DATA SCIENCE

from
FranR Kane

17/05/2020

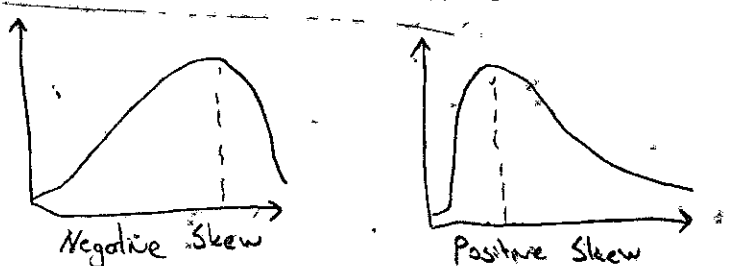
Mean → Average
Median → 0011[2]2234 (the value in the middle) (better if there is a lot of outliners)
Mode → 1 (Most common).

If you wanna show the average person salary in somewhere, the median is better. Because the extremely rich or w/ salary people could highly affect the mean. For US, mean → 72.000 $, median → 52.000

* Numerical Data (Discrete Data, Continous Data)
* Categorical Data
* Ordinal Data (Mix, for example rating stars)

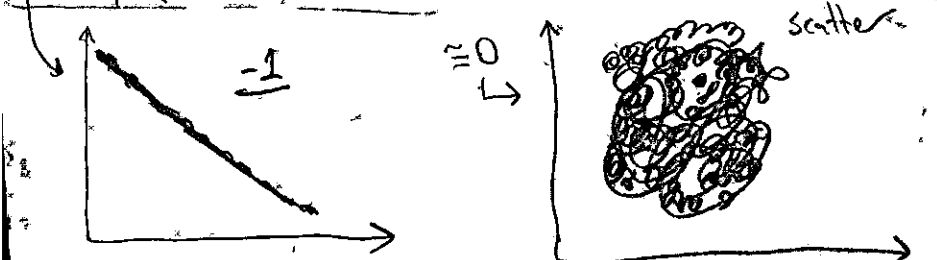
→ Variance measureces how "spread-out" the data is.
It is simply the average of the squared differences from the me
→ Standart deviation is root of variance
→ For sample variance, you just divide the squared variances by N-1 instead of N.

Probability Mass Function → Discrete
Probability Density Function → Continuous



Correlation

-1 means a perfect inverse correlation
0 no correlation
1 perfect correlation



Conditional Prob

$P(A,B)$ = Probability of A and B both occuring
$P(B|A)$ = Probability of B given that A has occured

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

Bayes theorem

We have a drug test that can accurately identify users of a drug 99% of the time, and accurately has a negative result for 99% of non-users. But only 0.3% of the population uses this drug.

⇒ A = User of the drug B = tested positively for the drug

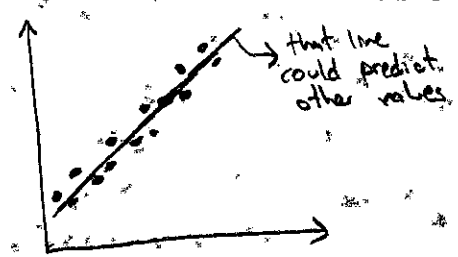
⇒ P(B) = prob. of testing positive if you do use, plus the probability of testing positive if you don't → $0.99 \times 0.003 + 0.01 \times 0.997 = 0.013$

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)} = \frac{0.003 * 0.99}{0.013} = \%22.8$$

So the odds of someone being an actual user of the drug given that they tested positive is only %22.8 !

Linear Regression (from Numerical Methods)

↳ Gradient Descent (can make sense when dealing with 3D data)



* How do we measure how well our line fits our data?
↳ with R-squared → Ranges from 0 to 1
0 is terrible → none of the variance is captured
1 is perfect → all " " " "

Multivariate regression

More than one variable influences the one you're interested in.
Ex= predicting a price for a car based on its attributes
↳ we just end up with coefficients for each factor.

*price = $\alpha + \beta_1$ mileage + β_2 age + β_3 doors

Multiple levels (advanced topic tbh)

The concept is some effects happen at various levels (hierarch
You must identify factors that affect the outcome at each leve
Ex= SAT scores based on genetics of individual children, the home environment, crime rate of the neighborhood, the quality of teachers, the funding of their school district---

---

Unsupervised learning

The model is not given any "answers" to learn from.
It can be a powerful tool for discovering classifications that you did not even know there. (latent variables)
↳ = ortho
Ex= Cluster movies based on their properties (origin country, year...)
↳ = homelore
Perhaps our current concepts of genre are outdated?
Ex= Clustering users on a dating site based on their information.
Perhaps you'll find there are groups of people that emerge that don't conform to your known stereotypes.

