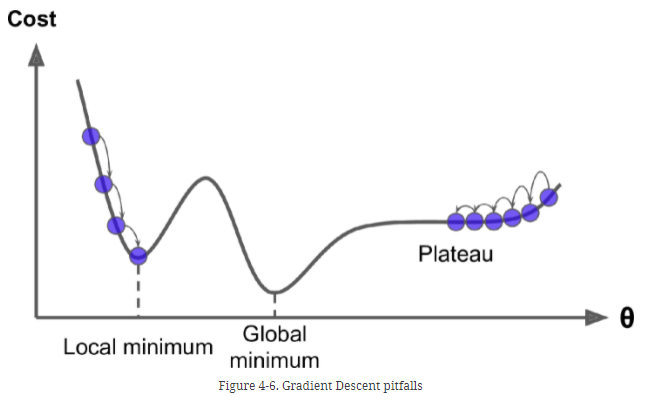
How does it decide how far to step? The “learning rate” hyper-parameter

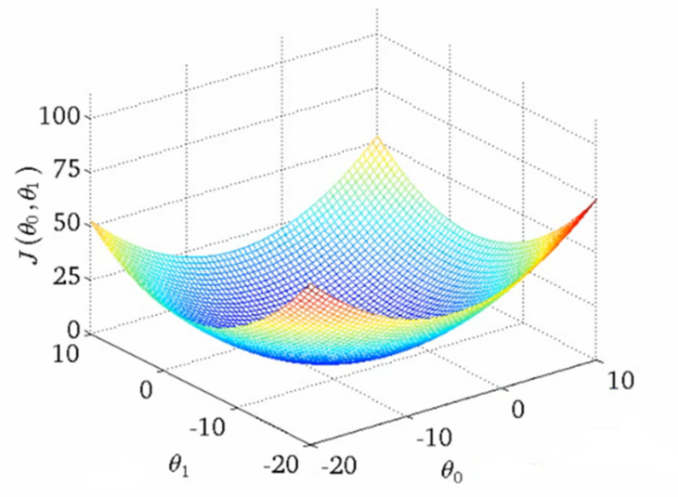
We have to choose the hyper-parameters for our learning algorithm carefully…





And we need to choose our cost function carefully…



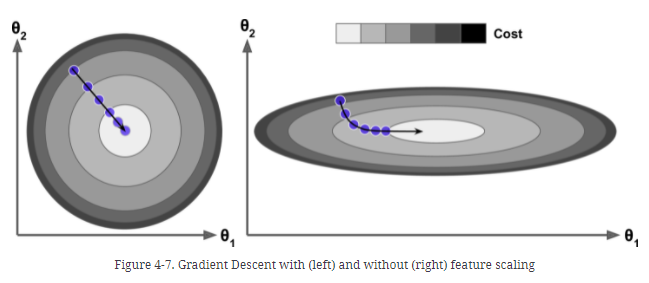
Fortunately, the MSE function is quadratic… so it’s a nice U-shape

Actually, in general, it’s a multi-dimensional bowl shape

When you are modeling based on multiple features, each feature may be on a completely different scale

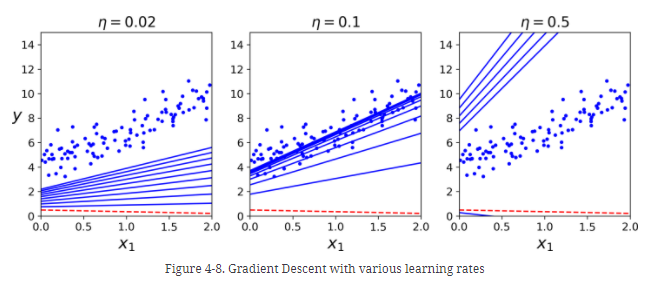
This can lead to longer distances to travel in one dimension and a longer run-time to train

So, we “regularize” the data: scale the data so it is normalized, e.g. all feature values are in [-1.0, 1.0]…



…and it’s faster to train!

Here’s how this looks for our model’s fit to the training data…



Of course, it’s really easy to do this with sklearn…

from sklearn.linear\_model import LinearRegression

lin\_reg = LinearRegression()

lin\_reg.fit(X, y)

y\_new = lin\_reg.predict(X\_new)

# Correlation

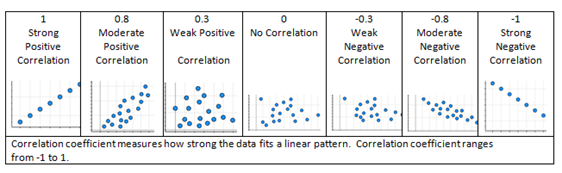
Which attributes/features should we select for our model? Why not all of them?

Training a model with O(n2), where n is the number of features, so…

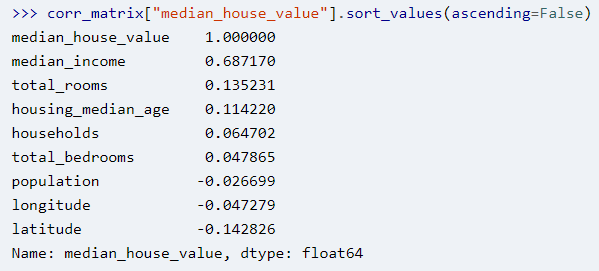
…choosing twice as many features, increases run-time by 4 times!

So, how do we know which ones to choose?

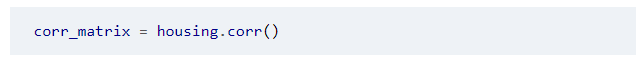
We can use the correlation between each parameter and the label



For example in Ch 2, we have housing data…



We can get this with the corr() method…



Visualization of some of these attributes vs. each other…

