Hello, everyone. Welcome back. Let's get started with the introduction to this class. This class is mainly about machine learning. What is machine learning? You can view this as a machine that process data and get some output. The input is data and output are some tasks. Depending on your problem, it could be generally some useful information, making predictions, testing validate hypothesis, making classifications, identifying patterns. And the goal of course is to uptime this machine to be able to get useful information as much as possible.

The scale of the problem, and we have tons of data nowadays. So, I've listed some of the information here, and Wikipedia has over 30 minutes of pages. Now, this slide was made a few years ago and nowadays its numbers could be even larger and needs to be updated. So, Facebook and social media have tons of users, and these places constantly generate new information that we could use. Besides these, there are tons of different applications and machine learning becomes more and more relevant to many scientific and engineering problems as well. For example, study of brain activities to understanding brain functions, in the domain of neuroscience and there are sensors in our eyes, different ways to get data out of the gray, and these data could be different images or some signals. And just looking at this raw data, how do we make sense of them? Understanding how brain function, that's the task we're facing here. So, I've listed some other applications here, including astronomy, self-driving car, robots, finance, weather forecasting, music, design, and so on. This list is expanding because machine learning as you can see it's really becoming a building block to many of these modern problems.

So, let's look at a few applications and specific problems, and we can see what the different challenges and tasks are arriving from these problems. First one is, how do we organize images? Suppose you have a very large image database, one such image database now become their standard is called ImageNet. It has over millions of pictures. These pictures are collected for different classes. For example, these are images about the beach and sea, about flowers, sunset, and so on. And how do we organize these pictures? Making sense of them that you look at these pictures, all about the seaside, these are all about sunset. So, there are some similarities between these images. How do we discover this similarity that these pictures belong to one group, this is another group? so, this problem of finding similarities and patterns in the data is related to a task called clustering in machine learning. And so, if you could find a way to represent these images and reduce their dimensions, project them into a two-dimensional space, ideally, we wanted them to see this clustering behavior, for example, these are all the images corresponding to the sunset and these are all the image corresponding to the flowers, you can see they form two different clusters. So, when you look at this problem, we typically follow through these thought process. What is the desired outcome? In this case, we want the images to be clustered using the right representation of the images and right processing tools, algorithms to be able to cluster the images. What are the input? And in this case, is images and what are the learning objective or paradigm? We want to be able to find similarities and clustering behavior of these images.

And the next problem is **finding communities in social networks**. And here is one visualization of the large social networks based on their interests. And as you can see, this is a pretty cool visualization of that data. And we're able to discover the communities in these very large social networks. For example, these nodes, each node is supposed to be representing one user, and you can see these are the users that are mostly interested in NBA, and these are interested in anime and so on. So, how do we discover the interests or different clusters of these nodes inside a very large network? It's a problem here. So, in this study, what is the data? Data are going to be the network data, social networks, and network data is very common nowadays from modern application social networks, one important application, network data. In network data, you represent the data by nodes and the edges between them. So, the links and nodes, and each node represent one user, and the edges between them representing there is a connection between the two users. There could be direction in a connection as well, maybe you find someone, but not vice versa. And it could be directional. How do we look at these data to find what do I mean by community? For example, if I have a network like this, and so, here I have these nodes, as you can see, that these three nodes are all connected to each other, and these nodes are all connected to each other. But in-between these two groups of nodes, the connection is not fully connected. And the nodes are more densely connected and in-between the communities are not as tightly connected. And that's the definition of the community. And how do we understand and be able to identify these clusters of nodes in a very large network is a problem we're still facing here. And the input data here are going to be the network data, and desired outcome are going to be find out the communities, and these communities are totally connected nodes. And how do we achieve this goal? Learning paradigm is to be able to find clusters. But this time the clusters are defined in terms of the connectivity of your nodes. So, again, in case you found the three regimes.

