Summary of CS109

Friday, December 17, 2021 7:01 PM

Week 1 and Week 2.
Basic Introductory stuff. Setting up GIT, Python, Numpy, Pandas. In homework 0, we use python to simulate
The Monthy Hall game. We are able to show that switching doors helps.

Week 3 : Databases, SQL and more Pandas
Lab 2 introduces web scraping with requests and then parsing html with beautiful soup 4.
Lecture 4 - pandas, SQ
Lecture 5 - basic probability and statistics

Week 4: Probability and Regression.

Lab 3 - Probability and Distributions in Python

- Numpy is Introduced.
- The idea of probability is defined. How expected value only applies when the sample size is infinite.
- How probability is simulated in python. Using 1D array of floats between 0 and 1. This creates an **Empirical Probability distribution.** It is the empirical distribution associated with the empirical measure of the sample. At n tends to infinity it approaches the population distribution. EPD is created through a series of steps:
  a. Obtain Empirical Data.
  b. Convert Empirical Data to probabilities using a certain method.
  c. Use arrays of floats, and plot.
- **PDF** tells us the probability of where a **continuous/discrete random variable** will be in set of possible values that random variable can be (the sample space). **(e.g. Typical Gaussian or Poisson PDF)**
- **PMF** tells us the probability that a **discrete random variable** will be exactly equal to some value. **(i.e. Binomial Distribution)**
- **CDF** function tells us the probability that a **random discrete or continuous variable X** will take a value less than or equal to X. **(Sigmoid Curve, Y-axis goes from 0 to 1)**
- We learnt about np.random. You can choose from array, random float, integer, Gaussian.
- Scipy.stats.distrib can give us **theoretical distributions.**

Lab 3 - Sampling and Distributions

Expectation Value

- The **expectation value** is the weighted sum of the quantity where the weights are probabilities from the distribution. Formula
- For discrete variable E(x) not equal to mean. Mean of unfair die is 3.5, E(x) is 6.
- For a continuous variable. This is an integral f(x) x. Note: mean and expected value are defined as the same thing for continuous variables.

Samples and Populations

- **Law of Large Numbers:** Sample mean converges to Population mean. Provided that x is IID (**Independent and Identically Distributed**). Identically distributed x are part of the same distribution. Independent means p(x) is constant.
- The mean and variance of a sample are different from that of the population. i.e.
  - $S_n \sim N(\mu, \frac{\sigma^2}{n}) \, as \, n \to \infty.$

  Distribution of Sample means ~ (Population mean, Standard Error^2)
- Where **Standard Error**. Sd(x)/sqrt(N of x) This means that your sample mean has an sd of SE.

Sd(x) is the population Standard Deviation.

- **Population Mean and Population Standard Deviation cannot be determined** but, under **Lyapunov** conditions, can be estimated:
- **Population Mean = Mean of the Sample Means**
- **Population Standard deviation = Average Sample Sd.**
- Note:
    - Distribution of means will always be a gaussian. (**Central Limit Theorem**)
    - The Gaussian's mean will be equal to the population mean for large N. (**Law of Large Numbers)**
- This is why the cancer rate in some small populations is absurdly high or low. (Article: MostDangerousEquation) 10^7 samples were needed to get sd 1 from the true mean.
- Applications:
    a. **We can know if our sample mean is representative of the population mean**.
    b. **We can know how big of a sample we need.**
    c. **We can do Hypothesis testing, and Bias testing (between two samples)**

  E.g. We get a distribution of Sample means for the weight of gym rats. They centre around population mean 70kg, and have SE 3kg. I am given a new sample mean that is 7 standard deviations away from my distribution. Now I know this group of people are not from the same population. (Hypothesis testing.)

  Note: The expected value of the sample variance is slightly than the actual variance. This is because you tend to pick values that are more probable from the population. This effect disappears for large sample sizes. This is what happens with **Genetic Drift.**