Supervised learning

The data the algorithm learns from comes with the "correct" answers.

The model created is then used to predict the answer for new, unknown values.

* Train/test is a great way to guard against overfitting.
* The data sets must be selected randomly
* Train/test is not infallible
↳ hatosiz

Naive Bayesian Method

I want a spam classifier. How would we express the probability of an e-mail being spam if it contains word "free"?

$$P(\text{Spam}|\text{Free}) = \frac{P(\text{Spam}) \cdot P(\text{Free}|\text{Spam})}{P(\text{Free})}$$

* The numerator is the probability of a message being spam and containing the word "free".
* The denominator is the overall probability of an email containing the word "free"

So together - this ratio is the % of emails with the word free that are spam.

We can construct P(Spam|Word) for every (meaningful) word we encounter during training.

Then multiply these together when analyzing a new email to get the probability of it being spam.

→ Scikit-learn to the rescue!

* The CountVectorizer lets us operate on lots of word at once and MultinomialNB does all the heavy lifting on Naive Bayes

K-means Clustering (Unsupervised learning technique).

→ the point is to find latent values

Attempts to split data into K groups that are closest to K centroids. Uses only the positions of each data point (unsupervised)

Ex= where do rich people live?

Maybe, there is some interesting geographical cluster where "rich" people live.

↳ Randomly pick K centroids (k-means)

How K-means work?

↳ It just keeps iterating until finding the right centroids.

* The hard part is trying to figure out what are that groups representing.

* Try increasing K values until you stop getting large reductions in square error (distances from each point to their centroids)

* The random choice of initial centroids can yield different results

Run it few times

Entropy

Measure of how disorder your data set is.
The entropy is 0 if all of the classes in the data are the same.

Decision Trees (supervised learning)

Constructing a flowchart to help you decide a classification for something with machine learning.

Random forests

* Decision trees are very susceptible to overfitting (→ bayes)
  - ↳ To fight this, we can construct several alternate decision trees (randomly re-sample the input data for each tree = bootstrap aggregating) and let them "vote" on the final classification.

Ensemble Learning

Random forest is an example.
It means using multiple models to try and solve the same problem, and let them vote on the results.

A bucket of models

trains several different models using training data, and picks the best

Stacking

runs multiple models at once on the data, and combines the results together

Boosting

You run a model, figure out its weak points, and try to focus on those weak points as you go

Bootstrap Aggregating

Support Vector Machines (SVM) (Supervised technique)

Works well for classifying higher-dimensional data (lots of features)

Support Vector Classification (SVC) for classifying data.
↳ You can use different "kernels" with SVC → (linear, RBF, Polynomial...)

Recommender Systems

User based collaborative Filtering

X	Matrix	Star Wars	King Lion	LOTR
User 1	✓	✓	X	✓
User 2	✓	✓	X	X
User 3	X	X	✓	X
User 4	✓	X	X	✓

⇒ You can recommend LOTR to user 2 since he has a similar interest to user 1

* Build a matrix of things each user bought/rated/viewed
* Compute similarity scores between users
Find users similar to you.
Recommend stuff they viewed/rated that you haven't yet.

(-) People are fickle; their tastes change.
There are usually lot more people than items (more computation)
Its easy to fabricate fake users for their item to recommended
(shilling attack)

Item based collaborative Filtering → Better, Amazon uses this technique

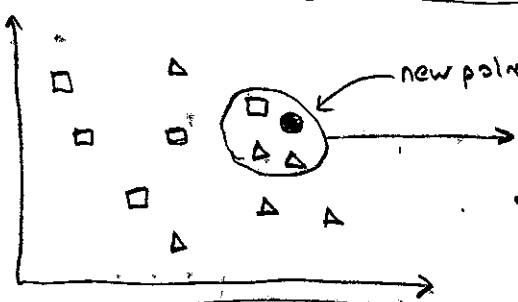
(+) Items dont change.
Usually fewer items than people (less computation to do).
Harder to game to system (It is difficult to create fake items)
* Its better the behaviour that you're basing it off of is based on people actually spending money (what people bought...
Because that way, it is lot harder to game the system.