Next application is to **visualize the image relations**. And here, each image is represented by thousands or millions of pixels. So, you can see these pictures are all about the same person, but the orientation and the expression of the faces are different. So, clearly, there seems to be some logic order of these faces. For example, on this side, as you can see, it's a smiley face, and on this side seems like making a funny face, and this side seems to be not so, happy faces. And is it possible to be able to find out the semantic information between each face and organize the phases accordingly? It's a challenge here. And so, to find out the relations when you organize these pictures is the challenge here. So, here the desired outcome is to be able to understand the relations through the so-called **embedding**. Because we project these very high dimensional images into two-dimensional space where each dot in the picture representing one image and we want to achieve an output, the input are these images, which is high-dimensional with millions of pixels, and the objective is to be able to find out relations. Relations could be based on their semantics, with actual meaning of these pictures. And semantics could be possibly extracted based on similarities between the pictures as well. So, how do we develop algorithm to find out deeper meanings and connections between the images?

The next question is ‘**feature selection’**. So, in this example, I have three clusters here. And as you can see, if I choose the right feature, meaning how to represent the data, each data point is represented using a two-dimensional point in this Euclidean space. So, if you choose the right way to represent your data, and we're able to separate these three clusters. Each of them belong to one class. And so, how do we define this x_1 and x_2? These are the definitions of features for each of the data points. For example, what does it mean by features? so, for example, I talk about a human I can say, what is the height and what is the weight of this person, and in this case, there are two features. And then these three groups, one belong to female, the other is male, and one group is possibly kids versus adults. So, by definition of the features, you could separate different groups. And spontaneously you can say, maybe this is not enough feature we're going to introduce maybe gender. And maybe other things like health condition, whether or not this person is healthy in general or what is the income maybe. So, you can throw in all the features and this feature can become long and long. And this is very common in modern machine learning applications as well. And when you collect the data, there's tons of information collected for each data points. So, when you have so, many features, the question you're facing is, which feature is important that helps you to perform your missionary task? And for example, if you want to classify, so, you have the doctor to make a decision this is a patient versus the healthy person. For example, COVID-19 diagnosis, that's exactly the problem we're facing. What are the most important feature that needs to come into play when you want to make this decision. And do you have these important features in your existing data already? If they are, can you identify these are the most important features. That's a problem of feature selection fitting here, the desired outcome or to find out the most important feature for your machine learning task. And the input data are going to be these data points with all the features. Each one of them is the one data points with the features. And learning paradigm is to be able to identify the most important dimensions or variables in your raw data definition.

Next one is **novelty/abnormality detection**. So, the novelty and anomalies are the same concept, but put in different contexts, it could mean different things. And in this example, you have a picture taken and possibly surveillance camera pictures. And we want to be able to identify things that are abnormal. Abnormal means they're not things you normally would accept to appear. And in this case, you want to build an algorithm to tell you that this car, it seems to be abnormal object because they normally wouldn't appear on campus in this case. So, it could also, be novelty, meaning that you have seen, for example, you have collected lots of picture of the bear. These are all bear, like brown bear that I know, this is one species of bear that I know. And then you have a new data coming in, and then you compare this picture with t data you already have in your training data, and you'll realize, oh, this is a new species of bear that I haven't seen in my training data. Can we possibly use learning algorithm to tell you this particular data point or this belong to a new species? so, in this case, the desired outcome is, the algorithm can tell you, given a new task data point, this is anomaly, or it is a novelty that something you haven't seen, it's something new. And the input data are going to be the training data that you have labeled as normal. You sometimes may have training data for abnormal data, sometimes you don't. So, you could totally face a one class classification problem. The learning paradigm here is going to be some classification. It could be two class or one class. And you can see we're hinting now into also, the concept of supervised versus unsupervised learning. So, it could either be supervised or unsupervised here.