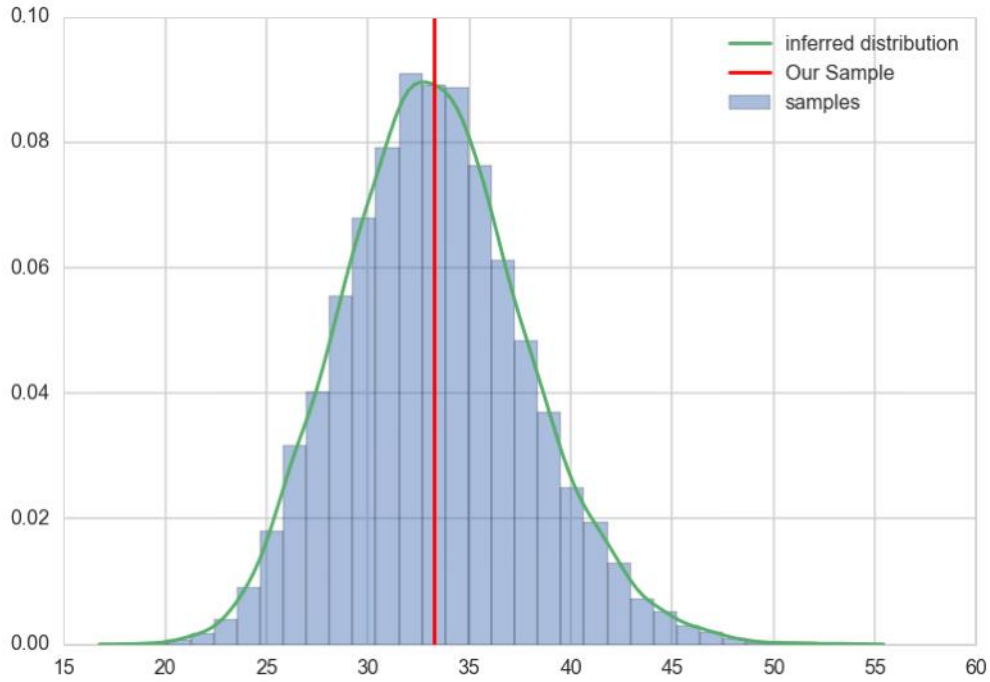
Lab 3 - Frequentism, Samples and the Bootstrap

- There's a civil war between frequentists and Bayesian statisticians
- Basically, Frequentists think of the data we have in hand as a sample from a defined population. Bayesians think of a set of data as being the population.
- We learn about Poisson Distribution. Poisson is used to express the probability of a number N of events occurring in a given time. I.e. number childbirths, bus, shooting stars. Assumes the events are random, no clustering, and independent. Hence it is a discrete probability distribution.
- The pdf of a Poisson is exponential. Goes to zero exponentially.
- Note: Sample distribution is always given in a histogram.

BootStrapping

Bootstrapping gives you a notion of the various values the population lambda could have taken.

- **Parametric Bootstrapping,**
    1. we first determine the sample statistics (i.e. lambda for Poisson, mean and sd for Gaussian).
    2. Then, use them to generate many sets of data points.
    3. Calculate lambda for each data set.
    4. Plot the lambda distribution histogram.



0

- **Bootstrapping with Replacement**
    1. Use np.choice to pick data points from your data set.
    2. Same steps as before.

Lecture 6: Story Telling and Effective Communication
Good insights on how to tell a story with data. Infer, model, use an algorithim and draw conclusions (and check!).

Start with two fundamental questions:

1. Whats the goal? think first of that rather than going first to all the many ways you can slice and dice data.
2. Who cares? Know your audience and tell them a story. Have a clear sense of direction and logic.

Read some howto's on scientific writing
have some memorable examples or small stories

Tell a story:

- know your audience and why/what they care about this data - what do they want?
- Don't make them think - clearly tell them what you want them to know in a way they can follow. highlight key facts and insights.
- unexpectedness - show something the audience didn't expect. I liked the story which highlighted that bigfoot sightings are dropping sharply
- What tools can we give the audience? For example, a web app for them to further explore the data, or a takeaway presentation with key points.
- be careful of your point of view and don't distort the data, but depending on the audience you can frame your story - for example presenting war deaths in red rather than a regular plot color.
- important to put the message up front - what does my graph show? Show it in stages if a lot of data, highlighting what to look at. Design matters.

More resources:
The Functional Art

Lecture 7: Bias and Regression

Different types of Bias:

- Selection Bias - i.e. Wald and Bullet Holes. Surviving planes are planes that survived.

Average longevity of someone in a profession.
    - Publication Bias - need p<0.5 to publish a study.
    - Non-response bias - people that responded are different from the population
    - Length Bias - Most people in prison have life sentences.

  Think about what is not being observed, not just the observed.
- **Fisher Weighting** - It is an optimal way of combining data sets to estimate a parameter like mean etc. The weighting for these 'estimators' should be 1/Var(estimator) Note the estimators are unbiased and independent.
- **Nate Silver Weighting for Polls -** This dude went very deep into how to weight different data sets.
- Bias can be estimated for past Data that have results.
- **The Bonferroni Correction** - It is a method for dealing with accidental correlations. For 100 predictors, do p_value/100. (Called Multiple Testing)
- **Regression towards the mean** - example: **Sophomore Slump. The idea that everything is a combination of luck and skill. Skill determines the expected value, luck the variance.**
- This is why people think praise is bad and punishment is good.

