Intro to AI & Machine Learning JumpStart

TriveraTech
TECHNOLOGY TRAINING

Table of Contents

1. What is machine learning? : 3
2. Types of machine learning : 31
3. Drawing a line close to our points: Linear regression : 74
4. Using lines to split our points: The perceptron algorithm : 126
5. A continuous approach to splitting points: Logistic regression : 231
6. Using probability to its maximum: The naive Bayes algorithm: 301
7. Splitting data by asking questions: Decision trees : 370
8. Combining models to maximize Results Ensemble learning : 463

1: What is machine learning?

TriveraTech
TECHNOLOGY TRAINING

What is machine learning?

This Lesson covers:

- What is machine learning?
- Is machine learning hard? (Spoiler: No)
- Why you should take this course?
- What will we learn in this course?
- How do humans think, how do machines think, and what does this have to do with machine learning?

Why this Course?

- We play the music of machine learning; the formulas and code come later.

- Most of the times, when I read a machine learning or attend a machine learning lecture, I see either a sea of complicated formulas, or a sea of lines of code.

- For a long time, I thought this was machine learning, and it was only reserved for those who had a very solid knowledge of both.

Why this Course?

- This course is a step on that journey, that I'm very happy you're taking with me!

Music

Machine learning

- **There is also a melody, and in this course we sing it.**
- **In the same way, machine learning is not about formulas and code.**
- **Music is not only about scales and notes. There is a melody behind all the technicalities.**

Is machine learning hard?

- Machine learning requires imagination, creativity, and a visual mind.
- It helps a lot if we know mathematics, but the formulas are not required.
- It helps if we know how to code, but nowadays, there are many packages and tools that help us use machine learning with minimal coding.

But what exactly is machine learning?

- Once upon a time, if we wanted to make a computer perform a task, we had to write a program, namely, a whole set of instructions for the computer to follow.

- This is good for simple tasks, but how do we get a computer to, for example, identify what is in an image?

- For example, is there a car in it, is there a person in it.

But what exactly is machine learning?

- Machine learning is common sense, except done by a computer.

In the same way that humans make decisions based on previous experiences, computers can make decisions based on previous data. The rules computers use to make decisions are called models. Machine learning is about computers making decisions based on experience.

What is the difference between artificial intelligence and machine learning?

Machine learning is a part of artificial intelligence.

What is the difference between artificial intelligence and machine learning?

I think of artificial intelligence in the following way:

- Artificial intelligence encompasses all the ways in which a computer can make decisions.

- When I think of how to teach the computer to make decisions, I think of how we as human make decisions.

What about deep learning?

- If this course was about vehicles, then AI would be motion, ML would be cars, and deep learning (DL) would be Ferraris.

Deep learning is a part of machine learning.

Humans use the remember-formulate-predict framework to make decisions (and so can machines!)

- How does the computer make decisions based on previous data?
- For this, let’s first see the process of how humans make decisions based on experience.
- And this is what I call the remember-formulate-predict framework.

How do humans think?

When we humans need to make a decision based on our experience, we normally use the following framework:

1. We remember past situations that were similar.
2. We formulate a general rule.
3. We use this rule to predict what will happen if we take a certain decision.

How do humans think?

SPAM AND HAM

- Spam is the common term used for junk or unwanted email, such as chain letters, promotions, and so on.
- The term comes from a 1972 Monty Python sketch in which every item in the menu of a restaurant contained spam as an ingredient.

How do humans think?

- We may be right or wrong, but at least, we are trying to make an accurate prediction.

The remember-formulate-predict framework

How do humans think?

Rule 1: 4 out of every 10 emails that Bob sends us are spam.

How do humans think?

- Let’s try to analyze the emails a little more.
- Let’s see when Bob sent the emails to see if we find a pattern.

A very simple machine learning model.

How do humans think?

Monday: Ham
Tuesday: Ham
Saturday: Spam
Sunday: Spam
Sunday: Spam
Wednesday: Ham

Friday: Ham
Saturday: Spam
Tuesday: Ham
Thursday: Ham

How do humans think?

- In the next we'll see a few more features.

A slightly more complex machine learning model, done by a human.

How do humans think?

Example 3: Things are getting complicated!

- Now, let’s say we continue with this rule, and one day we see Bob in the street, and he says “Why didn’t you come to my birthday party?”
- We have no idea what he is talking about.

How do humans think?

1KB: Ham
12KB: Ham
16KB: Spam
20KB: Spam
18KB: Spam

3KB: Ham
5KB: Ham
25KB: Spam
1KB: Ham
3KB: Ham

How do humans think?

Rule 3: Any email larger of size 10KB or more is spam, and any email of size less than 10KB is ham.

How do humans think?

- We look at the email we received today, and the size is 19KB.
- So we conclude that it is spam.

Another slightly more complex machine learning model, done by a human.

How do humans think?

Rule 4: If an email is larger than 10KB or it is sent on the weekend, then it is classified as spam. Otherwise, it is classified as ham.

Rule 5: If the email is sent during the week, then it must be larger than 15KB to be classified as spam. If it is sent during the weekend, then it must be larger than 5KB to be classified as spam. Otherwise, it is classified as ham.

How do humans think?

- It is classified as ham.

An even more complex machine learning model, done by a human.

How do machines think?

- **Remember:** Look at a huge table of data.
- **Formulate:** Go through many rules and formulas, and check which one fits the data best.
- **Predict**: Use the rule to make predictions about future data.

How do machines think?

- then we classify the message as spam.
- Otherwise we do not.

A much more complex machine learning model, done by a computer.

What is this course about?

Good question. The 8 rules we discussed previously are examples of machine learning models, or classifiers. As you saw, these are of different types.

- Some use an equation on the features to make a prediction.
- Others use a combination of if statements.
- Others will return the answer as a probability.
- Others may even return the answer as a number!

Summary

As follows:

- **Remember:** Use previous data.
- **Formulate:** Build a model, or a rule, for this data.
- **Predict:** Use the model to make predictions about future data.

2: Types of machine learning

Types of machine learning

This Lesson covers:

- Three main different types of machine learning.
- The difference between labelled and unlabeled data.
- What supervised learning is and what it’s useful for.
- The difference between regression and classification, and what are they useful for.
- What unsupervised learning is and what it’s useful for.
- What reinforcement learning is and what it’s useful for

Types of machine learning

- Can you think of some fields in which you can apply machine learning? Here are some of my favorites:
- Predicting housing prices based on their size, number of rooms, location, etc.
- Predicting the stock market based on other factors of the market, and yesterday’s price.

Types of machine learning

- The main three families of machine learning models are: supervised learning, unsupervised learning, and reinforcement learning.

- In this Lesson, we overview them all, However, in this course, we only cover supervised learning, as it is the most natural one to start learning, and arguably the most commonly used.

What is the difference between labelled and unlabelled data?

Actually, what is data?

- Let’s first establish a clear definition of what we mean by data.

Data is simply information.

- Any time we have a table with information, we have data. Normally, each row is a data point.

What is the difference between labelled and unlabelled data?

Ok. And what are features?

- Features are simply the columns of the table.
- In our pet example, the features may be size, name, type, weight, etc.
- This is what describes our data, some features are special, though, and we call them labels.

What is the difference between labelled and unlabelled data?

So now we can define two very important things, labeled and unlabeled data.

- **Labeled data:** Data that comes with a label.
- **Unlabeled data:** Data that comes without a label.

What is the difference between labelled and unlabelled data?

Labeled data is data that comes with a tag, like a name, a type, or a number. Unlabeled data is data that comes with no tag.

What is supervised learning?

- Supervised learning is the type of machine learning you find in the most common applications nowadays, including image recognition, various forms of text processing, recommendation systems, and many more.

- As we stated in the previous section, it is a type of predictive machine learning in which the data comes with labels, where the label is the target we are interested in predicting.

What is supervised learning?

- This means, if we bring in a new image without a label, the model would guess if the image is of a dog or a cat, thus predicting the label of the data point.

A supervised learning model predicts the label of a new data point.

What is supervised learning?

Supervised learning follows the Remember-Formulate-Predict framework from Lesson 1.

- The model makes a prediction about what the label of the image is, namely, is it a dog or a cat.

Remember

Formulate

Predict

What is supervised learning?

- **Regression models:** These are the types of models that predict a number, such as the weight of the animal.

- **Classification models:** These are the types of models that predict a state, such as the type of animal (cat or dog).

What is supervised learning?

- You can see the difference between models 1 and 2.

- Example 1, the housing prices model, is a model that can return many numbers, such as

1. $100
2. $250,000
3. $3,125,672

Thus it is a regression model.

What is supervised learning?

Example 2, the spam detection model, on the other hand, can only return two things: spam or ham.

- Thus it is a classification model.
- Let’s elaborate some more on regression and classification.

Regression models predict numbers

- This number is predicted from the features.
- In the housing example, the features can be the size of the house.
- The number of rooms, the distance to the closest school, the crime rate in the neighborhood, etc.

Regression models predict numbers

Other places where one can use regression models are the following:

- **Stock market:** Predicting the price of a certain stock based on other stock prices, and other market signals.

- **Medicine:** Predicting the expected lifespan of a patient, or the expected recovery time, based on symptoms and the medical history of the patient.

Classification models predict a state

- Classification models are those that predict a state, from a finite set of states.
- The most common ones predict a ‘yes’ or a ‘no’, but there are many models which use a larger set of states.

Classification models predict a state

- The example we saw in this figure is of classification, as it predicts the type of the pet, namely, ‘cat’ or ‘dog’.

Remember

Formulate

Predict

Classification models predict a state

- **Sentiment analysis**: Predicting if a movie review is positive or negative, based on the words in the review.

- **Website traffic**: Predicting if a user will click on a link or not, based on the user's demographics and past interaction with the site.

- **Social media**: Predicting if a user will befriend or interact with another user or not, based on their demographics, history, and friends in common.

What is unsupervised learning?

- Unsupervised learning is also a very common type of machine learning.
- It differs from supervised learning in that the data has no labels.
- What is a dataset with no labels, you ask? Well, it is a dataset with only features, and no target to predict.
- For example, if our housing dataset had no prices, then it would be an unlabeled dataset.

What is unsupervised learning?

An unsupervised learning model can still extract information from data, for example, it can group similar elements together.

What is unsupervised learning?

- **Clustering:** This is the task of grouping our data into clusters based on similarity.
- **Dimensionality reduction:** This is the task of simplifying our data and describing it with fewer features, without losing much generality.

Clustering algorithms split a dataset into similar groups

- let’s look at examples we have a dataset with information about houses, but no prices.
- What could we do? Here is an idea: we could somehow group them into similar houses.
- We could group them by location, by price, by size, or by a combination of these factors, this is called clustering.

Clustering algorithms split a dataset into similar groups

- Let’s look at our second example which have a dataset of emails.
- Because the dataset is unlabeled, we don’t know if each email is spam or not.
- However, we can still apply some clustering to our dataset.

Clustering algorithms split a dataset into similar groups

- For example, let's say that we have 9 emails, and we want to cluster them into different types.
- We have, say, the size of the email, and the number of recipients.
- And the data looks like this, ordered by number of recipients:

Clustering algorithms split a dataset into similar groups

Email	Size	Recipients
1	8	1
2	12	1
3	43	1
4	10	2
5	40	2
6	25	5
7	23	6
8	28	6
9	26	7

- To the naked eye, it looks like we could group them by size, where the emails in one group would have 1 or 2 recipients, and the emails in the other group would have 5 or more recipients.

Clustering algorithms split a dataset into similar groups

- We get the following plot.

A plot of the emails with size on the horizontal axis and number of recipients on the vertical axis. Eyeballing it, it is obvious that there are three distinct types of emails.

Clustering algorithms split a dataset into similar groups

- They are the ones we see in this Figure

Clustering the emails into three categories based on size and number of recipients.

Clustering algorithms split a dataset into similar groups

Other applications of clustering are the following:

- **Market segmentation:** Dividing customers into groups based on demographic and purchasing (or engagement) behavior, in order to create different marketing strategies for the groups.
- **Genetics:** Clustering species into groups based on similarity.
- **Medical imaging:** Splitting an image into different parts in order to study different types of tissue.

Dimensionality reduction simplifies data without losing much information

Let’s say that we want to predict the price, and the features are the following:

- Size.
- Number of bedrooms.
- Number of bathrooms.
- Crime rate in the neighborhood.
- Distance to the nearest school.

Dimensionality reduction simplifies data without losing much information

- That is five columns of data.
- What if we wanted a simpler dataset, with fewer columns, but that can portray the information in as faithful a way as possible.
- Let’s do it using common sense.
- Take a closer look at the five features.
- Can you see any way to simplify them, maybe to group them into some smaller and more general categories?

Dimensionality reduction simplifies data without losing much information

- Using dimensionality reduction to reduce the number of features in a housing dataset, without losing much information.

Dimensionality reduction simplifies data without losing much information

- When we went from five dimensions down to two, we reduced our 5-dimensional city into a 2-dimensional city, thus applying dimensionality reduction.

Matrix factorization and other types of unsupervised learning

- It seems that clustering and dimensionality reduction look very different, but in reality they are not so different.
- If we have a table full of data, each row is a data point, and each column is a feature.

Matrix factorization and other types of unsupervised learning

Matrix factorization and other types of unsupervised learning

Matrix factorization and other types of unsupervised learning

- Places like Netflix use matrix factorization extensively to make recommendations.
- Imagine a large table where each row is a user, each column is a movie, and each entry in the matrix is the rating that the user gave the movie.

What is reinforcement learning?

- Reinforcement learning is a different type of machine learning, in which no data is given, and we must solve a problem.
- Instead of data, an environment is given, and an agent who is supposed to navigate in this environment.

What is reinforcement learning?

Example 1: Grid world

- We seen a grid world with a robot on the bottom left corner that is our agent.
- The goal is to get to the treasure chest in the top right of the grid.
- In the grid, we can also see a mountain, which means we cannot go through that square, since the robot cannot climb mountains.

What is reinforcement learning?

What is reinforcement learning?

What is reinforcement learning?

Reinforcement learning has numerous cutting edge applications, and here are some of them

1. Games
2. Robotics
3. Self driving cars

Summary

- There are several types of machine learning, including supervised learning and unsupervised learning.
- Supervised learning is used on labelled data, and it is good for making predictions.
- Unsupervised learning is used on unlabelled data, and it is normally used as a preprocessing step.

3: Drawing a line close to our points: Linear regression

TriveraTech
TECHNOLOGY TRAINING

Drawing a line close to our points: Linear regression

This lesson covers

- What is linear regression?
- How to predict the price of a house based on known prices of other houses
- How to fit a line through a set of data points.
- How to code the linear regression algorithm in Python.