Image classification, so, this image classification, you could have tons of images and this image actually here are taking out of this famous dataset, ImageNet. So, you have these images labeled with the content. As you can see, these are mite, these are container ships, and each image could also, have multiple labels attached to it. This is most likely to be mite, but the cost would be a tick or starfish. So, based on this training data with labels, we want to develop an algorithm that if I gave you a new picture, you could tell me based on the training data you had seen which class is most likely for this picture to be. I give you a picture, you tell me this is a person or So on. So, this is a learning paradigm. This belongs to a supervised classification, supervised learning because we have labels that tells you the content of each of the pictures. And these labels are provided by people. For example, human supervision is most common, adding labels, sometimes labels are not available. For example, getting back to what we talked about before finding patterns, organizing images, and finding communities. These usually belong to the unsupervised learning regime. Unsupervised learning, this is also, most likely unsupervised in most scenarios.

So, this next one is ‘**face detection’**. I think this has now become pretty common that I gave you a picture and an algorithm can find that in the picture these correspond to human faces. And because you have the features that defines this is a face. So, how do we do this. The desired outcome is to identify interesting pattern. My handwriting is not the best, interesting features. These interesting features here are going to be the faces and the input data are the images and learning paradigm. So, what does this belong to? And there could be multiple ways of doing this. Well, the common way of doing this is through the match filtering or nowadays this is also, commonly implemented using convolutional neural networks (CNN), which is basically a way of extracting patterns of features that you're interested in finding out. So, how do I define a face? A human face consists of eyes, a nose and a mouth, right? so, that's the basic definition. So, if your machine learning algorithm knows the definition of the face, then basically it's going to try to match this template and scan through your picture and finding out if there something that looked like a face, and you would label this as the face. So, the basic idea is to be able to define the feature and that defines your matching template, and then performing the matching, and then tell you that's the desired feature. Face detection.

And the next one, **weather forecasting**. This is getting into the regime of science and engineering. Given the task of observations there's a lot of physics define, evolved as well. Given the current observation in terms of the temperature, humidity, wind, and so on, can we predict what's going to happen in the future? Given all of the information, so, in order to make the prediction desired outcome, we want to be able to predict either these numerical values or maybe predict just categorical values. Is it sunny or rainy? The input data are your observations. in the past. So, you observed the whole trajectory of the evolution, for example, a cloud's coming here and in the next hour it's going to move across here. So, it's going to be spatial and temporal as well. Spatial-temporal problem because you're going to have the dynamics going on. Things could happen and propagate over space and time. And so, you could also, observe, given that pattern what's the outcome, is going to be rainy and thunderstorms or severe weather? And so, there's some supervision involved because from the history data, you can observe what actually happens under that condition. So, the learning paradigm here is most likely going to be supervised, although I would say in a world of weather prediction. So, the most commonly used method is using physics model. You can solve dynamical system like a PDE, and then you try to model the whole dynamic of the weather and then using the current observation to make the prediction of the future. But then, what's the opposite of physics model is using data-driven. You don't necessarily model what's happening behind the physics. And using this data-driven approach. Your machine learning algorithm is trying to find as a pattern using data entirely, not necessarily from physics. So, we're talking about the data-driven approach here. And it's become more interesting nowadays because sometimes it's harder to entirely model everything in physics. When we talk about the weather we talk about chances. So, what is the chance of having a rain? And the reason is, no matter how good you're building your physics model, the clouds could be driven by a lot of uncertainties and randomness. So, this can be hardly modeled through the physics model. So, in that case, sometimes data is going to capture things that it's hard to capture through the physics model. So, we'll talk about physics versus data-driven models.

So, there are different classifiers and linear classifiers we can talk about, for example, SVM Support Vector Machine. But sometimes the data cannot be well separated using linear decision boundary. In these cases, we only need non-linear decision boundaries. And so, how do we develop non-linear decision boundary, especially for high-dimensional cases. When the data is high dimensional, you can imagine this high-dimensional space. This is called the decision boundary, is going to be hard to characterize in high-dimensional space. So, how do we achieve that? There are kernel method and neural networks. These are all very good classifiers that are commonly used nowadays to achieve nonlinear classification. So, desired outcome be able to have this generalized the non-linear decision boundaries and input our supervised data with labels. And so, this is a supervised learning.