Linear regression

- AKA **Ordinary Least Squares**, but OLS is not the model, it's the error.
- It's called linear because the y=XB+e. (Matrix notation)
- There are N predictors and K variables. i.e. you are trying to predict someone's height or income based on where do they live, how old they are, etc.
- In Big Data problems, k>N. Normally, N>k. i.e. Now, 5 customers, with 10 000 columns of information about them.
- Variance R^2 measures goodness of fit, but doesn't mean model is good. It means, how much of the variance of y is accounted for by X.
- Best way to check a model is prediction.
- Note: OLS can be used with predictor squared etc.
- You can plot the residuals to know if there is a non linear relationship.

Week 5: Scikit learn & regression

Lab 4 - Regression in Python

- R2 can be obtained for each predictor.
- Linear regression assumes normal distribution of residuals.
- There are many ways to fit a linear regression model, most common is the Least Squares Method. This is called an **Error function**
- **Polynomial Regression** - Overfitting explained. This is caused by noise.
- **Bias Error is the error created from approximation**. i.e. using a line to predict non-linear relationships.
- **Variance is the error associated with overfitting.**
- By using **Bootstrapping**, you can visulise the effect of overfitting.
- We can split the data set into training and testing set. itrain,itest = train_test_split()
- Very interesting Lab. Do it later.
- This lab has very foundational concepts for machine learning. How to visualise error in prediction, and minimise the test set error based on that.
- **Cross Validation Explained (See Mindmap)**

Lecture 8: More Regression

- **collinerarity** - when some variables are highly correlated with each other - this is bad
- Logistic Regression
- **Odds Ratio**: ratio of two different people's odds of an outcome.
- **Confounding Factors** - i.e is one group pre-disposed to our outcome for some reason?
- **Curse of dimensionality** - in high d settings, vast majority of data is near boundaries, not center. But, high dim can also be a blessing.
- dealing with high dimensionality:
    - **Lasso regression**

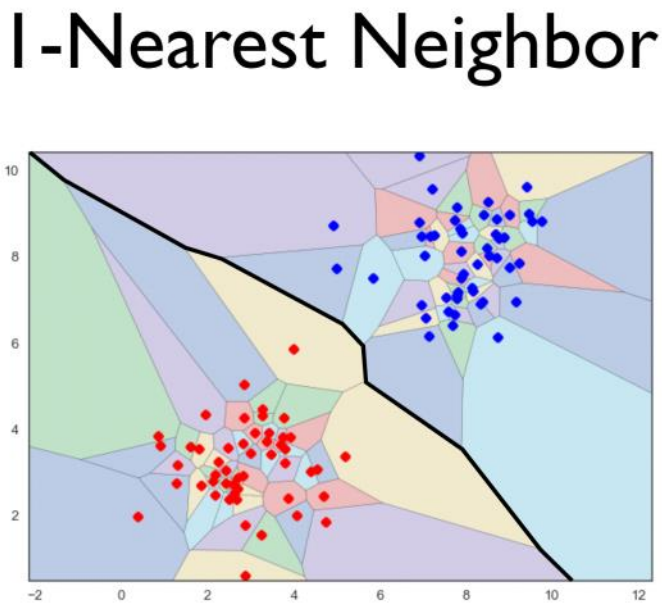
- Stein's Paradox wikipedia, good article
- **LASSO** and **Ridge** help with high D data by reducing features YouTube
    - **Lasso** is called L1 regularization, reduces number of features
    - **Ridge** is called L2 regularization, doesn't necessarily reduce features but reduces the impace of features on the model by reducing coefficient value
- **LASSO** minimises (sum of squares and lambda times magnitude of m coefficients). This results in some m becoming zero. Hence, handling **collinearity**.
- **Ridge** minimises (sum of sqares and lambda times slope of regression line). This reduces overfitting.
- Elasticnet does both L1 and L2 regularization. It also gets rid of **multicollinearity**.

Lecture 9: Machine Learning
Classification kNN Cross Validation Dimensionality Reduction PCA MDS (Hanspeter Pfister)

- **Examples of Classification:**
    - Classification of Email into Spam or not Spam.(Using Naïve-Bayes)
    - Employers classifying good and bad candidates based on Resume.
    - Determining whether a text is in Chinese or English etc.
    - Classifying pictures into cat/dog/chair.
- **Hanspeter tells us about his research**: Picture Classification - This is hard because of occlusion, 3D object into 2 D objects, Shadows, lighting, deformation, different types of chair. This is called computer vision. What led to the great breakthrough in Computer Vision? They had people assign labels to training data. Check out: The Unreasonable Effectiveness of Data
- Basically, Classification involves plotting features against each other. The data points have a colour representing their labels.