Drawing a line close to our points: Linear regression

- The mental picture of linear regression is simple.
- Let us say we have some points, that roughly look like they are forming a line, like in this figure

Drawing a line close to our points: Linear regression

- The goal of linear regression is to draw the line that passes as close as possible to these points.
- What line would you draw, that goes close to those points?
- The one I drew is in this figure.

Drawing a line close to our points: Linear regression

- What do we mean by “points that roughly look like they are forming a line”?
- What do we mean by “a line that passes really close to the points”?
- How do we (or the computer) find such a line?

The problem: We need to predict the price of a house

- Let’s say we are a real estate agent, and we are in charge of selling a new house.
- We don’t know the price, and we want to infer it by comparing it with other houses.

The solution: Building a regression model for housing prices

No of Rooms	Price
1	150
2	200
3	250
4	?
5	350
6	400
7	450

The solution: Building a regression model for housing prices

Take a look at table on the previous slide.

- What price would you give to house 4, just from this table? If you said $300, that is also my guess.
- You probably saw a pattern, and used it to infer the price of the house.
- Congratulations! What you did in your head was linear regression.

The solution: Building a regression model for housing prices

More specifically, we can think of the price of a house as a combination of two things:

- A base price of $100, and an extra charge of $50 for each of the rooms.
- This can be summarized in this simple formula:

Price = 100 + 50 * (Number of rooms)

The remember step: looking at the prices of existing houses

No of Rooms	Price
1	155
2	197
3	244
4	?
5	356
6	407
7	448

The remember step: looking at the prices of existing houses

The formulate step: formulating a rule that estimates the price of the house

- The dataset in previous table is close enough to the one in Table 2, so we should feel safe to use the same formula for the price.
- The only difference is that now the prices are not exactly what the formula says, and we have a small error.
- The formula is as follows:

Price = 100 + 50*(Number of rooms) + (Small error)

The formulate step: formulating a rule that estimates the price of the house

- Now the question is, how did we find this equation?
- And most importantly, how does a computer find this equation?
- Let’s go back to the plot, and see what the equation means there.
- What happens if we look at all the points in which the vertical (y) coordinate is 100 plus 50 times the horizontal (x) coordinate? This set of points forms a line.

The formulate step: formulating a rule that estimates the price of the house

- The y-intercept being 100 means that our estimate for the price of a (hypothetical) house with zero rooms, would be the base price of $100.

- The slope being 50 means that each time we add one room to the house, we estimate that the price of the house will go up by $50.

- This line is drawn in next figure.

The formulate step: formulating a rule that estimates the price of the house

TriveraTech
TECHNOLOGY TRAINING

The predict step: what do we do when a new house comes in the market?

- Now, on to using our model to predict the price of the house with 4 rooms.

- This is very simple, all we do is plug the number 4 into our formula, to get the following:

Price = 100 + 50*4 = 300.

The predict step: what do we do when a new house comes in the market?

- Therefore, our model predicted that the house costs $300.
- This can also be seen graphically by using the line, as illustrated in this figure.

How to get the computer to draw this line: the linear regression algorithm

- Let's make a model that says that the price is
  $51 + $41*(Number of rooms).
- This model assigns the house a price of
  $51 + $41*2 = $133,
  which is closer to **$150**.

- It's not perfect, but it is a little better
- What if instead we had a house with two rooms that costs $80? Then we'll decrease the base price and the price per room a little bit.

How to get the computer to draw this line: the linear regression algorithm

1. Start with random values for the base price and the price per room.
2. Repeat many times:

a) Pick a random house, and use it to increase or decrease the base price and price per room by a small amount, so the model will predict its price better.

3. Enjoy your model!

How to get the computer to draw this line: the linear regression algorithm

1. Picking a random point.
2. Moving the line towards this point, by a very small amount.

- You may think, what if I have a point that is really far away, and it messes up my procedure?

- That is ok, if we only move the line by small amounts, this point can’t harm us very much.

Crash course on slope and y-intercept

When we think of a line, we can think of it graphically, or as an equation. If we think of it as an equation, then it has two components:

- The slope.
- The y-intercept

Crash course on slope and y-intercept

$y = 0.5x + 2.$

Slope = 0.5

y-intercept = 2

Crash course on slope and y-intercept

- What does this equation mean? It means that the slope is 0.5, and the y-intercept is 2.
- The slope being 0.5 means that when we walk along this line, for every unit that we move towards the right, we are moving 0.5 units up.
- The slope can be zero if we don’t move up at all, or negative if we move down.
- However, many lines can have the same slope.

A simple trick to move a line closer to a set of points, one point at a time

Now we get to our problem, which is:

- we have a point, and a line, and we need to move the line closer to the point.

TriveraTech
TECHNOLOGY TRAINING

A simple trick to move a line closer to a set of points, one point at a time

TriveraTech
TECHNOLOGY TRAINING

A simple trick to move a line closer to a set of points, one point at a time

Rotate clockwise and counterclockwise

Translate up and down

A simple trick to move a line closer to a set of points, one point at a time

A simple trick to move a line closer to a set of points, one point at a time

A simple trick to move a line closer to a set of points, one point at a time

Case 1: If the price of the house is higher than the price the model predicted, and the number of rooms is positive:

- Add 1 cent to the price per room
- Add 1 cent to the base price.

Case 2: If the price of the house is higher than the price the model predicted, and the number of rooms is negative:

- Subtract 1 cent to the price per room
- Add 1 cent to the base price

A simple trick to move a line closer to a set of points, one point at a time

Case 3: If the price of the house is lower than the price the model predicted, and the number of rooms is positive:

- Add 1 cent to the price per room
- Subtract 1 cent to the base price.

Case 4: If the price of the house is lower than the price the model predicted, and the number of rooms is negative:

- Subtract 1 cent to the price per room
- Subtract 1 cent to the base price.

A simple trick to move a line closer to a set of points, one point at a time

```
def simple_trick(base_price, price_per_room, num_rooms, price, learning_rate):  #A
    predicted_price = base_price + price_per_room*num_rooms  #B
    if price > predicted_price and x > 0:  #C
        price_per_room += learning_rate  #D
        base_price += learning_rate  #E
    if price > predicted_price and x < 0:
        price_per_room -= learning_rate
        base_price += learning_rate
    if price > predicted_price and x > 0:
        price_per_room += learning_rate
        base_price -= learning_rate
    if price > predicted_price and x < 0:
        price_per_room -= learning_rate
        base_price -= learning_rate
    return price_per_room, base_price
```

The square trick: A much more clever way of moving our line closer to one of the points

- **OBSERVATION 1** In the simple trick, when the point is above the line, we add a small amount to the y-intercept, and when the point is below the line, we subtract a small amount to the y-intercept.

The square trick: A much more clever way of moving our line closer to one of the points

- **OBSERVATION 2** If a point is to the above the line, the difference of price minus predicted price is positive.
- If it is below the line, then this difference is negative.

The square trick: A much more clever way of moving our line closer to one of the points

The square trick: A much more clever way of moving our line closer to one of the points

- **OBSERVATION 3** In the simple trick, when the point is either above the line and to the right of the y-axis, or below the line and to the left of the y-axis, we add a small amount to the slope.
- Otherwise, we subtract a small amount to the slope

The square trick: A much more clever way of moving our line closer to one of the points

- **OBSERVATION 4** If the point is to the right of the y-axis, then the number of rooms is positive.
- If the point is to the left of the y-axis, then this quantity is negative.

The square trick: A much more clever way of moving our line closer to one of the points

The square trick: A much more clever way of moving our line closer to one of the points

- **Case 1:** If the price of the house is higher than the price the model predicted, and the number of rooms is positive:

(Price - Predicted price) * (Number of rooms) is positive.

- **Case 2:** If the price of the house is higher than the price the model predicted, and the number of rooms is negative:

(Price - Predicted price) * (Number of rooms) is negative

The square trick: A much more clever way of moving our line closer to one of the points

- **Case 3:** If the price of the house is lower than the price the model predicted, and the number of rooms is positive:

(Price - Predicted price) * (Number of rooms) is negative.

Case 4: If the price of the house is lower than the price the model predicted, and the number of rooms is negative:

(Price - Predicted price) * (Number of rooms) is positive.

The square trick: A much more clever way of moving our line closer to one of the points

All cases:

- Add the learning rate
  *** (Price - Predicted price) * (Number of rooms)**
  to the price per room.

- Add the learning rate
  *** (Price - Predicted price)**
  to the base price.

The square trick: A much more clever way of moving our line closer to one of the points

- And here is the code:

```
def square_trick(price_per_room, base_price, num_rooms, price, learning_rate):
    predicted_price = base_price + price_per_room*num_rooms  #A
    base_price += learning_rate*(price-predicted_price)  #B
    price_per_room += learning_rate*num_rooms*(price-predicted_price)  #C
    return price_per_room, base_price
```

The linear regression algorithm: Repeating the square trick many times

- **PSEUDOCODE FOR FITTING A LINE THROUGH A SET OF POINTS (LINEAR REGRESSION)**

- Start with random values for the slope and y-intercept
- Repeat many times:
    - o Pick a random point
    - o Update the slope and the y-intercept using the square (or the simple) trick.

The linear regression algorithm: Repeating the square trick many times

```
import random  #A
def linear_regression(features, labels, learning_rate=0.01, epochs = 1000):
    price_per_room = random.random()
    base_price = random.random()  #B
    for i in range(epochs):  #C
        i = random.randint(0, len(features)-1)  #D
        num_rooms = features[i]
        price = labels[i]
        price_per_room, base_price = square_trick(base_price,         #E
                                                  price_per_room,
                                                  num_rooms,
                                                  price,
                                                  learning_rate=learning_rate)
    return price_per_room, base_price
```

Plotting dots and lines

- The first plot we'll show here, which is the plot of the points in our small housing dataset.

Using the linear regression algorithm in our dataset

- Now, let's apply the algorithm to our dataset! The following line of code runs the algorithm (using the square error) with the features, the labels, the learning rate equal to 0.01, and the number of epochs equal to 10000.

- The result is a plot of figure which is on next slide.

```
linear_regression(features, labels, learning_rate = 0.01, epochs = 10000)
```

Using the linear regression algorithm in our dataset

```
Price per room: $51.07296115119787
Base price: $99.47510567502614
```

Using the linear regression algorithm in our dataset

Starting point

Epochs 1-10

Epochs 1-50

Epochs 51-10,000

Applications of linear regression

- The impact of machine learning is measured not only by the power of its algorithms, but also by the breadth of useful applications it has.
- Machine learning is used widely to generate good recommendations in some of the most well known apps, including YouTube, Netflix, Facebook, Spotify, and Amazon.
- Regression plays a key part in most of these recommender systems.

Video and music recommendations

- One of the ways used to generate video and music recommendations is to predict the amount of time a user will watch a video or listen to a song.

- For this, one can create a linear regression model where the labels on the data are the amount of minutes that each song is watched by each user.

Product recommendations

- Stores and e-commerce websites also use linear regression to predict their sales.
- One way to do this is to predict how much will a customer spend in the store.

Healthcare

Regression has numerous applications in healthcare. Depending on what problem we want to solve, predicting the right label is the key. Here are a couple of examples:

- Predicting the lifespan of a patient, based on their current health conditions.
- Predicting the length of a hospital stay, based on current symptoms.

Summary

- Regression is a very important part of machine learning.
- It consists of training an algorithm with labelled data, and using it to make predictions on future (unlabeled) data.
- Labelled data is data that comes with labels, which in the regression case, are numbers.
- For example, the numbers could be prices of houses.

4: Using lines to split our points: The perceptron algorithm

TriveraTech
TECHNOLOGY TRAINING

Using lines to split our points: The perceptron algorithm

This lesson covers

- What is classification?
- Sentiment analysis: How to tell if a sentence is happy or sad using machine learning.
- What are perceptron's, or discrete logistic classifiers.
- What is the perceptron algorithm?
- How to draw a line that separates points of two different colors.
- How to measure the error of a line that separates points of two different colors.
- How to code the perceptron algorithm in Python.

Using lines to split our points: The perceptron algorithm

Sentiment analysis is used in many practical applications. Here are some examples:

- Companies analyzing the conversations between customers and technical support, to see if the service is good.

- Companies analyzing their online reviews and tweets, to see if their products or marketing strategies are well received.

Using lines to split our points: The perceptron algorithm

Here are some words and scores off the top of my head:

- Wonderful: 4 points
- Delighted: 3 points
- Happy: 2 points
- Sad: -2 points
- Awful: -4 points
- Every other word: 0 points.

Using lines to split our points: The perceptron algorithm

Sentence 1:	I	feel	awful	today	
Score:	+0	+0	-4	+0	= -4

Prediction: The sentence is sad.

Sentence 2:	Everything	is	wonderful	I	am	delighted
Score:	+0	+0	+4	+0	+0	+3

= +7

Prediction: The sentence is happy.

Using lines to split our points: The perceptron algorithm

1. How do we know the score of every single word?

2. The scores we came up with are based on our perception. How do we know these are the good scores?

3. What if we have to build a sentiment analysis in a language that we don’t speak?

Using lines to split our points: The perceptron algorithm

4. The sentences “I am not sad, I am happy.”, and “I am not happy, I am sad.” have completely different sentiments, yet they have the exact same words, so they would score the same, no matter the classifier, How do we account for this?

5. What happens with punctuation? What about sarcastic comments? What about the tone of voice, time of the day, or the overall situation? Too many things to take into account, aaahhh!

Using lines to split our points: The perceptron algorithm

- There are classifiers that take into account the order of the words, and they have great performance.
- Of course, they require lots of data and more computational power, although they are out of the scope of this course
- I recommend you to look at recurrent neural networks, long short-term memory networks (LSTM), and hidden Markov models.

The problem: We are in an alien planet, and we don't know their language!

- Imagine the following scenario, we are astronauts and we have just landed on a distant planet, where a race of unknown aliens lives.

- We would like to interact a bit with the aliens, but for this, we feel like we need a system that will help us determine if any alien we encounter is happy or sad.

The problem: We are in an alien planet, and we don't know their language!

Dataset:

- Alien 1
  - Mood: Happy
  - Sentence: "Aack, aack, aack!"
- Alien 2:
  - Mood: Sad
  - Sentence: "Beep beep!"
- Alien 3:
  - Mood: Happy
  - Sentence: "Aack beep aack!"
- Alien 4:
  - Mood: Sad
  - Sentence: "Aack beep beep beep!"

The problem: We are in an alien planet, and we don't know their language!

Data

Aack aack aack!

Beep beep!