Spam filtering. And this is another very common successful application of machine learning algorithms and I think everybody probably have this experience. The spam filtering helps you to get rid of emails that are actually spams and scam. Sometimes they could be junk emails, they could be also, malicious scams. But sometimes they could also, be useful messages like someone sends you an email and you say, I didn't see it. And the person said, check your spam email, and you realize that your spam filter of your email make a mistake and misclassified email from your friend as a spam. So, you don't want that to happen. So, you want to develop a smart spam classification. Essentially this is classification to classify your email as a spam or an email from your friend. So, the desired outcome is to be able to classify it with high accuracy. So, classify with high accuracy. And the input data are going to be your own emails. But of course, when we build this spam filtering, these machine learners are going to collect a lot of training data, and these training data are going to be the normal data and the abnormal data that has been labeled as the spams. So, based on that, we can perform supervised learning. But it's more interesting here than just a normal, regular classification because there are additional information in the data that you could use. For example, if I know the connections with relations between these users, for example, I have listed someone as my friend. Then even if I didn't list them as a friend, but I have constantly clicked on the emails received from this particular user, then it's very likely that email from this user is not a spam. So, they're additional side information which is modeled using this network and modeling the connection between the users. This side of information should be used for you to connect the users and be able to do a better job of email spam filtering. So, this is the midpoint is sometimes in classification algorithms, it has a special feature like this. I could use additional side information to help you to improve a vanilla version of the algorithm.

Next example is **handwritten digit recognition**, text annotation given the input data. So, this is the handwriting. Could we translate the handwriting into the type writing words. So, the desired outcome is given input, could we get the correct meaningful transcript of your handwriting results? The inputs are going to be these handwritten images, and the outputs are going to be the meaningful paragraphs. So, how do we achieve a good accuracy here? so, most likely this is going to be supervised as well because you're going to tell you, machine, your algorithm to start with what's the meaning of what's likely to be each of the characters. But the additional feature or citing information that can help you in this problem to improve your accuracy is the inter-character dependency. So, basically, these characters are not independent or separate from each other because they're part of the sentence. For example, here, this clear, you're saying according to a study at Cambridge. And so if you try to decipher each of the image. And then the result becomes AO, CC, DR and IG. And so, this seems to be misspelled according. But then if you put everything together, you start to understand at a high level, was trying to say here is according to. So, finding out or using the inter-character dependency will help you to be able to do a better job, to do handwritten digit recognition. So, how to model this sequential interdependence between the characters is quite important in this particular problem. I'm going to touch it down the very basic modeling of the sequence data in this class as well.

Speech recognition is another instance where the goal is given the raw data, which is the input, you talk to your microphone and your voice signal is going to be recorded, which are these raw wave forms. And based on this can we translate this into actual understanding of what's actually being said. Machine learning is the preferred method for speech recognition. And so, by translating these raw wave forms to the actual text, there are a few steps involved. And so, one of the important components in speech recognition, recognizing the meaning of the speech is called a Hidden Markov Model. And Hidden Markov model is developed to capture the interdependence between the sequence of observations. So, in this example, y is going to be applicable because what has been said is basically language. Language has meanings. This meaning, and the way the words, letters are organized is governed by grammar of your language. So, it's not arbitrary sequence. This grammar of your language cost very strong dependence of these words. And so, be able to model that will help you do a better job of recognizing speech similar to here. So, here you capture the interdigitate dependence here, we're trying to capture the interdependence between the bay forms to do better job.

