Machine Learning
9/19/2016
Ali Bramson

Linear Regression Recap
Buying 3 Corvettes- make an equation…

Price = year + Options + Condition + Mileage
So…need to scale to present fact that different variables interact with price in different way and get the units right: Price = A*Year + B*Options + C*Condition + D*Mileage

Linear Regression

- Trying to predict a variable in a continuum
- Have features or independent variables
-

Other applications for regressions: house prices, budget for a movie, effect of treatment, political inclination (1-100), number of likes, number of sales (for a product)

Feature Engineering

Y=bx+b0 (b is affine, b0 is bias)
Could also have y=b1x + b2x^2 + b0 and variable substitution says x=x1 and x^2 = x2
And leads to a really nice non-linear fit. Can fit parabolas by adding that extra complexity to fit more complex spaces…model was too simple when just y=bx+b0

By adding extra features, we move to higher dimensional spaces. Everything will be linear as long as don't start putting cos, sin in the b's.

Create a new exist of x^2 to make float in center of the space instead of just being in x-y plane. If that 3D space not enough, can add another and go to 4D space. Could do powers, or sqrt(x), etc…this is just a simple transformation example.

Optimization Problem
Optimize y=XB… linear algebra matrix approach…use transpose… J(B) is cost…if want to find best value you take its derivative and set equal to 0 and solve equation…that gives best beta out of the space. Linear algebra it is simple….goes to matrix form….set to 0…solve for beta using "the normal equations"

Matrix inversion one of most expensive computational things…. Having this inversion is going to haunt us. Needs time and memory.

Gradient descent- how much of error controlled by changes in the parameter youre looking for.
=

If you have a large amount of data: gradient descent rules. Matrix algebra is a pain. Small datasets matrix algebra is way faster. In practice, most implementations use gradient descent (google uses).

Last time discussed overfitting – where fit to noise…do this to cost function to add penalty (lambda or alpha depending on software) to say best fit of betas but don't make them too large… if make betas large then derivative will be large with respect to the variable you care about which means big jumps and then get squiggly lines that fit noise not underlying signal…tries to make next point close to point you were just at. This called "ridge regression"

Go through python notebooks and codes…

News of the Day
Fancy linear regression to take frame in video to predict next frame in video…predict next interaction in image is to shake hands. Just fancy linear regression where each data point is an image

Support Vector Machines-most used ML technique…best off-the-shelf ML technique. (might be responsible for ML renaissance)…

- Is relatively straight forward to use
- Has few parameters to optimize
- Works amazingly well, regardless the size of the data
- Is fast, compared with other ML techniques

Same problem as regression but framed in different way. SVM trying to find best line to separate data. Multiple solutions to same problem because over specified equations. Can choose any of these lines and will be good classifier for these points. What if a point is moved and now some of the lines don't work anymore. Try to find line that keeps points as far apart as possible. == Optimal Hyperplane (can be of any dimensions) try to find maximum margin
Find closest points to hyper plane….distances from these are support vectors…nice to see but not super informative. Don't often use the support vectors themselves. Max margin and optimal hyperplane are most important. Problem depends on distances - same problem as linear regression. Max margin is max distance… how define distances affects how distances look…. That's where kernels come into play

Kernel: like in linear regression, using only lines is impractical to solve classification problems. We use the concept of kernel (distance) and different definitions of kernels allow for different spaces. The most common ones: linear (logistic regression), polynomial (expanded powers/power of polynomial) and RBF (Gaussian kernel/"variance")

Different distances give different shapes…different spaces.

From scikit-learn.org site: different kernels give different spaces… RBF can give diff spaces not just lines. People use SVM with RBF with 0.1 variance…because those are default matlab values…. Can tell if people aren't thinking about how to pick best value…

Support Vector Machines (SVMs) form part of a family known as kernel methods (what everyone is using) – kernel methods are incredibly versatile- lots of research on what is a good kernel. Is a great can of worms. The panacea is to have self-defining kernels.