Aack beep aack!

Aack beep beep beep!

Prediction

Is this alien happy or sad?

aack beep aack aack!

Simple sentiment analysis classifier

- Count the number of appearances of ‘aack’ and ‘beep’.
- If there are more ‘aack’s, then the alien is happy.
- If there are more ‘beep’s, then the alien is sad.
- Notice that we didn’t specify what happens when the number of ‘aack’s is equal to the number of ‘beep’s.

Mathematical sentiment analysis classifier

The classifier is defined by some scores, and a rule, as follows:

- **Scores:**

1. Aack: 1 point
2. Beep: -1 points

- **Rule:** Add the scores of all the words.

1. If the score is positive or zero, predict that the alien is happy.
2. If the score is negative, predict that the alien is sad.

Mathematical sentiment analysis classifier

Sentence	Aack	Beep	Mood
Aack aack aack!	3	0	Happy
Beep beep!	0	2	Sad
Aack beep aack!	2	1	Happy
Aack beep beep beep!	1	3	

Mathematical sentiment analysis classifier

Mathematical sentiment analysis classifier

#aack - #beep = 0.

If you like equations with variables, this is simply the equation **y=x**, or equivalently, the equation **y-x=0**. We also have two areas, as follows:

- The positive (happy) area, where the sentences have more aack's than beep's, with equation **#aack - #beep > 0.**
- The negative (sad) area, where the sentences have more beep's than aack's, with equation **#aack - #beep < 0.**

Mathematical sentiment analysis classifier

Geometric sentiment analysis classifier

The classifier is defined by a line of equation **#aack - #beep = 0**, and a rule.

Rule:

- If the point is in the positive region of this line, with equation **#aack - #beep > 0**, or over the line, then the alien is classified as happy.
- If the point is in the positive region of this line, with equation **#aack - #beep < 0**, then the alien is classified as sad.

Mathematical sentiment analysis classifier

PERCEPTRON A perceptron is a classification model which consists of a set of weights, or scores, one for every feature, and a threshold.

- The perceptron multiplies each weight by its corresponding score, and adds them, obtaining a score.
- If this score is greater than or equal to the threshold, then the perceptron returns a ‘yes’, or a ‘true’, or the value 1.

A slightly more complicated planet

Dataset:

- Alien 1
  - Mood: Sad
  - Sentence: "Crack!"
- Alien 2:
  - Mood: Sad
  - Sentence: "Dunk."
- Alien 3:
  - Mood: Sad
  - Sentence: "Crack dunk!"
- Alien 4:
  - Mood: Happy
  - Sentence: "Crack crack dunk dunk."
- Alien 5:
  - Mood: Happy
  - Sentence: "Crack dunk dunk crack crack!"
- Alien 6:
  - Mood: Happy
  - Sentence: "Dunk dunk crack dunk crack!"

A slightly more complicated planet

Sentence	Crack	Dunk	Mood
Crack!	1	0	Sad
Dunk dunk!	0	2	Sad
Crack dunk!	1	1	Sad
Crack dunk dunk!	1	2	Sad

A slightly more complicated planet

Crack crack dunk dunk!	1	3	Happy
Crack dunk dunk crack crack!	2	2	Happy
Dunk dunk crack dunk crack!	3	2	Happy
Crack dunk dunk crack dunk!	2	3	Happy

Simple sentiment analysis classifier

- If the number of words spoken by the alien is 4 or more, then the alien is happy.
- If it is less than 3, the alien is sad. But let’s put that into the same language as the previous classifier.
- In this case, we can simply attach a score of 1 to each of the words.

Mathematical sentiment analysis classifier

- The classifier is defined by the scores, and a rule, as follows:

Scores:

1. Aack: 1 point
2. Beep: 1 points

Rule: Add the scores of all the words.

1. If the score is larger than or equal to 3.5, predict that the alien is happy.
2. If the score is smaller than 3.5, predict that the alien is sad.

Modified mathematical sentiment analysis classifier

The classifier is defined by the scores, a bias, and a rule, as follows:

Scores:

1. Aack: 1 point
2. Beep: 1 points
3. Bias: -3.5 points

Modified mathematical sentiment analysis classifier

Rule:
Add the scores of all the words plus the bias.

1. If the score is positive or 0, predict that the alien is happy.
2. If the score is negative, predict that the alien is sad.

Thus, we can slightly modify our perceptron definition to the following:

Modified mathematical sentiment analysis classifier

Modified mathematical sentiment analysis classifier

- In other words, the equation of this line is:

  #crack + #dunk = 3.5.

- For convenience, we'll write the equation as follows:

  #crack + #dunk - 3.5 = 0.

Geometric sentiment analysis classifier

The classifier is defined by a line of equation **#crack + #dunk - 3.5 = 0**, and a rule.

Rule:

- If the point is in the positive region of this line, with equation **#crack - #dunk - 3.5 >0**, then the alien is classified as happy.

- If the point is in the positive region of this line, with equation **#crack - #dunk - 3.5 <0**, then the alien is classified as sad.

Geometric sentiment analysis classifier

The bias, the y-intercept, and the inherent mood of a quiet alien

- A related question is, what happens when an alien says absolutely nothing.
- Is that alien happy or sad? In the latest model, if an alien says nothing, then the score of the sentence is 0 (since there are no words) plus the bias, which is -3.5.

The bias, the y-intercept, and the inherent mood of a quiet alien

Can you help me think of two scenarios in sentiment analysis, one where the threshold is positive, and one where it's negative? Here's what I can think, based on my experience, but maybe you can find better examples!

- Positive threshold (or negative bias)
- Negative threshold (or positive bias)

The bias, the y-intercept, and the inherent mood of a quiet alien

Negative bias
(positive y-intercept)
Quiet alien is sad

Positive bias
(negative y-intercept)
Quiet alien is happy

More general cases

Mathematical classifier: A score attached to each of the words, plus a bias score that we add to every sentence.

- If the sentence scores positive (or 0), it is classified as happy, and if it scores negative, it is classified as sad.

Geometric classifier: A line that splits two kinds of points in the plane.

- A sentence corresponding to a point above (or over) the line is classified as happy, and one below the line is classified as negative.

More general cases

More general cases

More general cases

- This line could have any equation.
- Since the coordinates are the number of appearances of ‘aack’ and ‘beep’, the equation could be, for example:

1.9 . ‘aack’ - 3.1 . ‘beep’ – 2.2 = 0

More general cases

ALPHABETS WITH 3 WORDS

More general cases

More general cases

More general cases

- Now, how does the equation of a plane look? If the equation of a line looked like a linear combination of the two words plus a bias

2.’aack’+3.’beep’- 2.5 = 0

- Then the equation of a plane looks exactly the same, except with 3 words instead of 2.

2.’aack’-3.’beep’+1.’crack’-25 = 0

More general cases

This classifier above assigns the following scores to the words:

- ‘Aack’: 2 points
- ‘Beep’: -3 points
- ‘Crack’: 1 point.
- Threshold: 2.5 points.

ALPHABETS WITH MANY WORDS

Scores:

- A: 0.1 points
- Aardvark: 1.2 points
- Aargh: -4 points
- …..
- Zygote: 0.4 points.

Threshold:

- 2.3 points.

ALPHABETS WITH MANY WORDS

- Ok, now a question is, how do we come up with these scores and this threshold?
- And even for the simpler classifiers, how do we come up with the scores of ‘aack’ and ‘beep’?
- In other words, how do we draw a line, or a plane, that splits happy and sad points in the best possible way?

How do we determine if a classifier is good or bad? The error function

- The 1 million dollar question here is: How do we find these classifiers?

- We need to find an algorithm, or a sequence of steps, which receives as input a dataset of sentences with ‘happy’ and ‘sad’ labels attached to them, and returns a set of scores for each of the words, plus a threshold.

How do we determine if a classifier is good or bad? The error function

- This will be done in a similar way as we did for regression in lesson three, namely:

1. Start with a set of random scores, and a random threshold.
2. Take a step to improve these scores just a little bit.
3. Repeat step 2 many times.
4. Enjoy your classifier!

How to compare classifiers? The error function

Bad classifier

Good classifier

How to compare classifiers? The error function

ERROR FUNCTION 1: NUMBER OF ERRORS

- Here's an idea, the error of a classifier is the number of points it classifies incorrectly, In summary, the way we calculate this error is as follows:

1. Points that are correctly misclassified produce an error of 0.

2. Points that are misclassified produce an error equal to the distance from that point to the line.

How to compare classifiers? The error function

Bad classifier

Error: 8

Good classifier

Error: 3

How to compare classifiers? The error function

Not badly misclassified

Badly misclassified

How to compare classifiers? The error function

ERROR FUNCTION 2: DISTANCE

- In next figure, what tells the two points apart is that the one in the left is close to the line, while the one in the right is far from the line.

- Therefore, if we simply consider the distance from the point to the line, we have a very good error function.

How to compare classifiers? The error function

Small distance
Small error

Large distance
Large error

How to compare classifiers? The error function

Bad classifier

Error: .....

Good classifier

Error: .....

How to compare classifiers? The error function

- So this is a good error function.
- Is it the one we'll use?
- Unfortunately no, it misses one thing: it is too complex.
- Let's think mathematically, how does one calculate a distance? One uses the Pythagorean theorem.

How to compare classifiers? The error function

ERROR FUNCTION 3: SCORE

- Recall that classifiers can be thought of both geometrically and mathematically.
- In order to construct this error function, we'll use the mathematical definition.
- Let's say that our classifier has the following scores and threshold:

Scores:

1. 'Aack': 1 pt.
2. 'Beep': 2 pts.
3. Bias: -4 pts.

How to compare classifiers? The error function

How to compare classifiers? The error function

Now, let's say that we have two points misclassified by this classifier. They will be the following two sad sentences:

1. Sentence 1: "Aack aack aack beep."

2. Sentence 2: "Aack aack aack aack beep beep beep."

How to compare classifiers? The error function

Not very badly classified point

Very badly classified point

How to compare classifiers? The error function

Furthermore, the scores are the following:

- Sentence 1, coordinates (3,1):
  - **Score =#aack + 2#beep - 4 = 1.**
- Sentence 2, coordinates (4,3):
  - **Score = #aack + 2#beep - 4 = 6**

How to compare classifiers? The error function

What if we have happy points misclassified, would the story be the same? Almost. Let's say we have the following two sentences, that are happy:

- Sentence 3: “”
- Sentence 4: “Aack aack beep.”

How to compare classifiers? The error function

Let's evaluate the scores of these sentences.

- Sentence 3, coordinates (0,0):
  - **Score =#aack + 2#beep - 4 = -4.**

- Sentence 4, coordinates (1,1):
  - **Score = #aack + 2#beep - 4 = -1**

How to compare classifiers? The error function

- In this case, as you can see in this Figure, Sentence 3 is very badly misclassified, while Sentence 4 is misclassified, but not as badly, since it is much closer to the boundary line.

Very badly classified point

Not very badly classified point

How to compare classifiers? The error function

This means, if the point is labelled ‘happy’ and it is misclassified, we should take its error to be the negative of the score, In other words, the error is defined as follows:

- If the point is correctly classified:
  - **Error = 0**
- Else, if the point is incorrectly classified:
  - If the point has a positive label, and the prediction is negative:
    - **Error = -Score**
  - Else, if the point has a negative label, and the prediction is positive:
    - **Error = Score.**

How to compare classifiers? The error function

Perceptron error

- If the point is correctly classified:
  - **Error = 0**
- Else, if the point is incorrectly classified:
  - **Error = |Score|.**

How to compare classifiers? The error function

And we'll compare the following two classifiers:

- **Classifier 1:** #aack + 2#beep - 4
- **Classifier 2:** -#aack + #beep = 0

The points and the classifiers can be seen in the next figure.

How to compare classifiers? The error function

How to compare classifiers? The error function

- We then conclude that Classifier 2 is better than Classifier 1.
- The summary of these calculations is in this figure.

Error = 0 + 2 + 0 + 3 = 5

Error = 0 + 0 + 0 + 0 = 0

How to find a good classifier? The perceptron algorithm

Pseudocode for the perceptron algorithm:

- Begin with a random classifier
- Loop many times:
    - o Improve the algorithm a small amount
- Output a good classifier.

How to find a good classifier? The perceptron algorithm

By now, your head may be full of questions, such as the following:

1. What do we mean by making the classifier a little better?

2. How do we know this algorithm gives us a good classifier?

3. How many times do we run the loop?

The perceptron trick

- If the point is correctly classified, we won't touch the classifier, It's good as it is.
- If the point is misclassified, we will move the line a slight amount towards the point.
- Why? Because if the point is in the wrong side of the line, then we'd actually like the line to move over the point, putting it in the correct side.

Since we only want to make small steps, moving the line slightly towards the point is a good start.

The perceptron trick

The perceptron trick

- Sentence 1 (sad): ‘Aack beep beep beep aack beep beep!’.

- We obtain the following classifier.

Better Classifier 1

- ‘Aack’: 0.98 points
- ‘Beep’: 0.95 points
- Bias: -4.01 points.

The perceptron trick

- Now, what is the score that the Better Classifier gives this sentence? It is

0.98 .#aack + 0.95.# beep - 4.01 = 0.98.2 + 0.95.5 -4.01 = 2.7

- We did it! The Better Classifier has a smaller score on this sentence than the OK Classifier.

- It hasn't improved by much, but imagine doing this step many times, always picking some point in our set.

The perceptron trick

Sentence 2 (happy): ‘Aack.’

- This is again a misclassified point, and let’s see why.
- The sentence has a happy label.
- However, its score is , **1.1+1.0 – 4 = -3** ,as it contains the word ‘aack’ once and the word ‘beep’ zero times.
- Since the score is negative, the classifier predicts that the sentence is sad.

The perceptron trick

Better Classifier 2

- ‘Aack’: 1.01 points
- ‘Beep’: 1 point
- Bias: -3.99 points.

The perceptron trick

Perceptron trick (pseudocode):

Input:

- A classifier with the following scores:
  - Score of 'aack': a.
  - Score of 'beep': b.
  - Bias: c.
- A point with coordinates (x, y) (where x is the number of appearances of the word 'aack', and y of the word 'beep').
- A learning rate η.

Procedure:

- If the point is correctly classified:
  - Output the exact same classifier.
- Else, if the point has a negative label but is misclassified as positive:
  - Output a classifier with the following scores:
    - Score of 'aack': $a - \eta x$.
    - Score of beep: $b - \eta y$.
    - Bias: $c - \eta$.
- Else, if the point has a negative label but is misclassified as positive:
  - Output a classifier with the following scores:
    - Score of 'aack': $a + \eta x$.
    - Score of beep: $b + \eta y$.
    - Bias: $c + \eta$.