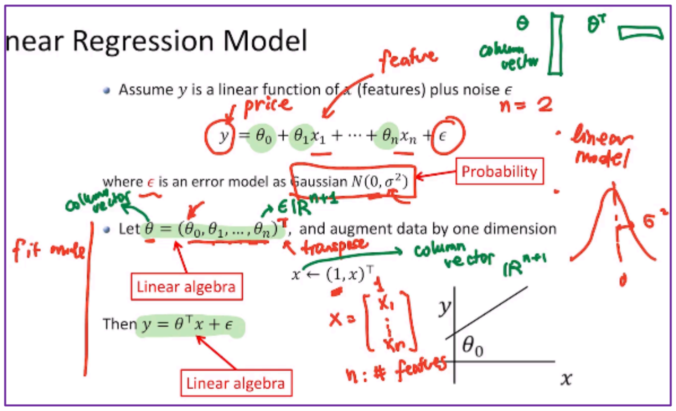
Our next example is how do we organizing documents? And so, this is in a regime of natural language processing. And this type of data nowadays is very common since we have lots of documents, clearly, lots of webpages with work content. These are all natural language processing data we are facing. And for example, how do we make sense of the meaning of an article and be able to be organizing these documents. And so one thing that's very common is called topic modeling. And so, the topic modeling, for example, is to say that based on what has been written here, you have this raw text. And let's try to extract some keywords out of this text from the raw documents to reputation of this document is through some model called the bag-of-words model. So, we'll be able to see how many times each of the words appeared in this document. And depending on the content of this document, the frequency can be quite different. For example, if this paragraph is talking about education and you can see this seems to be about education because it's mentioned a few times, education here and about school. And so, that help to define the actual content by looking at the distributions of the keywords. And so, for example here we define a few classes of topics. If it's about children, then you're going to see these keywords appear, children, women, people, years, family. If it's about budgets, you're going to start to see million, tax, program, budget, and so on. And so, by looking at different distribution of the keywords in your document will help to define topic. And how do we identify and be able to discover the topic? And so, there are different ways for doing that. One is called the latent semantic indexing (LSI). And it's basically a principal component analysis approach. And so, we'll talk about this in this class as well when we start to talk about principal component analysis and dimensionality reduction.

Next one is **product recommendations**. And so, I'm sure everybody have experience of using online shopping and for example, use Amazon. You click on the book; you want to buy our textbook Pattern Recognition and Machine Learning. And you can look at here, there are all recommendations if say, hey, check out these books. These are frequently bought together, and they can also, recommend things to you here and saying, hey, check out these books based on your browsing history, based on your purchase history, these are likely to be the books you might like. And so, that's a product recommendation, one of the most important components for these online shopping companies, for Amazon, for iTunes shop recommend songs or Netflix recommends movies to you. And so, how are these recommendations made? so, this is very important problem here and we're going to talk about this in our class as well. The basic idea is we can observe something about you. So, we represent the data using matrix (mXn). This is user (m), the other side is item (n). For example, have you purchased this item before? If you purchased it, did you like it? Have you ever rated this particular item? so, you have some observations about this user, about these items and so on. So, you're facing a giant matrix with partially observed entries. These entries are whether or not this user have used or rated or liked or disliked this item. And the question of course is, I haven't observed anything about the other entries in this matrix. And these question marks if I can make a good guess about value of these question mark. And I could recommend the items that this user is likely to like. And so, that's the idea of thinning out. I may be a little bit clearer. And this is also, known as the problem of matrix completion. You're solving a giant matrix completion problem to perform a good product recommendation. And so, in solving a matrix completion problem, it's evolved with things like finding out low-rank representation of your matrix and so on. And as you can see as we move along and the problem has drifted from the very basic classification problem, clustering into more involved complex problem with additional considerations. For example, this one. The desired outcome is to be able to recommend the correct item to user that a user is likely to purchase and likely to like. And input data are some partial observations of what has been liked by the user. And the learning paradigm here is, I would say, partially supervised. So, how to be able to figure out what hasn't been observed. It's really a missing data completion problem.