K Nearest Neighbours - Typically K = nearest points

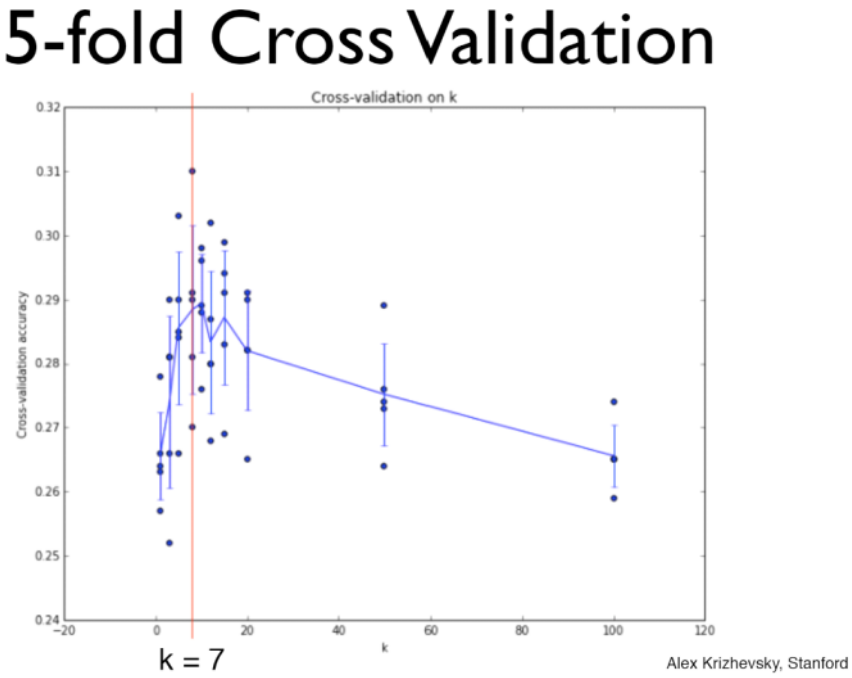
- There are many distance functions
- For a 1 Nearest Neighbour Classifier **INN**, a **Voronoi** diagram is used to draw a **decision boundary.**



- Outliers are converted to islands.
- **What is the Training complexity?** Order One. Because all you have to do add it when training.
- **What is Testing Complexity?** O(mxn)
- No work during training, all work during testing. Normally we want the opposite.
- We also say that **Variance is high**, since for every new training point, you get a new boundary line (i.e. the decision boundary is very rough).

- Why use KNN instead of INN?
    - Removes Islands
    - **Bias is low**, because on average all of them do quite well.

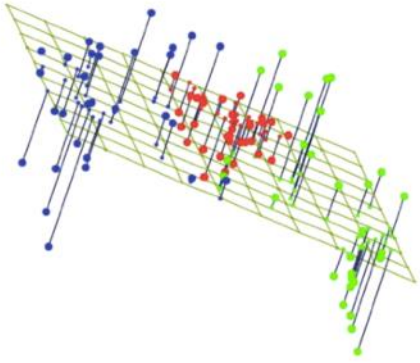
    - **Increasing K reduces variance** (makes line smoother) but can introduce bias(approximation occurs)
- Different ways to measure distance (Euclidean distance,) and do Majority voting.
- I made a kNN algo implementation to understand it better.
- **Do not use test data to estimate hyperparameters. (otherwise cannot generalise model)**
- **5 Fold Cross Validation Graph to find best k.**



- Number of folds depends on how much data you have.
- Interesting KNN project from Stanford (CS 231n).
- Turning pictures in to data points: (This is a **feature**, pixel by pixel difference)
    1. Unroll pixels into 1D array. Then substract from training.
    2. This is **L1 distance.** (**Manhattan distance**)
    3. Or use **Euclid distance**. Called **L2 distance.**
- Nowadays, we use SIFT features. (Scale, Rotation, Scale Invariant)

Dimensionality Reduction

- Too many features result in sparse models. (**Curse of Dimensionality)**
- The idea is to project high dimensional data into a lower dimension by preserving distances. This projection is linear, and uses hyperplanes.