Repeating the perceptron trick many times: The perceptron algorithm

- The perceptron trick takes one point, one line, and modifies the line a little bit (or not at all), in order for that line to be a better fit for the point.

- Now we need to use this trick to find the best line that will separate two classes of points.

- How do we do this? Simple, by repeating the perceptron trick many times.

Repeating the perceptron trick many times: The perceptron algorithm

TriveraTech
TECHNOLOGY TRAINING

Repeating the perceptron trick many times: The perceptron algorithm

Repeating the perceptron trick many times: The perceptron algorithm

Repeating the perceptron trick many times: The perceptron algorithm

Perceptron algorithm (pseudocode):

Input:

- A dataset of points, where every point has a positive or negative label.
- A number of epochs, n.
- A learning rate n

Repeating the perceptron trick many times: The perceptron algorithm

Procedure:

- Start with a random line. In other words, start with random values for the score of each word, and the bias.
- Repeat the following procedure n times:
    - Pick a random point.
    - Apply the perceptron trick to the point and the line.
    - In other words, if the point is well classified, do nothing, and if it is misclassified, move the line a little bit closer to the point.

Coding the perceptron algorithm

Aack	Beep	Happy/Sad
1	0	0
0	2	0
1	1	0
1	2	0
1	3	1
2	2	1
3	2	1
2	3	1

Coding the perceptron algorithm

```
import pandas as pd
X = pd.DataFrame([[1,0],[0,2],[1,1],[1,2],[1,3],[2,2],[3,2],[2,3]])
y = pd.Series([0,0,0,0,1,1,1,1])
```

Coding the perceptron algorithm

Coding the perceptron trick

- The features are the number of times each word appears, and they are denoted as follows.
    - Features: x1,x2,……,xn.

- The scores for the words are called the weights.

- They also include the bias, and they are denoted as follows.
    - Weights: w1,w2,…..wn.
    - Bias: b

Coding the perceptron trick

The score for a particular sentence is, just as before, the sum of the number of times each word appears (x_i) times the weight of that word (w_i), plus the bias (b).

- Score = $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = \sum_{i=1}^{n} w_ix_i + b$.

This formula above is known as the *dot product*. We say that the dot product between the vectors $(x_1, x_2, \ldots, x_n)$ and $(w_1, w_2, \ldots, w_n)$is the sum of the products of their entries, namely $w_1x_1 + w_2x_2 + \cdots + w_nx_n$ (if the word vector is not familiar, just think of it as an ordered set of numbers).

Coding the perceptron trick

```
def score(weights, bias, features):
return features.dot(weights) + bias          #A
```

```
def prediction(weights, bias, features):
      if score(weights, bias, features) >= 0:        #B
            return 1
      else:
            return 0
```

Coding the perceptron trick

```
def error(weights, bias, features, label):
    pred = prediction(weights, bias, features)
    if pred == label:          #C
        return 0
    else:                      #D
        return np.abs(score(weights, bias, features))
```

Coding the perceptron trick

- We now write a function that adds the errors of all the points in our dataset.

```
def total_error(weights, bias, X, y):
    total_error = 0
    for i in range(len(X)):                    #E
        total_error += error(weights, bias, X.loc[i], y[i])
    return total_error
```

Coding the perceptron trick

```
def perceptron_trick(weights, bias, features, label, learning_rate = 0.01):
    pred = prediction(weights, bias, features)
    if pred == label:                                   #F
        return weights, bias
    else:
        if label==1 and pred==0:   #G
            for i in range(len(weights)):
                weights[i] += features[i]*learning_rate
            bias += learning_rate
        elif label==0 and pred==1: #H
            for i in range(len(weights)):
                weights[i] -= features[i]*learning_rate
            bias -= learning_rate
    return weights, bias
```

Coding the perceptron trick

```
def perceptron_trick_clever(weights, bias, features, label, learning_rate = 0.01):
    pred = prediction(weights, bias, features)
    for i in range(len(weights)):
        weights[i] += (label-pred)*features[i]*learning_rate
        bias += (label-pred)*learning_rate
    return weights, bias
```

Coding the perceptron trick

```
def perceptron_algorithm(X, y, learning_rate = 0.01, epochs = 200):
    weights = [1.0 for i in range(len(X.loc[0]))]     #I
    bias = 0.0
    errors = []                          #J
    for i in range(epochs):              #K
        errors.append(total_error(weights, bias, X, y))     #L
        j = random.randint(0, len(features)-1)          #M
        weights, bias = perceptron_trick(weights, bias, X.loc[j], y[j])    #N
    return weights, bias, errors
```

Coding the perceptron trick

- Now, let's run the algorithm on our dataset! Below I show you the plots of the data.
- I have removed the plotting lines from here, but you can find them in the repo.

```
perceptron_algorithm(X, y)
```

- The answer I got for the weights and bias was the following:
    - Weight of 'aack': 0.52
    - Weight of 'beep': 0.05
    - Bias: -0.66

Coding the perceptron trick

Coding the perceptron trick

TriveraTech
TECHNOLOGY TRAINING

Applications

- Like linear regression, the perceptron algorithm has many applications in real life.
- Basically any time we try to answer a question with yes or no, where the answer is predicted from previous data, the perceptron algorithm can help us.
- Here are some examples of real life applications of the perceptron algorithm.

Applications of the perceptron algorithm

We can also use other features, for example,

- length of the email,
- size of attachments,
- number of senders,
- if any of our contacts is a sender (categorical variable),
- and many others

Applications of the perceptron algorithm

RECOMMENDATION SYSTEMS

- In many recommendation systems, recommending a video/movie/song/product to a user boils down to a yes/no as to answer.
- In these cases, the question can be any of the following:
  - Will the user click on the video/movie we're recommending?
  - Will the user finish the video/movie we're recommending?
  - Will the user listen to the song we're recommending?
  - Will the user buy the product we're recommending?

Applications of the perceptron algorithm

HEALTHCARE

- Many medical models also use classification algorithms such as the perceptron algorithm to answer questions such as the following:
    - Does the patient suffer from a particular illness?
    - Will a certain treatment work for a patient?

Applications of the perceptron algorithm

COMPUTER VISION

- Classification algorithms such as the perceptron algorithm are widely used in computer vision, more specifically in image recognition.
- Imagine if you have a picture and you want to teach the computer to tell if the picture is of a dog or not.
- You can use classification.

Some drawbacks of the perceptron algorithm, which will be addressed very soon!

- Perceptron are very useful for answering questions such as yes/no, which is good when our labels are one of two categories such as happy/sad, spam/ham, or dog/no-dog.

- However, what if we have more categories? Say we have an image dataset with dogs, cats, and birds.

- What would we do? In general, the approach is to use three perceptron, one for deciding if the image is a dog or not a dog, one for cat, and one for bird.

Some drawbacks of the perceptron algorithm, which will be addressed very soon!

- The solution to the previous two problems is continuous logistic classifiers, or continuous perceptron, which instead of answering a question with a ‘yes’ or ‘no’, they return a score between 0 and 1.

- A sentence with a low score such as 0.1 gets classified as sad, and a sentence with a high score such as 0.9 gets classified as happy.

Summary

- Classification is a very important part of machine learning.
- It is similar to regression in that it consists of training an algorithm with labelled data, and using it to make predictions on future (unlabeled) data, except this time the predictions are categories, such as yes/no, spam/ham, etc.

5: A continuous approach to splitting points: Logistic regression

TriveraTech
TECHNOLOGY TRAINING

A continuous approach to splitting points: Logistic regression

This lesson covers

- The difference between hard assignments and soft assignments.
- Activation functions such as the step function vs the sigmoid function.
- Discrete perceptron vs continuous perceptron.
- The logistic regression algorithm for classifying data.
- Coding the logistic regression algorithm in Python.
- Using the SoftMax function to build classifiers for more than two classes

A continuous approach to splitting points: Logistic regression

- In a nutshell, logistic regression is a type of model which works just like a perceptron, except instead of returning a yes/no answer, it returns a number between 0 and 1.

- In this case, the goal is to assign scores close to 0 to the saddest sentences, scores close to 1 to the happiest sentences, and scores close to 0.5 to neutral sentences.

Logistic Regression (or continuous perceptrons)

A probability approach to classification - The sigmoid function

- The question now is, how do we slightly modify the perceptron models from the previous section in order to get a score for each sentence, as opposed to a simple ‘happy’ or ‘sad’?
- Let’s recall how we made the predictions in the perceptron models.
- We scored each sentence by separately scoring each word and adding the scores, plus the bias.

A probability approach to classification - The sigmoid function

A probability approach to classification - The sigmoid function

- Many functions can help us here, and in this case, we'll use one called the sigmoid, denoted with the Greek letter σ The formula for the sigmoid is the following:

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

A probability approach to classification - The sigmoid function

A probability approach to classification - The sigmoid function

- The code for the sigmoid function in Python is very simple; we make use of the numpy function exp, which takes any real number as an input, and returns e to the power of that number.

```
import numpy as np
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

A probability approach to classification - The sigmoid function

x	σ(x)
-5	0.0067
-1	o.269
0	0.5
1	0.731
5	0.9933

A probability approach to classification - The sigmoid function

- Now we are ready to define a prediction.

- The prediction is obtained by applying the sigmoid function to the score, and it returns a number between 0 and 1 which, as I mentioned before, can be interpreted in our example as the probability that the sentence is happy.

- Here's the code for the prediction function