Item-based Collaborative Filtering

* Find every pair of the movies that watched by the same person
* Measure the similarity of their ratings across all users who watched both.
* Find similar movies (People who liked also liked) also dislike

(This is just way to do it) similarity

K-Nearest Neighbours (KNN)



Dimensionality Reduction

→ Attempts to distill higher-dimension data down to a smaller number of dimensions.

→ K-means clustering is the most basic example. It reduces data down to K dimensions.

Principal Component Analysis (PCA)

↳ Finds "eigenvectors" in the higher dimensional data

* Those define hyperplanes that split the data while preserving the most variance in it.
* The data gets projected onto these hyperplanes, which represent the lower dimensions you want to represent.

Really useful for things like image compression and facial recognition

* A popular implementation of this is called Singular Value Decomposition (SVD)

Data Warehousing

* A large, centralized database that contains information from many sources.
* Often used for business analysis in large organizations.
* Queried via SQL or tools (i.e. Tableau)

ETL: Extract, transform, load (Old school technique)

refers how data gets into a data warehouse.

But what if we're dealing with "big data"? That transform step can turn into a big problem.

ELT: Extract, Load, Transform

* Extract raw data as before
* Load it in as-is
* Then use the power of Hadoop to transform it in-place.

Things like "Hive" let you host massive databases on a Hadoop cluster.

Or, you might store it in a large, distributed NoSQL data store -- and query it using things like Spark or MapReduce

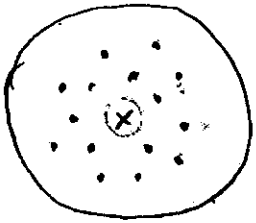
Reinforcement Learning

It's useful technique for building an agent that can navigate it's way through a possible set of states. Similar to maze problem. Whenever you have a situation where you need to predict behaviour of something, given a set of current conditions and a set of actions it can take, reinforcement and Q learning might be a way of doing it.

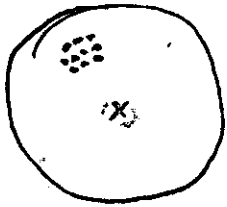
Bias and Variance

* Bias is how far removed the mean of your predicted values is from the "real" answer.
* Variance is how scattered your predicted values are from the "real" answer.

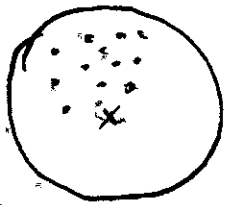
↳ Note: Real value we're trying to predict is in the centre.



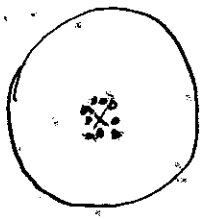
→ low bias
high variance



→ high bias
low variance



→ high bias
high variance (the Worst!)



→ low bias
low variance

$$\text{Error} = \text{bias}^2 + \text{variance}$$

A complex model will have high variance and low bias

A too simple model will have low variance and low bias
↳ under-fitting
↳ over-fitting

K-Fold Cross Validation

One way to protect against overfitting:

* Split your data into K randomly-assigned segments
* Reserve one segment as your test data
* Train on each of the remaining K-1 segments and measure their performance against the test set
* Take the average of the K-1 r-squared values.

Cleaning Data

The reality is, much of your time as a data scientist will be spent preparing and "cleaning" data.

Problems

→ Outliers (could be a attack) (something human can not do i.e.)
→ Missing Data (Do you throw that out? Do you create new class?
→ Malicious Data (actually attack)
→ Erroneous Data (could be software bug)
→ Irrelevant Data ("beside the point" data)
→ Inconsistent Data (huge problem) (means unstable)
  ↳ i.e. not everyone writes their address in the same orde
→ Formatting (i.e. datetime is different in U.S.A YYYY/MM/DD

* Clean data is extremely important

Normalizing Data

Some models may not perform well when different attributes are on very different scales

↳ i.e. ages may range from 0-100, and incomes from 0-billions.
↳ P.C.A implementation has a whiten option i.e.
* "Yes" and "No" needs to be converted to "1" and "0".

But dont forget the rescale your results when you're done.

* Not every algorithm (model) needs that! You can read its documentation.