- **PCA** does this for supervised learning methods. (**Principal Component analysis**)
- **Great Question** - Given that the dimensions have been reduced. What is the meaning of X and Y in the scatter plot? - No meaning, at all.
- **PCA handles Collinearity**
- **Data must be Standard Scaled before using PCA. (Due to rotations, and features being added.)**
- It is implemented using Single Value Decomposition **(SVD)**.
- By using a **Scree Plot** - Eigen values against Eigen Vectors, you will know at which point the

variance starts to rise, as your Dimensionality goes down. This is your **Intrinsic Dimensionality.**
- Example: **Eigen faces and handwritings.** Using raw pixels, PCA figured out that faces can be summarised into a set of faces.
- **MultiDimensional Scaling (MDS)** - similar to PCA. Instead of preserving orthogonal distances, it preserves distances between data points. It can figure out a 2d map. Based on distances between things.

Week 6: SVM, trees and forests

Lab 5: Classification
- Very important notebook.
- Learn and apply it when you have the chance.

Lecture 10: SVM, Evaluation (video, slides)

- but if we know the decision boundary (the **separating hyperplane**) we don't need all the data points
  - w: weight vector defines the orientation of the hyperplane, and bias b.
  - so a new point x is classified by w(transpose)*x + b
  - this is the mathematical model of a neuron, invented 1957 by Rosenblatt
- Support Vector Machines (SVM) add a new dimension to help seperate classes. (**separating hyperplane**)
- They use **maximum margin classification**.
- SVM is called svm becuase of the support vectors defining the max margin lines for the classification boundary.
- Large data is good for training svm as the points on the boundary are rare and svm cares about establishing the boundary
- since outliers can change the svm boundaries, there is a concept of **slack variables** - it allows the SVM to missclassify outliers to make a neat decision boundary. **sklearn uses the parameter C to define the slack**. the lower the number the more the slack.
- kernel tricks for svm - go to aribitarily many dimensions with little computational cost. need to think about what kernel to use. **Read** What are kernels in machine learning and SVM and why do we need them?.
- todo: tale sklearns 'faces' dataset and use svm to predict
- svm tips:
  - normalize data to 0,1 or -1,1 interval. (check whether the library already normalizes)
  - RBF kernel is a good default
  - **Read Chich-Wei Hsu practical guide to SVM**
  - tune paramaters - **which kernel, what parameters for it and what C?**
- **ROC curve**: plot true positive rate vs false positive rate
  - true +ve is tp/(tp+fn)
  - false +ve is fp/(fp+tn)
  - one useful summary stat is area under the curve
- **Precision Recall Curve** sklearn, quora
  - Precision: tp/(tp+fp) - how much are we getting right, or the probability a a random +ve sample is actually +ve (since some +ve samples are false positives).
  - Recall tp/(tp+fn) - same as in the ROC, how much of the data are we finding. for a random +ve sameple, the probability that it's making a correct prediction. (consider the false negatives.)
  - ideally we want both to be one.
- good way to compare classifiers
- How do you classify multple classes with a SVM, e.g 10 types of fruit
  - **One vs all** - pick one class, and train it against all the other classes one by one. So you train n classifers for n classes.
  - **One vs One** - Train n(n-1)/2 classifiers, take majority vote
  - use a confusion matrix of predicted label vs true labels to see classification results

Lecture 11: Decision Trees and Random Forests

- **Decision Tree** - fast training and prediction, easy to understand and interpret. DT basically paritions a feature space into cells using a series of decision rules on one feature at a time
- which feature to query and what thresholds to use?
- node purity: do splits so cells are pure, i.e have only one class in them
    - Gini Impurity gives us the expected error of predicted a class from a random sample
    - Gini Index
    - Node purity gain: compare gini impurity of parent vs child nodes. Lets us see whether a split has improved classification better than a simple missclassification number.
- Optimal trees - diff ways to get there
- **Tree pruning** - easy to overfit, so first we make the tree, then go backwards and remove 'bad' decisions, or merge cells etc.
- DT disadvatages: sensitive to small changes in data, overfit, only axis aligned splits
- **Bootstrap** is one way to do this. It's a resampling method from statistics, useful to get **error bags** on estimates.
    - Bootstrap lets us generate more sample sets from one dataset, each one slightly different.
    - Take N data points and draw N times with replacement, then get an estimate from each bootstrapped samples.
    - **bagging: bootstrap aggregrating**, where you learn a classifer from each bootstrap sample and average the . (normally uses just one type of classifier)
        - See bootstrap example notebook
        - bagged DT's perform better than one a single tree
        - not useful for linear models
    - Bias-Variance trade off: train models with a high variance, then the average might get close
- **Random Forest builds on bagging**, builds a tree from each bootstrap sample with node splits selected from random feature subsets. See 1, or rather 2