```
def lr_prediction(weights, bias, features):
    return sigmoid(score(weights, bias, features))
```

The error functions - Absolute, square, and log loss

In this section we cook up some error functions for a continuous perceptron classifier, But first let's stop and think, what properties would we like a good error function to have? Here are some I can think of:

- If a point is correctly classified, the error is a small number.
- If a point is incorrectly classified, the error is a large number.
- The error of a classifier at a set of points is the sum of errors at each point.

The error functions - Absolute, square, and log loss

The classifier then makes a prediction between 0 and 1 for each point in the plane, as follows:

- The points on the line are given a prediction of 0.5.
- Points that are up and to the right of the line are given predictions higher than 0.5, and the farther a point is from the line in that direction, the closer its prediction is to 1.
- Points that are down and to the left of the line are given predictions lower than 0.5, and the farther a point is from the line in that direction, the closer its prediction is to 0.

The error functions - Absolute, square, and log loss

Point	True label	Predicate label	Error
1	1(Happy)	0.95	Should be small
2	0(Sad)	0.8	Should be large
3	1(Happy)	0.3	Should be large
4	0(Sad)	0.1	Should be small

The error functions - Absolute, square, and log loss

The error functions - Absolute, square, and log loss

ERROR FUNCTION 1:

- **ABSOLUTE ERROR** The absolute error is very similar to the absolute error we defined for linear regression in lesson 3.

The error functions - Absolute, square, and log loss

ERROR FUNCTION 2:

- **SQUARE ERROR** Again, just like in linear regression, we also have the square error.

- This is the square of the difference between the prediction and the label, and it works for the same reason that the absolute error works.

The error functions - Absolute, square, and log loss

Point	True label	Predicate label	Absolute Error	Square Error
1	1(Happy)	0.95	0.05	0.0025
2	0(Sad)	0.8	0.8	0.64
3	1(Happy)	0.3	0.7	0.49
4	0(Sad)	0.1	0.1	0.01

The error functions - Absolute, square, and log loss

ERROR FUNCTION 3:

- LOG LOSS Notice that with the absolute and square error functions, points that are vastly misclassified have large errors, but never too large.
- Let’s look at an example: a point with label 1 but that the classifier has assigned a prediction of 0.01.
- This point is vastly misclassified, since we would hope that the classifier assigns it a prediction close to 1.

The error functions - Absolute, square, and log loss

- Point 1:
  - Label = 1 (happy)
  - Prediction = 0.95
  - Probability of being happy: 0.95
- Point 2:
  - Label = 1 (sad)
  - Prediction = 0.8
  - Probability of being sad: 1 - 0.8 = 0.2
- Point 3:
  - Label = 1 (happy)
  - Prediction = 0.3
  - Probability of being happy: 0.3
- Point 4:
  - Label = 1 (sad)
  - Prediction = 0.1
  - Probability of being happy: 1 - 0.1 = 0.9

The error functions - Absolute, square, and log loss

The error functions - Absolute, square, and log loss

- They look familiar, don't they? All we need to do is flip the function by multiplying it by -1.
- Thus, we obtain our log loss error function. It is precisely the negative natural logarithm of the probability of the point being its label.

log loss = -ln(Probability of a point being its label).

- Now let's calculate it for our four data points.

The error functions - Absolute, square, and log loss

Point	True label	Predicate label	Probability of being its label	Log loss
1	1(Happy)	0.95	0.95	-ln(0.95)=0.051
2	0(Sad)	0.8	0.2	-ln(0.2)=1.609
3	1(Happy)	0.3	0.3	-ln(0.4)=0.916
4	0(Sad)	0.1	0.9	-ln(0.9)=0.105

The error functions - Absolute, square, and log loss

We can write the log loss as following:

- If the label is 0:
  - **log loss = -ln(1 - prediction)**
- If the label is 1:
  - **log loss = -ln(prediction).**

As a matter of fact, we can make the formula even nicer. Check this out:

$$\textbf{log loss} = -\textbf{label} \times \ln(\textbf{prediction}) - (1 - \textbf{label}) \times \ln(1 - \textbf{prediction})$$

The error functions - Absolute, square, and log loss

COMPARING THE ERROR FUNCTIONS

Point	True label	Predicate label	Absolute Error	Square Error	Log loss
1	1(Happy)	0.95	0.05	0.0025	0.051
2	0(Sad)	0.8	0.8	0.64	1.609
3	1(Happy)	0.3	0.7	0.49	0.916
4	0(Sad)	0.1	0.1	0.01	0.105

The error functions - Absolute, square, and log loss

- If I haven't convinced you of the power of the log loss error function, let's look at an extreme point.
- Let's say we have a point with label 1 (happy), for which the classifier makes a prediction of 0.00001.
- This point is very poorly classified.

More on the log loss error function

- I made a case for why I prefer the log loss error function over the absolute and square error functions.

- In this section I give you another reason, a reason that has to do with independent probabilities. We've calculated the log loss for one point.

- The total log loss for a classifier is defined as the sum of the log loss for every point in the dataset.

More on the log loss error function

More on the log loss error function

More on the log loss error function

To summarize, these are the steps for calculating the log loss:

- For each point, we calculate the probability that the classifier predicts for its label (happy or sad).
- We multiply all these probabilities to obtain the total probability that the classifier has given to these labels.
- We apply the natural logarithm to that total probability.

More on the log loss error function

$$-\ln(P(\text{all})) = -\ln(0.7) - \ln(0.3) - \ln(0.4) - \ln(0.6)$$

$$\text{Log-loss} = 0.357 + 1.204 + 0.916 + 0.511$$

$$= 2.988$$

$$\ln(P(\text{all})) = \ln(0.7) + \ln(0.6) + \ln(0.7) + \ln(0.6)$$

$$\text{Log-loss} = 0.357 + 0.511 + 0.357 + 0.511$$

$$= 1.735$$

Reducing the log loss error: The logistic regression trick

The perceptron trick had the following steps:

1. If the point was correctly classified, it would not move the line.

2. If the point was incorrectly classified, it would move the line slightly towards the point.

Reducing the log loss error: The logistic regression trick

The logistic regression trick has the following steps:

- If the point is correctly classified, it moves the line slightly away from the point.
- If the point is incorrectly classified, it moves the line slightly towards the point.

Reducing the log loss error: The logistic regression trick

An example with a discrete perceptron & continuous perceptron

- Classifier (scores):
  - o **‘Aack’: 1 pt.**
  - o **‘Beep’: 1 pts.**
  - o **Bias: -4 pts**.

- Sentence 1:
  - o **Words**: ‘Aack beep beep beep aack beep beep!”
  - o **Label**: Sad

An example with a discrete perceptron & continuous perceptron

- This algorithm would calculate the score of the sentence, which is the sum of the scores of the words times the number of times each word appears in the sentence, plus the bias:

Score = (score of aack)*#aack + (score of beep)*#beep - 4
= 1*2 + 1*4 - 4
= 2

An example with a discrete perceptron & continuous perceptron

If the error rate was 0.01, we did the following:

- Update the score of ‘aack’ by subtracting **2*0.01**, thus obtaining 0.98.
- Update the score of ‘beep’ by subtracting **4*0.01**, thus obtaining 0.96.
- Update the bias by subtracting 0.01, thus obtaining -4.01.

An example with a discrete perceptron & continuous perceptron

- The new classifier would assign a score of

$$0.98*2 + 0.96*4 - 4.01 = 1.79.$$

An example with a discrete perceptron & continuous perceptron

Using the logistic regression algorithm to improve our discrete perceptron classifier

- What would happen with a continuous perceptron classifier?
- This one would apply the sigmoid function to the score, and use that as a prediction.
- The sigmoid function gives us

Prediction = σ(2) = 0.881.

An example with a discrete perceptron & continuous perceptron

- Update the score of ‘aack’ by adding **2*0.01*(-0.881),** thus obtaining 0.982.
- Update the score of ‘beep’ by adding **4*0.01*(-0.881),** thus obtaining 0.965.
- Update the bias by adding **0.01*(-0.881),** thus obtaining -4.009.

An example with a discrete perceptron & continuous perceptron

- The new classifier will give the point the following score and prediction.

$$\text{Score} = 0.982*2 + 0.965*4 - 4.009 = 1.815$$
$$\text{Prediction} = \sigma(1.815) = 0.86$$

A second example with a discrete perceptron and a continuous perceptron

- One of the main benefits of the logistic regression algorithm over the perceptron algorithm is that if a point is correctly classified

- The perceptron algorithm leaves it alone and doesn't improve, while the logistic regression algorithm still uses the information of that point to improve the model.

A second example with a discrete perceptron and a continuous perceptron

Let's update the scores in the exact same way as before.

- Update the score of 'aack' by adding **2*0.01*(0.119),** thus obtaining 1.002.
- Update the score of 'beep' by adding **4*0.01*(0.119),** thus obtaining 1.005.
- Update the bias by adding **0.01*(0.119),** thus obtaining -3.999.

A second example with a discrete perceptron and a continuous perceptron

- The score and prediction that this new classifier gives to our point is:

$$\textbf{Score} = 1.002*2 + 1.005*4 - 3.999 = 2.025$$
$$\textbf{Prediction} = \sigma(2.025) = 0.883$$

Moving the line to fit the points - The logistic regression algorithm

Logistic regression trick (pseudocode):

Input:

- A classifier with the following scores:
  - o Score of ‘aack’: a.
  - o Score of ‘beep’: b.
  - o Bias: c.
- A point with coordinates (x1, x2) (where x1 is the number of appearances of the word
- ‘aack’, and x2 of the word ‘beep’).
- A learning rate η

Moving the line to fit the points - The logistic regression algorithm

Procedure :

- Calculate the prediction that the classifier gives to the datapoint as:
  - **$\hat{y} = \sigma(ax1 + bx2 + c)$**

- Output a classifier with the following scores:
  - **Score of 'aack': $a + \eta\,(y - \hat{y})x1$**
  - **Score of beep: $b + \eta(y - \hat{y})x2$**
  - **Bias: $c + \eta$**

Moving the line to fit the points - The logistic regression algorithm

Logistic regression algorithm (pseudocode):

Input:

- A dataset of points, where every point has a positive or negative label.
- A number of epochs, n.
- A learning rate η

Moving the line to fit the points - The logistic regression algorithm

Procedure:

- Start with a random line. In other words, start with random values for the score of each word, and the bias.
- Repeat the following procedure n times:
    - o Pick a random point.
    - o Apply the logistic regression trick to the point and the line.

Coding the logistic regression algorithm

These weights include the bias, which corresponds to no word, it simply gets added to the score of every sentence.

- Features: x1, x2, … , xn
- Label: y

- Weights: w1, w2, … , wn
- Bias: b

Coding the logistic regression algorithm

The score for a particular sentence is the sigmoid of the sum of the weight of each word (wi) times the number of times that appears (xi), plus the bias (b) (which we called the dot product).

- Prediction: $: \hat{y} = \sigma(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b) = \sigma(\sum_{i=1}^{n} w_ix_i + b).$

Coding the logistic regression algorithm

```
def sigmoid(x):
    return np.exp(x)/(1+np.exp(x))

def lr_prediction(weights, bias, features):
return sigmoid(score(weights, bias, features))
```

- Now that we have the prediction, we can proceed to the log loss.
- We need to do a bit of math to figure out the formula for the log loss.

Coding the logistic regression algorithm

Since we called the label y and the prediction ŷ, we get the following:

- If the label is **y = 1**, then **log loss = ln (ŷ)**
- If the label is **y = 0**, then **log loss = ln (1 − ŷ)**

Note that we can encode these two easily into one formula:
log loss = y ln (ŷ) + (1 – y) ln (1 − ŷ)

Coding the logistic regression algorithm

- Let's code that formula.

```
def log_loss(weights, bias, features, label):
    pred = prediction(weights, bias, features)
    return label*np.log(prediction) + (1-label)*np.log(1-prediction)
```

- We need the log loss over the whole dataset, so we can add over all the data points.

```
def total_log_loss(weights, bias, X, y):
    total_error = 0
    for i in range(len(X)):
        total_error += log_loss(weights, bias, X.loc[i], y[i])
    return total_error
```

Coding the logistic regression algorithm

```
def lr_trick(weights, bias, features, label, learning_rate = 0.01):
    pred = lr_prediction(weights, bias, features)
    for i in range(len(weights)):
        weights[i] += (label-pred)*features[i]*learning_rate
        bias += (label-pred)*learning_rate
    return weights, bias

def lr_algorithm(features, labels, learning_rate = 0.01, epochs = 200):
    weights = [1.0 for i in range(len(features.loc[0]))]
    bias = 0.0
    errors = []
    for i in range(epochs):
        draw_line(weights[0], weights[1], bias, color='grey', linewidth=1.0,
       linestyle='dotted')
        errors.append(total_error(weights, bias, features, labels))
        j = random.randint(0, len(features)-1)
        weights, bias = perceptron_trick(weights, bias, features.loc[j], labels[j])
    draw_line(weights[0], weights[1], bias)
```

Coding the logistic regression algorithm

```
plot_points(features, labels)
plt.show()
plt.scatter(range(epochs), errors)
plt.xlabel('epochs')
plt.ylabel('error')
return weights, bias
```

- The code for loading our small dataset is below, and the plot of the dataset is in next figure.

```
import pandas as pd
X = pd.DataFrame([[1,0],[0,2],[1,1],[1,2],[1,3],[2,2],[3,2],[2,3]])
y = pd.Series([0,0,0,0,1,1,1,1])
```

Coding the logistic regression algorithm

Coding the logistic regression algorithm

- We run the algorithm to get the classifier.

```
lr_algorithm(features, labels)
([0.46999999999999953, 0.09999999999999937], -0.6800000000000004)
```

The classifier we obtain has the following weights and biases.

- w1 = 4.7
- w2 = 0.1
- b = −0.6

Coding the logistic regression algorithm

Coding the logistic regression algorithm

The logistic regression algorithm in Turi Create

```
import turicreate as tc
data = tc.SFrame({'X1': X[0], 'X2': X[1], 'y': y})
```

X1	X2	y
1	0	0
0	2	0
1	1	0
1	2	0
1	3	1
2	2	1
3	2	1
2	3	1

[8 rows x 3 columns]

The logistic regression algorithm in Turi Create

- Now, we train the classifier.