Outliers

Some of them could be real (not fake or malicious)

↳ Depends on the situation, sometimes its appropriate to remove them. For Ex: a single user who rates thousands of movies could have a big effect on everyone elses ratings. It could be removed / But you can't toss out Trump just because he is extremely rich when you're dealing with mean "income"

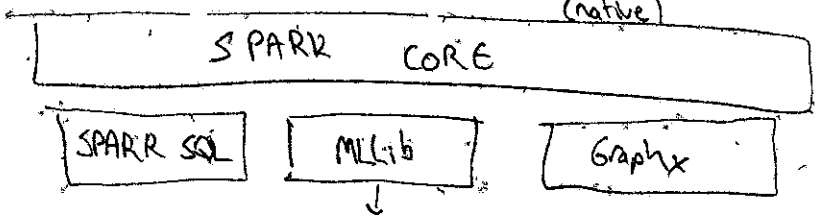
↳ A person should not have such power in your system (rating every movie)

Spark (Big data software)

A fast and general engine for large-scale data processing.

Basically, Spark works by letting you load your data, it automatically perform operations that transform your data, optimally spread that processing out amongst an entire cluster of computers, so no longer you are restricted to what you can do on a single machine, or single machine memory.

* Spark is 10x faster than disk
* Spark uses DAG ('directed' acyclic graph) engine
  ↳ Spark waits until you tell it to actually produce a result and only at that point does it actually go out and figure out how to produce that result (figuring the best way to split up processing and distribute that information to get the end result)
* Amazon, Ebay, NASA use Spark i.e.
* You can code in Python, Java or Scala. (native)



A/B test

A controlled experiment, usually in the context of a website.

→ Test the performance of some change to your website (the variant)

→ Measure conversion relative to your unchanged site (the control)

ie. → changing the color of purchase button (it could be anyth

What changed?

↳ Order amounts, Profit, Ad clicks, Order quantity

* How do you know the result you found is "real"?

You can't take the mean of all sales and decide you should understand not all of your products have a same price, or any other information.

The T-Test

A measure of difference between the two sets expressed in units of standard error.

The size of difference relative to the variance in data

A high T-value means there's probably a real difference between the two sets (which is great)

The p-value

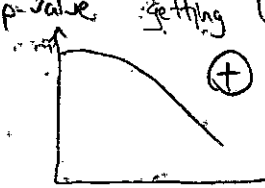
Low p-value means that there's a high probability that your change had a real effect. (which is great)

% 1 is great, % 5 is risky but acceptable.

* One day is enough for test, if the change has a highly negative impact it could be dangerous.
* Negative t-value means it is a bad change you made.
* How high is high for T-value? → that is debatable

NOTE: $\sqrt{\text{variance}}$ = Standart Deviation

If p-value getting low in the experiment, it is a good sign



* If p-value is lower than %1, you can finish the test

Some notes for A/B tests

* Users might click on something simply because it is new but this attention won't last forever.
  - ↳ Good idea to rerun experiments much late and validate their impact.

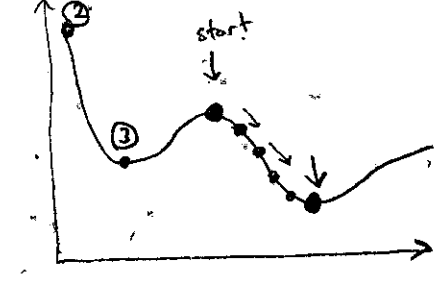
* An experiment run over a short period of time may only be valid to that period of time
  - ↳ i.e costumer behavior near Christmas is different than other times of year
    - ↳ Compare with the last years.

* A and B test groups should be fully random.
* You have to make sure they're not switching groups
  - ↳ Run an A/A test periodically to check.

* Are robots (both self-identified and malicious) affecting your experiment?.
  - ↳ Good reason to measure conversion based on something that requires spending money (real money)

DEEP LEARNING

Gradient descent



* We're picking some point at random with a given set of parameters that we measure the error for and pushing those parameters in a given direction until the error minimizes itself
* It could find the local minima, so what if it starts at point ② and end up at point ③ (not the least error).
  - ↳ In practical terms, it turns out that Local minima aren't really that big of a deal. Generally, you don't end up with shapes like this

Autodiff.

is a way of accelerating the process of finding local minima so we don't have to do quite as much math or quite as much computation to actually measure that gradient of the gradient descent.

* Tensorflow uses it

Softmax

Used for classification.

