# Useful links

<https://en.wikipedia.org/wiki/Gradient_descent>

<http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescen>

Regression and classification are actually same except predicting discrete or continuous outcomes.

Also don’t take regression always as linear regression. It can be nonlinear regression. Logistic regression is kind of nonlinear regression, even though it belongs to a generalized linear model.

Also supervised and unsupervised models are also same except for labels? How to reconcile that unlabel or label’s case are the special case of the each other.

About logistic regression.

y = 1/(1+exp(-z)), where z = theta\_transpose\*x. So it is a nonlinear regression, although it often said to be within a generalized linear model. We are actually fitting into a nonlinear function.

Figure out why choosing the sigmoid or logistic function is a natural one. Say notes of Ng.

# Questions

* Calculation of the step size. [1] See the simple calculation method in the Chinese article. [2] See the Wolfe conditions or Barzilai-Borwein method in the wiki link above. [3] In the DataCamp course about deep learning, there seems another way called ‘adam’. Need check the details later.
* Be careful what typical functions in data science are convex or not. Always try to find examples that SGD or SD converges to wrong points.
* See the questions and notes about derivation of Stanford notes in the end.
* First-order (Stochastic) gradient descent/ascent approaches are the most popular methods in data science? Can they completely replace the second-order approaches such as Newton method? Can they completely replace the closed-form solutions? From my understanding now, gradient descent/ascent approaches are very powerful as most of data science problems are minimizing/maximizing something by driving an initial cloud to a specific configuration.

# Easy Errors

* Note that for MLE we use gradient ascent, and for (log)loss/cost functions we use gradient descent. Not always GD, SGD, but also GA.
* Even for convex function, learning rate cannot be too large, otherwise it will not converge.
* For convex loss functions, the initial value for an iterative method does not matter. However, for neural networks, it is notoriously that they are not convex. So the choice of initial value is critical. Is this always the case?

# Key Points

* Be familiar in writing out the general GD formula and its derivation.
* With SGD and for some functions, it might never converge to a minimum or maximum but oscillate over the minimum region. For most cases, this is fine. If using BGD, it might be better. However, it might not be easy to do the batch case, particularly when the data set is huge.
* Notation conventions from Stanford notes: The input data X has n features, and parameters beta has n components. The total number of observations is m. However, due to the x0 = 1, the matrix is actually m x (n+1). In the notes, i = 1 to m for rows, and i = 0 to n for columns. This is different from that of Wikipedia in deriving linear regression.

# Applicable Scenarios

* Differentiable in a neighborhood of a point A.
* Convex functions (see wiki) such as x^2, exp(x), which has no more than one minimum in an open space.

# Exceptions

* If closed methods have exception on something, it should appear in the process of finding iterative. Always try to find that.
* The Rosenbrock function is a non-convex function. Converging to the minimum is very hard. See wiki for details.

# Calculations

* When approaching the minimum or maximum, SGD can sometimes oscillate as compared to batch GD. So normally we might use a smaller learning rate when approaching the minimum.

# Derivation of Stanford Notes

* cs229notes1 around page 3. …here n is the number of input variables (not counting x0). What this will imply in the future numerical calculations? Later the cost/loss function is thus from m=1 but not 0. Note the convention: In the h function summing from 0 to n, but in the cost function summing from 1 to m.
* Page 5. theta\_i := theta\_i-…. instead of theta\_(i+1) = theta\_i… In python, this is OK?

# Questions of Stanford Notes

Python Code Example (There are also matlab, R, examples etc.)

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

data = np.array([

[1, 6],

[2, 5],

[3, 7],

[4, 10]

])

m = len(data)

X = np.array([np.ones(m), data[:, 0]]).T

y = np.array(data[:, 1]).reshape(-1, 1)

betaHat = np.linalg.solve(X.T.dot(X), X.T.dot(y))

**print**(betaHat)

plt.figure(1)

xx = np.linspace(0, 5, 2)

yy = np.array(betaHat[0] + betaHat[1] \* xx)

plt.plot(xx, yy.T, color='b')

plt.scatter(data[:, 0], data[:, 1], color='r')

plt.show()

# Assumption of Linear Regression

* A linear regression model assumes: Linearity: µ{Y|X} = β0+ β1X
* Constant Variance: var{Y|X} = σ2
* Normality Dist. of Y’s at any X is normal
* Independence Given Xi’s, the Yi’s are independent. (**If correlated, or not independent, then this is related to the independent columns in the matrix**?)
* Check the ppt file for details:

C:\Users\Public\Fortran1C\courses\finance\Trading\cplusplus\UnEditedNotes\LinearRegression.pdf

# Meanings of regression and its applicable scenarios

* In statistics, regression means returning (re-gress-) to the mean. Imagine my mean-reversion method in stocks. So whenever work on regression-related method, always think about returning to mean. Mean is the most probable case in many situations, so returning to mean will normally give us the most probable case. In normal distribution, mean value has the highest probability. Also means can give either minimum or maximum scenario, which is the destination of iterative process such as KMeans, etc.
* Let’s use the stock case to understand when regression is applicable. In fact, when no the stock price is not strongly driven by the fundamental news, then the mean is the most probable case. Therefore, mean-reversion (similar to regression) is application when there is no fundamental big news. In other words, if the variables is really have a normal distribution, then mean-reversion or regression method is OK.