```
classifier = tc.logistic_classifier.create(data,
                                           features = ['X1', 'X2'],
                                           target = 'y',
                                           validation_set=None)
```

- First, let's evaluate the coefficients of this model, with the following command.

classifier.coefficients

The logistic regression algorithm in Turi Create

We get the following coefficients:

- W1=2.97
- W2=2.5
- b=-8.96(the intercept)

The logistic regression algorithm in Turi Create

Classifying into multiple classes - The softmax function

- If we had different labels, such as, for example, dog and cat, we could still use a perceptron.

- The perceptron would return a number between 0 and 1, which can be interpreted as the possibility that the data point is classified as a dog.

- To find the probability that the data point is classified as a cat, we simply take 1 minus the probability that the data point is classified as a dog.

Classifying into multiple classes - The softmax function

As an example, say we have our three classifiers: a dog classifier, a cat classifier, and a bird classifier. For a particular data point, they all output scores in the following way:

- Dog classifier: 4
- Cat classifier: 2
- Bird classifier: 1

Classifying into multiple classes - The softmax function

This works, but what if we have the following numbers:

- Dog classifier: 2
- Cat classifier: 0
- Bird classifier: -2

Classifying into multiple classes - The softmax function

So we'll go with ex. We apply it to all the scores, to get the following.

- Dog classifier: **e2 = 7.389**
- Cat classifier: **e0 = 1**
- Bird classifier: **e-2 = 0.135**

Classifying into multiple classes - The softmax function

Now, we do what we did before, we normalize, or divide by the sum of these three numbers in order for them to add to 1. The sum is **7.389 + 1 + 0.135 = 8.524**, so we get the following:

- Probability of dog: **7.389/8.524 = 0.867**
- Probability of cat: **1/8.524 = 0.117**
- Probability of bird: **0.135/8.524 = 0..016**

Classifying into multiple classes - The softmax function

- If we have n classifiers which output the n scores **a1, a2, … , an**, the probabilities obtained are p1, p2, … , pn, where

$$p_i = \frac{e^{a_i}}{e^{a_1} + e^{a_2} + \cdots + e^{a_n}}.$$

- This formula is known as the SoftMax formula.

Summary

- The log loss is an error function for continuous perceptron's.
- It is calculated separately for every point as the natural logarithm of the probability that the point is classified correctly according to its label.
- The total log loss of a classifier on a dataset is the sum of the log loss at every point.

6: Using probability to its maximum: The naive Bayes algorithm

TriveraTech
TECHNOLOGY TRAINING

The naive Bayes algorithm

This lesson covers

- What is Bayes theorem?
- When are events dependent of independent?
- The prior and the posterior probabilities.
- Calculating conditional probabilities based on events.
- What is the naive Bayes algorithm?
- Using the naive Bayes algorithm to predict if an email is spam or ham, based on the words in the email.
- Coding the naive Bayes algorithm in Python.

The naive Bayes algorithm

- Bayes Theorem is a fundamental theorem in probability and statistics, and it helps us calculate probabilities.
- The more we know about a certain situation, the more accurate the probability is.
- For example, let’s say we want to find the probability that it will snow today.
- If we have no information of where we are and what time of the year it is, we can only come up with a vague estimate.

Sick or healthy? A story with Bayes Theorem

Consider the following scenario. Your (slightly hypochondriac) friend calls you, and the following conversation unfolds:

- **You:** Hello!
- **Friend:** Hi, I have some terrible news!
- **You:** Oh no, what is it?
- **Friend:** I heard about this terrible and rare disease, and I went to the doctor to be tested for it. The doctor said she would administer a very accurate test. Then today she called me and told me that I tested positive! I must have the disease!

Sick or healthy? A story with Bayes Theorem

You: First let's calm down, mistakes happen in medicine. Let's try to see how likely it is that you actually have the disease. How accurate did the doctor say the test was?

Friend: She said it was 99% accurate. That means I'm 99% likely to have the disease!

You: Wait, let's look at all the numbers. How likely is it to have the disease, regardless of the test? How many people have the disease?

Sick or healthy? A story with Bayes Theorem

Friend: I was reading online, and it says that on average, 1 out of every 10,000 people have the disease.

You: Ok, let me get a piece of paper (*puts friend on hold*).

Sick or healthy? A story with Bayes Theorem

- Let’s do some rough calculations to see what the probability would be.
- Let’s pick a random group of 10,000 people, Since on average, one out of every 10,000 people are sick, then we expect one of these people to have the disease.
- Now, let’s run the test on the remaining 9,999 healthy ones.
- Since the test makes mistakes 1% of the time, we expect 1% of these 9,999 healthy people to be misdiagnosed as sick.

Sick or healthy? A story with Bayes Theorem

- **You:** Don’t worry, based on the numbers you gave me, the probability that you have the disease given that you tested positive is actually only around 1%!
- **Friend:** Oh my God, really? That’s such a relief, thank you!
- **You:** Don’t thank me, thank math (winks eye).

Sick or healthy? A story with Bayes Theorem

The facts that we have so far are the following:

1. Out of every 10,000 people, 1 has the disease.

2. Out of every 100 sick people who take the test, 99 test positive, and one tests negative.

3. Out of every 100 healthy people who take the test, 99 test negative, and one tests positive.

Sick or healthy? A story with Bayes Theorem

So we have our first piece of information out of 1 million people:

- 999,900 of them are healthy, and
- 100 of them are sick.

Sick or healthy? A story with Bayes Theorem

Out of 99,900 healthy people:

- 99% of them, or 989,901, are diagnosed as healthy, and
- 1% of them, or 9,999, are diagnosed as sick.

Out of 100 sick people:

- 99% of them, or 99, are diagnosed as sick, and
- 1& of them, or 1, is diagnosed as healthy.

Sick or healthy? A story with Bayes Theorem

Sick or healthy? A story with Bayes Theorem

- This is a bit surprising, if the test is correct 99% of the time, why on earth was it so wrong?
- Well, the test is not bad, if it’s only wrong 1% of the time.
- But since one person out of every 10,000 is sick with the disease, that means a person is sick 0.01% of the time.

Sick or healthy? A story with Bayes Theorem

Sick or healthy? A story with Bayes Theorem

Notice that in the tree in previous slode, the patients get divided into four groups:

1. Sick patients who were diagnosed sick.
2. Sick patients who were diagnosed healthy.
3. Healthy patients who were diagnosed sick.
4. Healthy patients who were diagnosed healthy.

Sick or healthy? A story with Bayes Theorem

Sick or healthy? A story with Bayes Theorem

- From previous tree, we can again see that the probability that since among the patients that are diagnosed sick, 99 are actually sick, and 9,999 are healthy, from which we again can deduce that the probability that our friend is sick is:

- Probability of being sick when being diagnosed sick

$$\frac{99}{99 + 9{,}999} = 0.0098.$$

Prelude to Bayes Theorem: The prior, the event, and the posterior

PRIOR The initial probability that we calculate.

EVENT What gives us information to calculate better probabilities.

POSTERIOR The final (and more accurate) probability that we calculate using the prior probability and the event.

Prelude to Bayes Theorem: The prior, the event, and the posterior

Prelude to Bayes Theorem: The prior, the event, and the posterior

- Bayes Theorem is one of the most important building blocks of probability and of machine learning.

- It is so important that and several fields are named after it, Bayesian learning, Bayesian statistics, Bayesian analysis, and so on.

Use-case: Spam detection model

- Now consider the following situation, you are reading your email, and you are sick and tired of receiving spam (garbage) email.

- You would like to build a tool that helps you separate spam from non-spam emails

Finding the prior: The probability that any email is spam

- What is the probability that an email is spam? That is a hard question, but let's try to make an estimate.
- We look at our current inbox and count how many emails are spam and ham.
- Say there are 100 emails, of which 20 are spam, and 80 ham. Thus, 20% of the emails are spam.

Finding the prior: The probability that any email is spam

20 spam

80 ham

Probability of spam: 20%

Finding the prior: The probability that any email is spam

What the math just happened? Turning ratios into probabilities

- One way to visualize the previous example is with a tree of all the four possibilities, namely: that the email is spam or ham, and that it contains the word ‘lottery’ or not.

- We draw it in the following way, we start by the root, which splits into two branches.

- The top branch corresponds to spam, and the bottom branch corresponds to ham.

What the math just happened? Turning ratios into probabilities

What the math just happened? Turning ratios into probabilities

What the math just happened? Turning ratios into probabilities

Many times, we don't know how many emails are spam or ham, all we know is the following:

- The probability that an email is spam is **⅕.**
- The probability that a spam email contains the word 'lottery' is **¾.**
- The probability that a ham email contains the word 'lottery' is **1/40**.

What the math just happened? Turning ratios into probabilities

RULE OF COMPLEMENTARY PROBABILITIES For an event E, the complement of the event E, denoted E^c, is the event opposite to E. The probability of E^c is 1 minus the probability of E, namely,

$$P(E^c) = 1 - P(E).$$

So we have the following:

- *P(spam)*= $\frac{1}{5}$.The probability of an email being spam.
- *P(ham)*= $\frac{4}{5}$. The probability of an email being ham.

What the math just happened? Turning ratios into probabilities

- Now, let's look at the other information, The probability that a spam email contains the word 'lottery' is ¾.

- This can be read as, the probability that an email contains the word 'lottery' given that it is spam, is ¾, This is a conditional probability.

- The condition is that the email is spam, We denote condition by a vertical bar, so this can be written as P ('lottery'|spam). Now, what would P ('lottery'|spam),

What the math just happened? Turning ratios into probabilities

- **P ('lottery'|spam) =3/4** The probability that a spam email contains the word 'lottery'.
- **P ('no lottery'|spam) =1/4** The probability that a spam email does not contain the word 'lottery'.
- **P ('lottery'|ham) =1/6** The probability that a ham email contains the word 'lottery'.
- **P (no 'lottery'|ham) =15/6** The probability that a ham email does not contain the word 'lottery'.

What the math just happened? Turning ratios into probabilities

These events are called intersections of events, and denoted with the symbol, thus, we need to find the following probabilities:

- P(’lottery∩spam)
- P(no ’lottery∩spam)
- P(’lottery∩ham)
- P(no ’lottery∩ham)

What the math just happened? Turning ratios into probabilities

- We multiplied the probability that an email is spam times the probability that a spam email contains the word ‘lottery’, to obtain the probability that an email is spam and contains the word lottery.

What the math just happened? Turning ratios into probabilities

PRODUCT RULE OF PROBABILITIES For events E and F, the probability of their intersection is precisely

$$P(E \cap F) = P(E|F).\ P(F).$$

What the math just happened? Turning ratios into probabilities

- So now we can calculate these probabilities:

  - $P('lottery' \cap spam) = P('lottery' \mid spam) \cdot P(spam) = \frac{1}{5} \cdot \frac{3}{4} = \frac{3}{20}$
  - $P(no\ 'lottery' \cap spam) = P(no\ 'lottery' \mid spam) \cdot P(spam) = \frac{1}{5} \cdot \frac{1}{4} = \frac{1}{20}$
  - $P('lottery' \cap ham) = P('lottery' \mid ham) \cdot P(ham) = \frac{4}{5} \cdot \frac{1}{16} = \frac{1}{20}$
  - $P(no\ 'lottery' \cap ham) = P(no\ 'lottery' \mid ham) \cdot P(ham) = \frac{4}{5} \cdot \frac{15}{16} = \frac{15}{20}.$

What the math just happened? Turning ratios into probabilities

What the math just happened? Turning ratios into probabilities

- Among the four events we just studied, in only two of them does the word ‘lottery’ appear, Thus, we only need to consider those, namely:

  - $P('lottery' \cap spam) = \frac{3}{20}$
  - $P('lottery' \cap ham) = \frac{1}{20}$.

What the math just happened? Turning ratios into probabilities

What the math just happened? Turning ratios into probabilities

In this case, the numbers become $\frac{3/20}{3/20 + 1/20}$ and $\frac{1/20}{3/20 + 1/20}$. These simplify to ¾ and ¼, which are the desired probabilities. Thus, we conclude that

- $P(spam \mid 'lottery') = \frac{3}{4} = 75\%$
- $P(ham \mid 'lottery') = \frac{1}{4} = 25\%.$

What the math just happened? Turning ratios into probabilities

- This is the same thing as dividing each one of them by their sum, In math terms, we did the following:

$$P(spam \mid 'lottery') = \frac{P('lottery' \cap spam)}{P('lottery' \cap spam) + P('lottery' \cap ham)}.$$

- If we remember what these two probabilities were, using the multiplication rule, we get the following:

$$P(spam \mid 'lottery') = \frac{P('lottery' \mid spam) \cdot P(spam)}{P('lottery' \mid spam) \cdot P(spam) + P('lottery' \mid ham) \cdot P(ham)}.$$

What the math just happened? Turning ratios into probabilities

To verify, we plug in the numbers to get:

$$P(spam \mid 'lottery') = \frac{\frac{1}{5}\cdot\frac{3}{4}}{\frac{1}{5}\cdot\frac{3}{4}+\frac{4}{5}\cdot\frac{1}{16}} = \frac{\frac{3}{20}}{\frac{3}{20}+\frac{1}{20}} = \frac{\frac{3}{20}}{\frac{4}{20}} = \frac{3}{4}.$$

And this is the formula for Bayes Theorem! More formally:

BAYES THEOREM For events E and F,

$$P(E \mid F) = \frac{P(F \mid E)\cdot P(E)}{P(F \mid E)\cdot P(E) + P(F \mid E^C)\cdot P(E^C)}.$$

What about two words? The naive Bayes algorithm

- Okay, this can't be the end of it, right? There are a lot more words in the dictionary.
- Let's say we notice that another word, the word 'sale' also tends to appear a lot in spam email.
- We can do a similar calculation. Let's count how many spam and ham emails contain the word 'sale'.

What about two words? The naive Bayes algorithm

Probability of spam given the word ‘sale’: 60%

6 spam emails containing ‘sale’

4 ham emails containing ‘sale’

What about two words? The naive Bayes algorithm

- For now, let's simplify things by only using the two words 'lottery' and 'sale'. So here's an idea (a bad idea), let's look at all the emails that have both words, 'lottery' and 'sale'.

- Count how many of them are spam, how many of them are ham, and find the probability based on those two numbers using Bayes theorem. Tada!

What about two words? The naive Bayes algorithm

- So what's the solution here, should we gather more data? That is always a good idea, but many times we can't, so we have to deal with the data we have.

- In some way, we can't rely on the emails that contain both words 'lottery' and 'sale'.

What about two words? The naive Bayes algorithm

- What our naive assumption really means is that the event of containing the words ‘lottery’ and ‘sale’ are independent.

- This is, the appearance of one of the words in no way affects the appearance of the other one.

- This means, if 10% of the emails contain the word ‘sale’, then we can assume that if we restrict to the emails that contain the word ‘lottery’, 10% of those emails also contain the word ‘sale’.

What about two words? The naive Bayes algorithm

What about two words? The naive Bayes algorithm

PRODUCT RULE FOR INDEPENDENT PROBABILITIES

- If two events and are independent, namely, the occurrence of one doesn't influence in any way the occurrence of the other one, then the probability of both of them happening is the product of the probabilities of each of the events.
- In other words.

$$P(E \cap F) = P(E).P(F)$$

What about two words? The naive Bayes algorithm

Spam emails:

- Among the 20 spam emails, 15 of them contained the word ‘lottery’.
- Therefore, the probability of the word ‘lottery appearing in a spam email is 15/20, or 0.75.
- Among the 20 spam emails, 6 of them contained the word ‘sale’.
- Therefore, the probability of the word sale appearing in a spam email is 6/20, or 03.

What about two words? The naive Bayes algorithm

- In probability terms, our assumption boils down to the following:
  - $P(lottery \mid spam) = \frac{3}{4}$
  - $P(sale \mid spam) = \frac{3}{10}$
  - $P(lottery \cap sale \mid spam) = \frac{3}{4} \cdot \frac{3}{10} = \frac{9}{40} = 0.225.$

What about two words? The naive Bayes algorithm

Ham emails:

- Among the 80 ham emails, 5 of them contained the word ‘lottery’.
- Therefore, the probability of the word ‘lottery appearing in a ham email is 5/80, or 0.0625.
- Among the 80 ham emails, 4 of them contained the word ‘sale’.
- Therefore, the probability of the word sale appearing in a ham email is 4/80, or 0.05.

What about two words? The naive Bayes algorithm

- In probability terms, our assumption is the following:

  - $P(lottery \mid ham) = \frac{5}{80} = \frac{1}{16}$
  - $P(sale \mid ham) = \frac{4}{80} = \frac{1}{20}$
  - $P(lottery \cap sale \mid ham) = \frac{1}{16} \cdot \frac{1}{20} = \frac{1}{320} = 0.0625.$

What about two words? The naive Bayes algorithm

4.5 spam emails
containing 'lottery and sale'

0.0625 ham emails
containing lottery and 'sale'

What we did here, using probability, was using Bayes Theorem, except with the events

- $E = lottery \cap sale$
- $F = spam$,

to get the formula

$$P(spam \mid lottery \cap sale) = \frac{P(lottery \cap sale \mid spam) \cdot P(spam)}{P(lottery \cap sale \mid spam) \cdot P(spam) + P(lottery \cap sale \mid ham) \cdot P(ham)}.$$

What about two words? The naive Bayes algorithm

$$P(lottery \cap sale \mid spam) = P(lottery \mid spam) \cdot P(sale \mid spam),$$

$$P(lottery \cap sale \mid ham) = P(lottery \mid ham) \cdot P(sale \mid ham).$$

Plugging them into the previous formula, we get

$$P(spam \mid lottery \cap sale)$$
$$= \frac{P(lottery \mid spam) \cdot P(sale \mid spam) \cdot P(spam)}{P(lottery \mid spam) \cdot P(sale \mid spam) \cdot P(spam) + P(lottery \mid ham) \cdot P(sale \mid ham) \cdot P(ham)}.$$

What about two words? The naive Bayes algorithm

- $P(lottery \mid spam) = \frac{3}{4}$
- $P(sale \mid spam) = \frac{3}{10}$
- $P(spam) = \frac{1}{5}$
- $P(lottery \mid ham) = \frac{1}{16}$
- $P(sale \mid ham) = \frac{1}{20}$
- $P(ham) = \frac{4}{5}$

we get

$$P(spam \mid lottery \cap sale) = \frac{\frac{3}{4} \cdot \frac{3}{10} \cdot \frac{1}{5}}{\frac{3}{4} \cdot \frac{3}{10} \cdot \frac{1}{5} + \frac{1}{16} \cdot \frac{1}{20} \cdot \frac{4}{5}} = 0.9863.$$

What about more than two words?

Among the spam emails:

- The probability of an email containing ‘lottery’ is 15/20.
- The probability of an email containing ‘sale’ is 6/20.
- The probability of an email containing ‘mom’ is 1/20.

What about more than two words?

Among the ham emails:

- The probability of an email containing ‘lottery’ is 5/80.
- The probability of an email containing ‘sale’ is 4/80.
- The probability of an email containing ‘mom’ is 40/80.

Building a spam detection model with real data

- In the repo, we have processed the dataset as a Pandas Data Frame with the following command:

```
import pandas
emails = pandas.read_csv('emails.csv')
```

- If we look at the first 10 rows of this dataset, this is how it looks .

Building a spam detection model with real data

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1
5	Subject: great nnews hello , welcome to medzo...	1
6	Subject: here ' s a hot play in motion homela...	1
7	Subject: save your money buy getting this thin...	1
8	Subject: undeliverable : home based business f...	1
9	Subject: save your money buy getting this thin...	1

Data preprocessing

- Let’s start by turning the text string into a list of words.
- This is done in the following function, which uses the split() function.
- Since we only check if each word appears in the email or not, regardless of how many times it appears, we turn it into a set, and then into a list again.

```
def process_email(text):
    return list(set(text.split()))
```

Data preprocessing

- Now we use the apply() function to apply this change to the entire column.
- We call the new column emails[‘words’].