Robotic control. And so, self-driving car has been a very hot topic in the past few years. And so, how do you build a self-driving car as a big problem? so, if you have a car and then you can mount some camera and then you have radar and you have these are going to be the sensors that help the car collect real-time information about the environment. So, I forgot to draw wheels to define car. And then you're driving here. And so, these sensors help you to collect real-time information about the road and giving this information that feed into some algorithm here. Can the algorithm make a decision regarding what the car should do in the next moment? so, in this problem, this is an instance for controlling a robot where the car is controlled by these. The sensor collect inputs, and then the hours will have to make decision. So, in a sequential decision-making regime, so, this is touching upon reinforcement learning. One of the hard topics in machine learning, how do we make decisions sequentially? And the idea is, as you collect more and more information, so, before you start, you probably have built ours (model?) and that as you collect more and more information, as you move along, you always can improve itself over the time and be able to get better and better. So, that's the problem we're facing here. Reinforcement learning, sequential learning, and online learning, these are all genes, are problems and keywords you would see associated with controlled robots and a self-driving car. The desired outcome is you want to be able to perform sequential learning. This is different from classification where you have training data and you can take all the time you have and then develop the best always in and then testing it beforehand and then your fix your algorithm and then we have test data, you just apply it. The algorithm, you have already sign and to detrained and tuned without making any change. In the sequential decision-making, you can see I have to be able to update and revise my algorithm as I get more data and feedback so, that's the difference here as well. In this case, the input data are going to be these real-time observations randomize xt(?). And so, you keep getting new information, your observations and then using this to constantly improve your performance and learning paradigm is basically three, sequential decision-making, reinforced learning, and online learning. By the way, these are all going to be covered so, reverse motoring is relatively new topic. We're going to include this as the last part of a luxury. It's like a bonus. So, we're now going to assign any homework for this, but I do want to talk about these new topics to make the course of their complete. As you can see, the field of machine learning is moving so fast, so, there are constantly new topics developing and the field itself is evolving very quickly. And so, that's where we're trying to adapt our course is where our course itself is reinforced learning process. And so, as you read the syllabus, you realize there are certainly new things we added. We don't necessarily want to, as homework yet, but we do want you to learn that.

So, we've talked about different applications and motivating applications and so, now we're going to talk a little bit about what's are some prerequisites, what we need from you before you start. So, before you take the class, I hope to gain understanding that this class will have certain depth because it's a graduate-level class. So, we'll go deeper into the derivations of the reason. I'm going to give you one example in a couple of minutes. So, the basics we want to be able to understand our undergraduate level probabilities. For example, you understand what do I mean by distribution densities and marginalization, marginal distributions, conditional distribution, and statistics. So, what do I mean by maximum likelihood estimation, for example? And linear algebra plays an important role in our algorithm's development. So, this is one place if you are taking the class before but rusted-on this topic, I would highly encourage you to go back to your undergraduate course and just refresh your memory, get familiar with the linear algebra part of it. And so, I'm going to do some programming. And so, here I listed MATLAB, but also, Python is okay. And in fact, we encourage you to use Python as well. Sometimes you have to program from scratch implementing the algorithm yourself without calling it package so, I think that's the best way to learn. That's what actually happened in the nuts and bolts in algorithms and we're going to touch a little bit about optimization so, the basics will be covered in this lecture.

Machine learning for apartment hunting: Next one we will give you one illustrative example to see how do we actually develop a machine-learning algorithm to solve a problem. This problem is, suppose you want to move to Atlanta, you want to find the most reasonable price apartment to satisfy your need in terms of the square footage, number of bedrooms, and this is camps and so on so, you go ahead. What is the budget? Of course, if I have all the money I have, then I can rent my favorite department, but there's a budget, so, I want to find a reasonably priced apartment and so, you go ahead and collect information about living area, number of bedrooms, and how much it costs. So, now you have this information you want to go to see. Given that I'm considering this apartment, what is a reasonable price that I should be able to consider.


near Regression Model

- Assume y is a linear function of x (features) plus noise ϵ

$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n + \epsilon$$

where ϵ is an error model as Gaussian $N(0, \sigma^2)$

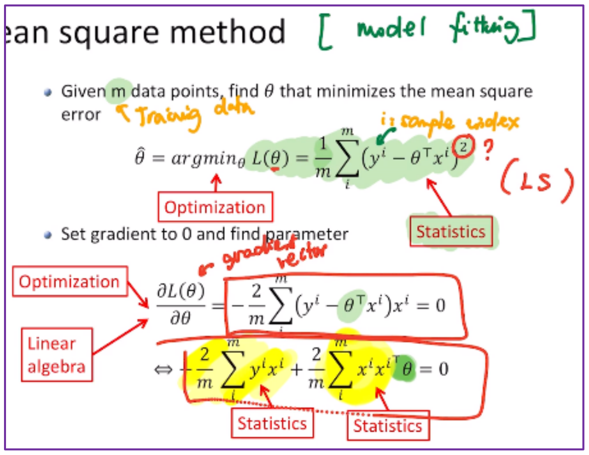
- Let $\theta = (\theta_0, \theta_1, \dots, \theta_n)^\top$, and augment data by one dimension