↳ Given a score for each class

↳ It produces probability of each class

↳ The class with the highest probability is the answer you get

Review

* Gradient deseent is an algorithm for minimizing error over multiple steps
* Autodiff is a calculus trick for finding gradients in gradient descent
* Softmax is a function for choosing the most probable classification given several input values

→ The deep learning and AI comes from biological inspiration, our own brain.

Backpropagation - - → gen system

So Gradient descent using reverse-mode autodiff for training Multi-Layer perceptron's (MLP's).

For each training step:

→ Compute the output error

→ Compute how much each neuron in the previous hidden layer contributed

→ Back-propagate that error in a reverse pass

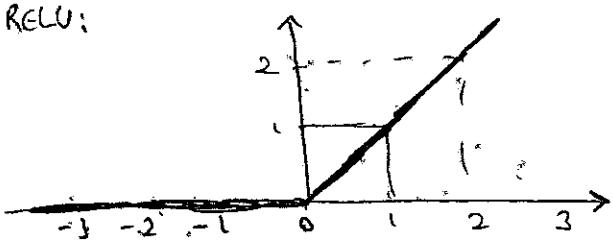
→ Tweak weights to reduce the error using gradient descent. (keep doing until system converges)

Activation function

function that determines the output of a neuron given the sum of its inputs.

→ RELU is common, Fast to compute and works well.

RELU:



if the sum is ≤0 → 0
if >0, gives the actual sum

*Leaky RELU: instead of flat zero, it has a little bit of slope at the left side of graph going down

ELU (Exponential Linear Unit)
↳ often produces faster learning, gaining popularity

Optimization functions

Adam: Adaptive moment estimation
↳ momentum optimizer and RMSprop combined,
* popular choice today because it works really well and it's very easy to use.

Momentum optimizer: It slows down as things starts to flatten and speeds up as the slope is steep.

RMSprop: Adaptive learning rate to help point toward minimum.

Nesterov Accelerated Gradient: computes momentum based on the gradient slightly ahead of you, not where you are
* (a small tweak on momentum optimization)

Avoiding overfitting

Early stopping: when performance starts dropping, stop!

Dropout: ignore say 50% of all neurons randomly at each training step:
* Works surprisingly well!
* Forces your model to spread out its learning.

→ More layers will often yield faster learning than having more neurons and less layers.

→ Model Zoo: A site for discovering pre-trained models.

Tensorflow

It's not specifically for neural networks - it's more generally an architecture for executing a graph of numerical operations.

* Tensor is just a fancy name for an array or matrix of values (structured collection of numbers)

→ Construct a graph describing our neural network
↳ Use placeholders for the input data and target labels
↳ This way we can use the same graph for training and tes

* Make sure your features are normalized
↳ That is, 0 mean and unit variance
↳ The real goal is that every input feature is comparable in terms of magnitude.
→ Scikit-learn's StandardScaler can do this for you

one-hot code → [2 → [0.0.1.0.0]] [3 → [0.0.0.1.0]]
0 1 2 3 4

KERAS

→ Available as higher-level API in Tensorflow 1.9+
→ Using Keras, you can integrate your deep neural networks with scikit-learn.
→ Created for deep learning, so often you can get better results without doing as much work, with less to think about.

Convolutional Neural Networks (CNN)

To find things in your data!
For Example: Finding stop sign in the image
Finding Nouns/Verbs in the text.

*Convolution is just a fancy word of saying "I'm going to break up this data into little chunks and process those chunks individually.

→ MaxPooling2D layers can be used to reduce a 2D layer down by taking the maximum value in a given block. (reducing size)
→ Flatten layers will convert 2D layer to a 1D layer

Typical usage for image processing

Conv2D → MaxPooling2D → Dropout → Flatten → Dense → Dropout → Softmax

Recurrent Neural Networks (RNN)

To predict future behavior based on past behavior.
For example: Machine-generated music.

A recurrent neuron

new data coming in that gets blended together with the output from the previous run through this neuron. So the past behavior of this neuron influences its future behavior.

* You can scale this out to have more than one neuron (output of the layer can get fed back to neuron

RNN Topologies

Sequence to sequence: Predict stock prices based on series of historical da
Sequence to vector: Words in a sentence to sentiment
Vector to sequence: Create captions from an image
Encoder → Decoder: Machine translation (sequence → vector → sequenc

LSTM (Long short-Term Memory cell)

If you're dealing with sequence of data where you don't want to give preferential treatment to more recent data, you probably want to use LSTM. It maintains seperate short-term and long-term states. (GRU cell is another option)