```
emails['words'] = emails['text'].apply(process_email)
```

Data preprocessing

	text	spam	words
0	Subject: naturally irresistible your corporate...	1	[all, through, portfolio, its, guaranteed, ,, ...
1	Subject: the stock trading gunslinger fanny i...	1	[and, merrill, is, nameable, clockwork, libret...
2	Subject: unbelievable new homes made easy im ...	1	[pre, and, all, show, being, visit, loan, 454,...
3	Subject: 4 color printing special request add...	1	[and, golden, 5110, 626, color, ca, an, canyon...
4	Subject: do not have money , get software cds ...	1	[comedies, all, old, tradgedies, be, money, is...
5	Subject: great nnews hello , welcome to medzo...	1	[va, groundsel, allusion, ag, tosher, confide,...
6	Subject: here ' s a hot play in motion homela...	1	[precise, all, chain, limited, indicating, ena...
7	Subject: save your money buy getting this thin...	1	[right, want, just, money, is, within, it, rea...
8	Subject: undeliverable : home based business f...	1	[unknown, grownups, co, telecom, is, mts, 000,...
9	Subject: save your money buy getting this thin...	1	[right, want, just, money, is, within, it, rea...

Finding the priors

- Let's first find out the probability that an email is spam (the prior).
- For this, we calculate the number of emails that are spam, and divide it by the total number of emails.
- Notice that the number of emails that are spam is simply the sum of entries in the 'spam' column.
- The following line will do the job.

```
sum(emails['spam'])/len(emails)
0.2388268156424581
```

Finding the posteriors with Bayes theorem

```
model = {}

for email in emails:
    for word in email['words']:
        if word not in model:
            model[word] = {'spam': 1, 'ham': 1}
        if word in model:
            if email['spam']:
                model[word]['spam'] += 1
            else:
                model[word]['ham'] += 1
```

Let's examine some rows of the dictionary:

```
model
{'woods': {'ham': 4, 'spam': 2},
 'spiders': {'ham': 1, 'spam': 3},
 'hanging': {'ham': 9, 'spam': 2}}
```

Finding the posteriors with Bayes theorem

- This means that for example the word ‘woods’ appears 4 times in spam emails, and 2 times in ham emails.
- Let’s find out the appearances of our words ‘lottery’ and ‘sale’

```
model[‘lottery’]
{'ham': 1, 'spam': 9}

model[‘sale’]
{'ham': 42, 'spam': 39}
```

Implementing the naive Bayes algorithm

- But we are interested in using more than one word, so let's code the naive Bayes algorithm.
- Our algorithm takes as input, an email.
- It goes through all the words in the email, and for each word, it calculates the probabilities that a spam email contains it, and that a ham email contains it.

Implementing the naive Bayes algorithm

- The code to train the model is the following:

```
def predict(email):
    words = set(email.split())
    spams = []
    hams = []
    for word in words:        #A
        if word in model:     #B
            spams.append(model[word]['spam'])    #C
            hams.append(model[word]['ham'])
    prod_spams = long(np.prod(spams))        #D
    prod_hams = long(np.prod(hams))
    return prod_spams/(prod_spams + prod_hams)      #E
```

Implementing the naive Bayes algorithm

- And that's it! Let's test the algorithm on some emails:

```
predict_naive_bayes('hi mom how are you')
0.0013894756610580057

predict_naive_bayes('meet me at the lobby of the hotel at nine am')
0.02490194297492509

predict_naive_bayes('enter the lottery to win three million dollars')
0.38569290647197135

predict_naive_bayes('buy cheap lottery easy money now')
0.9913514898646872
```

Summary

- Bayes theorem is a technique widely used in probability, statistics, and machine learning.
- Bayes theorem consists in calculating a posterior probability, based on a prior probability and an event.
- The prior probability is a basic calculation of a probability, given very little information.

7: Splitting data by asking questions: Decision trees

TriveraTech
TECHNOLOGY TRAINING

Splitting data by asking questions: Decision trees

This lesson covers

- What is a decision tree?
- Recommending apps using the demographic information of the users.
- Asking a series of successive questions to build a good classifier.
- Accuracy, Gini index, and Entropy, and their role in building decision trees.

Splitting data by asking questions: Decision trees

Splitting data by asking questions: Decision trees

- **DECISION TREE** A classification model based on yes/no questions and represented by a binary tree. The tree has a root, nodes, and leaves.

- **ROOT** The topmost node of the tree. It contains the first yes/no question.

- **NODE** Each yes/no question in our model is represented by a node, or decision stump, with two branches emanating from it (one for the ‘yes’ answer, and one from the ‘no’ answer).

- **LEAF** When we reach a point where we don’t ask a question and instead we make a decision, we have reached a leaf of the tree.

Splitting data by asking questions: Decision trees

Splitting data by asking questions: Decision trees

Splitting data by asking questions: Decision trees

PICKING A GOOD FIRST QUESTION

We need to pick a good first question for the root of our tree. What would this first question be? Initially, it can be anything. Let's say we come up with five candidates for our first question:

1. Is it raining?
2. Is it hot outside?
3. Am I hungry?
4. Is there a red car outside?
5. Is it Monday?

Splitting data by asking questions: Decision trees

Splitting data by asking questions: Decision trees

The problem: We need to recommend apps to users according to what they are likely to download

- **Atom Count:** An app that counts the number of atoms in your body.
- **Beehive Finder:** An app that maps your location and finds the closest beehives.
- **Check Mate Mate:** An app for finding Australian chess players.

The problem: We need to recommend apps to users according to what they are likely to download

Atom count

App for counting the number of atoms in your body

Beehive Finder

App for locating the nearest beehives to your location

Check Mate Mate

App for finding Australian chess players in your area

The problem: We need to recommend apps to users according to what they are likely to download

Gender	Age	App
F	15	Atom Count
F	25	Check Mate Mate

The problem: We need to recommend apps to users according to what they are likely to download

M	32	Beehive Finder
F	35	Check Mate Mate
M	12	Atom Count
M	14	Atom Count

The remember-formulate-predict framework

- **Remember:** Look at previous data (previous customers, their age, gender, and what apps they have downloaded).

- **Formulate**: Formulate a rule that tells us if which app a new customer is most likely to download.

- **Predict:** When a new customer comes in, we use the rule to guess what app they are most likely to download, and we recommend them that app

First step to build the model: Asking the best question

Gender	Age	App
F	young	Atom Count
F	adult	Check Mate Mate

First step to build the model: Asking the best question

M	adult	Beehive Finder
F	adult	Check Mate Mate
M	young	Atom Count
M	young	Atom Count

First step to build the model: Asking the best question

- Out of the females, one downloaded Atom Count and two downloaded Check Mate .
- Out of the males, two downloaded Atom Count and one downloaded Beehive Finder.

First step to build the model: Asking the best question

TriveraTech
TECHNOLOGY TRAINING

First step to build the model: Asking the best question

Second question: Is the user young or adult?

- Out of the young users, all downloaded Atom Count.
- Out of the adult users, two downloaded Atom Count and one downloaded Beehive Finder.

The resulting node is drawn in next figure

First step to build the model: Asking the best question

First step to build the model: Asking the best question

- **Classifier 1:** Asks the question ‘what is your gender?’, and from there, determines what app to recommend.

- **Classifier 2:** Asks the question ‘what is your age?’, and from there, determines what app to recommend.

First step to build the model: Asking the best question

Next and final step: Iterate by asking the best question every time

Gender	Age	App
F	adult	Beehive Finder
M	adult	Check Mate Mate
F	adult	Beehive Finder

Final tree

Next and final step: Iterate by asking the best question every time

- And we're done! This is our decision tree.
- Here is the pseudocode for the predictions that this model makes.

```
predict(user):
        if user is young:
          recommend Atom Count
        else:
          if user is female:
                  recommend Check Mate Mate
          else:
                  recommend Beehive Finder
```

Building the tree: How to pick the right feature to split

Gender	Age	Location	App
Female	Young	A	1
Male	Young	A	1
Female	Young	A	1
Male	Adult	A	1

Building the tree: How to pick the right feature to split

Female	Young	B	2
Male	Adult	B	2
Female	Adult	B	2
Male	Adult	B	2

Building the tree: How to pick the right feature to split

How to pick the best feature to split our data: Accuracy

- Classifier 1: What is your gender?
- Classifier 2: What is your age?
- Classifier 3: What is your location?

How to pick the best feature to split our data: Accuracy

- The three classifiers, together with their accuracies, are shown here.

How to pick the best feature to split our data: Gini impurity

- Set 1: App 1, App 1, App 1, App 1.
- Set 2: App 1, App 1, App 1, App 2.
- Set 3: App 1, App 1, App 2, App 2.
- Set 4: App 1, App 2, App 2, App 2.
- Set 5: App 2, App 2, App 2, App 2.

How to pick the best feature to split our data: Gini impurity

Low Gini impurity index

High Gini impurity index

How to pick the best feature to split our data: Gini impurity

- If the set has low Gini impurity index, then most balls are of the same color, Therefore the probability that the two balls are of different color is very low.

- If the set has high Gini impurity index, then the balls tend to be of different colors, Therefore the probability that the two balls are of different color is very high.

How to pick the best feature to split our data: Gini impurity

- Set 1: {black, black, black, black}
- Set 2: {black, black, black, white}
- Set 3: {black, black, white, white}
- Set 4: {black, white, white, white}
- Set 5: {white, white, white, white}

How to pick the best feature to split our data: Gini impurity

Set 1:

TriveraTech
TECHNOLOGY TRAINING

How to pick the best feature to split our data: Gini impurity

Set 2

TriveraTech
TECHNOLOGY TRAINING

How to pick the best feature to split our data: Gini impurity

Set 3

TriveraTech
TECHNOLOGY TRAINING

How to pick the best feature to split our data: Gini impurity

Sets 4 and 5:

- Set 1: (App 1, App 1, App 1, App 1}. Gini impurity index: 0.
- Set 2: (App 1, App 1, App 1, App 2}. Gini impurity index: 0.375.
- Set 3: (App 1, App 1, App 2, App 2}. Gini impurity index: 05.
- Set 4: (App 1, App 2, App 2, App 2}. Gini impurity index: 0375.
- Set 5: (App 2, App 2, App 2, App 2}. Gini impurity index: 0.

How to pick the best feature to split our data: Gini impurity

How to pick the best feature to split our data: Gini impurity

- Formula for Gini index Like everything in machine learning, Gini index has a formula, And the formula is rather simple.
- Let’s calculate it, for example, for the bag with three black balls and one white ball.
- What is the probability that we pick a ball out of this bag, we look at it and put it back, then we pick another ball, and both balls are of different color?
- This is called picking a ball with repetition, since we put the first ball back after we pick it.

How to pick the best feature to split our data: Gini impurity

$$\text{Gini Impurity Index} = P(\text{picking two balls of different color})$$

$$= 1 - P(\text{picking two balls of the same color})$$

$$= 1 - p_1^2 - p_2^2 - \ldots - p_N^2$$

P(Both balls are color 1)

P(Both balls are color 2)

P(Both balls are color N)

How to pick the best feature to split our data: Gini impurity

- Picking blue: $p_1 = \frac{3}{6} = \frac{1}{2}$.
- Picking red: $p_2 = \frac{2}{6} = \frac{1}{3}$.
- Picking yellow: $p_3 = \frac{1}{6}$.

Thus, the Gini impurity index is precisely $1 - {p_1}^2 - {p_2}^2 = 1 - (\frac{1}{2})^2 - (\frac{1}{3})^2 - (\frac{1}{6})^2 = \frac{22}{36}$.

How to pick the best feature to split our data: Gini impurity

$$\text{Gini Impurity Index} = P(\text{picking two balls of different color})$$

$$= 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{2}{6}\right)^2 - \left(\frac{1}{6}\right)^2$$

P(Both balls are black)

P(Both balls are grey)

P(Both balls are white)

How to pick the best feature to split our data: Gini impurity

HOW TO PICK THE BEST FEATURE? GINI GAIN

- Calculate the Gini impurity index of the root.
- Calculate the average of the Gini impurity indices of the two leaves.
- Subtract them, to obtain the gain on Gini impurity index.

How to pick the best feature to split our data: Gini impurity

How to pick the best feature to split our data: Gini impurity

WEIGHTED GINI IMPURITY

Gini impurity = 0.375 Gini impurity = 0

$$\text{Weighted average} = 0.375\left(\frac{6}{8}\right) + 0\left(\frac{2}{8}\right) = 0.25$$

Back to recommending apps: Building our decision tree using Gini index

Gender	Age	App
F	young	Atom Count
F	adult	Check Mate Mate
M	adult	Beehive Finder

Back to recommending apps: Building our decision tree using Gini index

F	adult	Check Mate Mate
M	young	Atom Count

M	young	Atom Count

Back to recommending apps: Building our decision tree using Gini index

- Using the formula that we learned in the previous section, the Gini impurity index for this set is:

Gini impurity index of (A, C, B, C, A, A} = $1 - (\frac{1}{2})^2 - (\frac{1}{6})^2 - (\frac{1}{3})^2 = 0.611$

Back to recommending apps: Building our decision tree using Gini index

SPLITTING BY GENDER

As we've seen before, splitting by gender gives us the two following sets:

- Females: {A, C, C}
- Males: {B, A, A}

The Gini impurity indices of these two are the following:

- Females: $1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = 0.444$
- Males: $1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = 0.444$

Therefore, the average Gini impurity index of this splitting is 0.611. Since the Gini impurity of the root was 0.444, we conclude that the Gini gain is:

Gini gain = 0.611 - 0.444 = 0.167.

Trivera Tech
TECHNOLOGY TRAINING

Back to recommending apps: Building our decision tree using Gini index

SPLITTING BY AGE

The Gini impuritv indices of these two are the following:

- Young: $1 - (\frac{3}{3})^2 = 0$
- Adults: $1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = 0.444$

Therefore, the average Gini impurity index of this splitting is 0.222 (the average of 0 and 0.444). Since the Gini impurity of the root was 0.611, we conclude that the Gini gain is:

Gini gain = 0.611-0.222 = 0.389.

Back to recommending apps: Building our decision tree using Gini index

COMPARING GINI INDICES

Back to recommending apps: Building our decision tree using Gini index

- The two females downloaded Check Mate Mate.
- If we make a node with these two, the node has Gini impurity index of 0.
- The male downloaded Beehive Finder.
- If we make a node with this one, the node has a Gini impurity index of 0.

Back to recommending apps: Building our decision tree using Gini index

Back to recommending apps: Building our decision tree using Gini index

Beyond questions like yes/no

- We learned a way to split data based on a feature that only has two classes.
- Namely, features that split the users into two groups such as female/male, or young/adult.
- What if we had a feature that split the users into three or more classes?

Features with more categories, such as Dog/Cat/Bird

When the feature has more classes, such as Dog/Cat/Bird, we simply use more binary (yes/no) questions. In this case, we would ask the following:

- Does the user own a dog?
- Does the user own a cat?
- Does the user own a bird?

Features with more categories, such as Dog/Cat/Bird

Features with more categories, such as Dog/Cat/Bird

Animal
Dog
Cat
Bird
Dog
Bird

One-hot encoding →

Dog?	Cat?	Bird?
1	0	0
0	1	0
0	0	1
1	0	0
0	0	1

Continuous features, such as a number

Gender	Age	App
F	15	Atom Count
F	25	

		Check Mate Mate
M	32	Beehive Finder
F	35	Check Mate Mate
M	12	Atom Count
M	14	Atom Count

Continuous features, such as a number

Question	First set	Second set
Is the user younger than 7?	empty	12, 14, 15, 25, 32, 35
Is the user younger than 13?	12	14, 15, 25, 32, 35
Is the user younger than 14.5?	12, 14	15, 25, 32, 35
Is the user younger than 20?	12, 14, 15	25, 32, 35
Is the user younger than 27?	12, 14, 15, 25	32, 35
Is the user younger than 33?	12, 14, 15, 25, 32	35
Is the user younger than 100?	12, 14, 15, 25, 32, 35	empty

Continuous features, such as a number

Continuous features, such as a number

- Younger than 20: {12, 14, 15, 25}
- Older than 20: {32, 35}

Continuous features, such as a number

Continuous features, such as a number

COMPUTING THE ACCURACY OF THIS SPLIT

Accuracy: 4/6

Continuous features, such as a number

- The left leaf has three users downloading Atom Count, and one downloading Check Mate Mate, so the Gini impurity is

$$1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

- The right leaf has one user downloading Instagram, and one downloading Check Mate Mate, so the Gini impurity is

$$1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

Continuous features, such as a number

- Also note that the left leaf has four users and the right leaf has two users.
- Since there are six users in total, then the weights for the entropies of the left and right leaves is 4/6 and 2/6, respectively (or ⅔ and ⅓).
- Thus, our weighted Gini impurity at the leaves is

$$\frac{4}{6} \cdot 0.375 + \frac{2}{6} \cdot 0.5 = 0.417$$

Continuous features, such as a number

Gini (weighted) average = $0.375\left(\frac{4}{6}\right) + 0.5\left(\frac{2}{6}\right) = 0.417$

Gini gain = 0.611 - 0.417 = 0.194

Continuous features, such as a number

- Comparing our metrics for all the splits Now, in order to find the best split, all we have to do is calculate the accuracy (or the Gini gain) for all of our possible 5 splits.

- I encourage you to calculate them by yourself, and check that you get the same answers I got.

Continuous features, such as a number

Splitting	Labels	Accuracy	Gini gain
{12} {14, 15, 25, 32, 40}	A A, A, C, B, C	3/6	0.078
{12, 14} {15, 25, 32, 40}	A, A A, C, B, C	4/6	0.194
{12, 14, 15} **{25, 32, 40}**	**A, A, A** **C, B, C**	**5/6**	**0.389**
{12, 14, 15, 25} {32, 40}	A, A, A, C B, C	4/6	0.194
{12, 14, 15, 25, 32} {40}	A, A, A, C, B C	4/6	0.144

Continuous features, such as a number

Splitting	Labels	Accuracy	Gini impurity	Information gain
Male/Female	P, P, I P, W, W	0.666	0.444	

Continuous features, such as a number