$x \leftarrow (1, x)^\top$

Then $y = \theta^\top x + \epsilon$

Linear Regressing Model: so, in this example, first we're going to introduce the map, this very practical problem into a mathematical problem. Here y is the response variable and in this case corresponds to the price and these x's are the features we're just talking about and in this hunting apartment application, the x corresponding to these two, we have two features, living area and number of bedrooms so n is 2. And then here these other Thetas and so, we have a model is a linear model basically we assume that the price is linearly dependent on the number of bedrooms and the size of the apartment. And the parameters for each of the features are going to be the Thetas and besides also, have my constant so, that's the intercept term. And of course, my model is now going to imperfect, so, I have errors and so, I'm going to assume that my error, if my model is a good model, my error is going to have zero mean, meaning that our Irish, my model is accurate it's not going to be having certain bias and it also, have variance. The variance is in the squares so, the meaning of that is that captures how large or how small your area is. And so, now we have a model. This is very simple linear model, and it is also, a probabilistic model because we have to make an assumption that your error is a Gaussian random variable, so, your error has a distribution centered at (0, σ^2). This captures the spread variability of your noise. And so, we would like to be able to figure out this model, meaning you want to be able to fit the coefficients, which is the Theta. So, now the next step is we discuss how do we fit your model from training data.

Model fitting is the rest. So, first of all, from now you can see we're starting to talk about how to represent data in a more compact way, we'll start using algebra. We will first introduce this vector Theta that contains all the coefficients so, zero is the intercept and the others are the coefficients for each of the variables. And here this transpose term is just the transpose. And so, we will augment the data by one dimension so, what does this mean? so, my x definition here is x_1, x_n, so, n is the number of our features. And so, after I have defined this, so, I will add a one on top of that and then I'm going to replace my old x versus new x so, this x is, I'm going to use a different color, this x is a column vector, and this Theta is also, a column vector. I want you to be able to come in that you can see it's very important to be mindful about the dimension. This dimension is n plus 1. This notation means this is a real-valued vector with dimension n plus 1 and x is also, n plus 1 dimensional. And so, after I've defined this, now have a very compact way of representing it. So, now I have y is equal to Theta transpose x. This is the linear algebra part. So, I have my reputation, and instead of writing this law equation now I have y is equal to Theta transpose x. By the way, what is transpose? so, my data is a column vector and Theta transpose is just transposing it, make it flax so, that's the transpose denoted by the t on the top.



ean square method [model fitting]

- Given m data points, find θ that minimizes the mean square error ↖ Training data

i: sample index

$$\hat{\theta} = argmin_\theta\, L(\theta) = \frac{1}{m}\sum_i^m (y^i - \theta^\top x^i)^{\boxed{2}}\ ?\quad (LS)$$

Optimization

Statistics

- Set gradient to 0 and find parameter

or gradient vector

Optimization

Linear algebra

$$\frac{\partial L(\theta)}{\partial \theta} = \boxed{-\frac{2}{m}\sum_i^m (y^i - \theta^\top x^i)x^i = 0}$$