```
predict(user):
        if user's age is less than or equal to 20:
          recommend Atom Count
        else:
          if user's gender is female:
                  recommend Check Mate Mate
          else:
                  recommend Beehive Finder
```

Continuous features, such as a number

TriveraTech
TECHNOLOGY TRAINING

Coding a decision tree with sklearn

```
import pandas as pd
app_dataset = pd.DataFrame({
    'Gender_Female':[1,1,0,1,0,0],
    'Gender_Male':[0,0,1,0,1,1],
    'Age': [15, 25, 32, 35, 12, 14],
    'App': ['Atom Count', 'Check Mate Mate', 'Beehive Finder', 'Check Mate Mate', 'Atom
      Count', 'Atom Count']})
print(app_dataset)
```

Coding a decision tree with sklearn

	Age	Gender_Female	Gender_Male	App
0	15	1	0	Atom count
1	25	1	0	Check Mate Mate
2	32	0	1	Beehive Finder
3	35	1	0	Check Mate Mate
4	12	0	1	Atom count
5	14	0	1	Atom count

Coding a decision tree with sklearn

```
features = app_dataset_new[['Age','Gender_Female','Gender_Male']]
labels = app_dataset_new[['App']]
```

Finally, we train our model by using the DecisionTreeClassifier() class from sklearn. The model will be called 'app_decision_tree', and the command for fitting the data is simply 'fit()'.

```
app_decision_tree = DecisionTreeClassifier()
app_decision_tree.fit(features, labels)
```

Coding a decision tree with sklearn

Coding a decision tree with sklearn

A slightly larger example: Spam detection again!

Lottery	Sale	Spam
7	1	No
3	2	No
3	9	No
1	3	No
2	6	No
4	7	No
1	9	Yes
3	10	Yes
6	5	Yes
7	8	Yes
8	4	Yes
9	6	Yes

A slightly larger example: Spam detection again!

- First, let’s put our data into a Pandas DataFrame.

```
spam_dataset = pd.DataFrame({
    'Lottery':[7,3,9,1,2,4,1,3,6,7,8,9],
    'Sale':[1,2,3,3,6,7,9,10,5,8,4,6],
    'Spam': [‘spam’,‘spam’,‘spam’,‘spam’,‘spam’,‘spam’,’ham,’ham,’ham,’ham,’ham,’ham]})
```

A slightly larger example: Spam detection again!

A slightly larger example: Spam detection again!

Next, we split into features and labels

```
features = spam_dataset[['Lottery', 'Sale']]
labels = spam_dataset['Spam']
```

Now, let's define our decision tree model and train it.

```
spam_decision_tree = DecisionTreeClassifier()
spam_decision_tree.fit(X,y)
```

A slightly larger example: Spam detection again!

A slightly larger example: Spam detection again!

```
predict(email):
        if number of appearances of ‘sale’ is less than or equal to 3.5:
          classify as ham
        else:
          if number of appearances of ‘lottery’ is greater than 5.0:
                  classify as spam
          else:
                  if number of appearances of ‘sale’ is less than or equal to 8.0:
                          classify as ham
                  else:
                          Classify as spam
```

A slightly larger example: Spam detection again!

A slightly larger example: Spam detection again!

A slightly larger example: Spam detection again!

A slightly larger example: Spam detection again!

Applications

In this section, I give some examples of decision trees used in real life in the following fields:

- Health care
- Recommendation systems

Decision trees are widely used in health care

- Decision trees are widely used in medicine, not only to make predictions, but also to identify features that are determinant in the prediction.

- You can imagine that in medicine, a black box saying “the patient is sick” or “the patient is healthy” is not good enough.

Decision trees are useful in recommendation systems

- In recommendation systems, decision trees are also very useful. One of the most famous recommendation systems problems, the Netflix prize, was won with the help of decision trees.

- In 2006, Netflix held a competition which involved building the best possible recommendation system to predict user ratings of their movies.

Summary

- Decision trees are a very important algorithm in machine learning, used for classification.
- The way decision trees work is by asking binary questions about our data, and making the prediction based on the answers to those questions.

8: Combining models to maximize Results Ensemble learning

TriveraTech
TECHNOLOGY TRAINING

Combining models to maximize results Ensemble learning

This lesson covers

- What is ensemble learning.
- Joining several weak classifiers to form a strong classifier.
- Bagging: A method to randomly join several classifiers.
- Boosting: A method to join several classifiers in a smarter way.
- AdaBoost: A very successful example of boosting methods.

With a little help from our friends

- Here is the scenario, you have to take an exam that consists of 100 true/false questions on many different topics, such as math, geography, science, history, music, and so on.
- Luckily, you are allowed to call your five friends, Alice, Bob, Carlos, Dana, and Emily to help you.
- What are some techniques that you can use to get their help? Let me show you two techniques that I can think of.

1. Technique 1
2. Technique 2

With a little help from our friends

Why an ensemble of learners? Why not just one really good learner?

Lottery	Sale	Spam
7	1	No
3	2	No
3	9	No
1	3	No
2	6	No
4	7	No

Why an ensemble of learners? Why not just one really good learner?

1	9	Yes
3	10	Yes
6	5	Yes
7	8	Yes
8	4	Yes
9	6	Yes

Why an ensemble of learners? Why not just one really good learner?

Why an ensemble of learners? Why not just one really good learner?

Why an ensemble of learners? Why not just one really good learner?

Buy	Lottery	Spam
8	6	No
2	2	Yes

Why an ensemble of learners? Why not just one really good learner?

TriveraTech
TECHNOLOGY TRAINING

Why an ensemble of learners? Why not just one really good learner?

- Now let's try to fit a decision tree to this data. We can use sklearn, like we did in lesson 7,with the following command in sklearn:

```
spam_decision_tree = DecisionTreeClassifier()
spam_decision_tree.fit(new_X,new_y)
```

Why an ensemble of learners? Why not just one really good learner?

TriveraTech
TECHNOLOGY TRAINING

Bagging - Joining some classifiers together to build a stronger classifier

- Bagging is a technique in which we build a strong learner based on a set of weak learners.
- The way the strong learner makes a prediction is simply by allowing the weak learners to vote.
- Whichever prediction gets more votes, is the one the strong learner makes.

Building random forests by joining several trees

Building random forests by joining several trees

Decision Tree 1

Decision Tree 2

Decision Tree 3

Building random forests by joining several trees

TriveraTech
TECHNOLOGY TRAINING

Coding a random forest in sklearn

- We will build one with five decision trees, or ‘estimators’.
- The command is the following:

```
from sklearn.ensemble import RandomForestClassifier
random_forest_model = RandomForestClassifier(random_state=0, n_estimators=5)
random_forest_model.fit(new_X,new_y)
random_forest_model.score(new_X,new_y)
```

Coding a random forest in sklearn

Coding a random forest in sklearn

Boosting - Joining some classifiers together in a smarter way to get a stronger classifier

- Boosting is very similar to bagging, except now we don't select the weak learners at random, but we select them in a more intelligent way.

- The way we do this is by training each learner to focus on the weaknesses of the previous ones.

- In other words, each learner tries really hard to correctly classify the points in which the previous classifiers.

Boosting - Joining some classifiers together in a smarter way to get a stronger classifier

A big picture of AdaBoost

- First let's try to fit a decision tree of depth one, That is simply a vertical or horizontal line.
- There are a few that work, so let's pick the vertical line in the left of next figure, which correctly classifies the two triangles in the left, and all the squares.
- That classifier is weak learner 1.

A big picture of AdaBoost

Train learner 1

Enlarge errors
Shrink correctly classified points

A big picture of AdaBoost

A big picture of AdaBoost

A detailed (mathematical) picture of AdaBoost

$$P(red\ ball) = \frac{\#\ red\ balls}{\#\ balls},$$

Then the formula for odds (the odds ratio--OR) is

$$OR(red\ ball) = \frac{\#\ red\ balls}{\#\ blue\ balls}.$$

A detailed (mathematical) picture of AdaBoost

Notice that since the total number of balls is #balls = #red balls + #blue balls, then we can conclude that

$$OR(red\ ball) = \frac{\#\ red\ balls}{\#\ balls - \#red\ balls}.$$

From here, we can see that in general, probability and odds are related via the following equation:

$$OR = \frac{P}{1-P},$$

where OR is the odds ratio. In the previous example, if the probability of a red ball is P(red ball) = ⅔, then the odds ratio is:

$$OR(red\ ball) = \frac{P(red\ ball)}{1 - P(red\ ball)} = \frac{2/3}{1 - 2/3} = 2.$$

A detailed (mathematical) picture of AdaBoost

$$Rescaling\ factor = \frac{Number\ of\ correctly\ classified\ points}{Number\ of\ incorrectly\ classified\ points}.$$

As a matter of fact, we can do better. Since we'll change the weights of the points during the process, the better way to formulate the rescaling factor is the following.

$$Rescaling\ factor = \frac{Sum\ of\ weights\ of\ correctly\ classified\ points}{Sum\ of\ weights\ of\ incorrectly\ classified\ points}.$$

A detailed (mathematical) picture of AdaBoost

Calculate rescaling factor
= 7/3
= 2.33

Multiply incorrectly classified points
by rescaling factor

Normalize

A detailed (mathematical) picture of AdaBoost

Calculate rescaling factor
= 7.84/2.13
= 3.68

Multiply incorrectly classified points
by rescaling factor

Normalize

A detailed (mathematical) picture of AdaBoost

Calculate rescaling factor
= 8.46/1.35 = 6.27

A detailed (mathematical) picture of AdaBoost

Accuracy	Odds = #Correct/#Errors	ln(Odds)
99%	99/1 = 99	4.595
70%	70/30 = 2.333	0.8473
50%	50/50 = 1	0
30%	30/70 = 0.4286	-0.8473

A detailed (mathematical) picture of AdaBoost

1%	1/99 = 0.0101	-4.595

The logarithm of the odds, which is commonly known as logit (short for "logistic unit"), is the weight that we assign to each model in the voting. In Figure 10.18 we can see a plot of this.

TriveraTech
TECHNOLOGY TRAINING

A detailed (mathematical) picture of AdaBoost

A detailed (mathematical) picture of AdaBoost

Coding AdaBoost in Sklearn

```
from sklearn.ensemble import AdaBoostClassifier
adaboost_model = AdaBoostClassifier(random_state=0, n_estimators=6)
adaboost_model.fit(new_X, new_y)
```

Coding AdaBoost in Sklearn

Coding AdaBoost in Sklearn

```
estimators = adaboost_model.estimators_
for estimator in estimators:
    plot_model(new_X, new_y, estimator)
    plt.show()
```

Coding AdaBoost in Sklearn

Applications of ensemble methods

- Ensemble methods are some of the most useful machine learning techniques used nowadays as they exhibit great levels of performance with relatively low cost.

- One of the places where ensemble methods are used the most is in machine learning challenges such as the Netflix challenge.

Summary

- Ensemble methods are ways we use to combine weak learners into a strong one
- There are two major types of ensemble methods: Bagging and boosting.
- Bagging, or bootstrap aggregating, consists of building successive learners on random subsets of our data, and then building a strong classifier based on a majority vote.