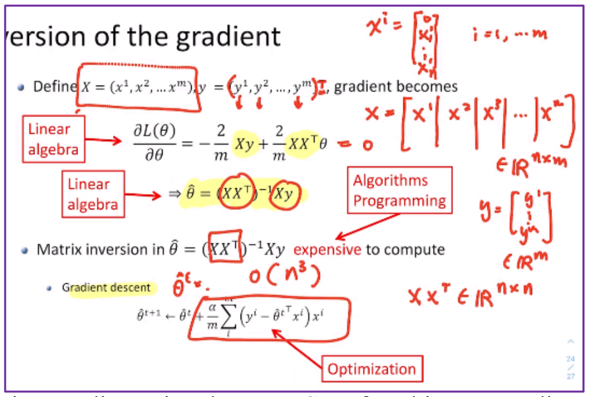
$$\Leftrightarrow -\frac{2}{m}\sum_i^m y^i x^i + \frac{2}{m}\sum_i^m x^i x^{i\top}\theta = 0$$

Statistics

Statistics

Least mean square method: So, this is square root method so, now this is trying to solve the model fitting. So, I'm going to introduce in some additional notations. So, here, as you can see, I'm writing this as a generic model, it doesn't have any superscript. The subscript representing dimensions may, but superscript represent samples. Let me use a different color just to emphasize now I'm talking about. The X representing simple index and I have m data points; these are going to be the m samples in my training dataset. So, this is my training data. And so, how do I find my theta? And so, we're going to solve this very simple objective. So, basically you remember we have this linear model, and then basically our data looks like it says like this, they are oscillating, of course, they're not going to exactly pass through this line. But if I look at my model EIR, these are going to be the difference of the points from this line. And so, if I tried to find a good model, I should minimize the total error made by my model. So, I'm going to find the best fit through this optimization problem. I'm going to minimize the theta and then this minimize theta for this loss function. This loss function defined as alpha-theta is the average sum of the squared errors. So, you can see here I have a square, so, squared errors of my fading. So, now I have reduced the problem of finding the best model to the problem of solving optimization problem. I want to find the theta that minimize this objective function. And so, here by the way, why do I consider sum of the squares of the errors? That's interesting. Have you thought about that? I think everybody take at least a square for granted, scholarly square LS method, the y there is a 2. I'll keep that open as one question could thinking about it, okay. They are actually a statistic meaning behind. Why do we choose Lisa square? As I'll explain in a minute.

How do I solve this optimization problem? And so, this part now can see a transition the problem that into the next stage solving this optimization problem. We're going to solve it by taking the derivative of the loss function with respect to theta and set it to be zero. Okay, and then, how do I find the derivative here? You require some linear algebra tricks as well because it's L. This L function is defined for theta. Theta is a vector. So, to find out the derivative of L with respect to a vector, that's going to give me a gradient vector. So, this actually is a gradient vector. And the optimizer that achieve the minimum of my problem should satisfy, because I don't have any constraints here, should satisfy that this gradient equal zero. So, by setting is going equal to zero, I'm any app have the equation that my optimal parameter theta should satisfy. So, after some rearrangement, I'm having this expression finally. And if I simplified further, I'll be able to write it into the more compact way using matrix and vector. So, let's look at this first. What does this mean? This means that my theta, where is my theta? I forgot it's here. Okay, that's the only place you want the theta. The other place is actually only involving data, so, let me highlight them. And you can see this term is basically like a covariance term between the response Y and each of your data point is x. And this term is more like the covariance for x itself. So, this basically these two terms that defines your equation depending on the statistics of the data again. So, optimization statistics are intertwined in solving this problem.



ersion of the gradient

$x^i = \begin{bmatrix} 0 \\ x_1^i \\ \vdots \\ x_n^i \end{bmatrix}$ $i = 1, \dots m$

- Define $\boxed{X = (x^1, x^2, \dots x^m)}$, $y = (y^1, y^2, \dots, y^m)^\top$, gradient becomes

$X = [x^1 | x^2 | x^3 | \cdots | x^m] \in \mathbb{R}^{n \times m}$

Linear algebra

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m}Xy + \frac{2}{m}XX^\top\theta = 0$$

Linear algebra

$$\Rightarrow \hat{\theta} = (XX^\top)^{-1}Xy$$

Algorithms Programming

$y = \begin{bmatrix} y^1 \\ \vdots \\ y^m \end{bmatrix} \in \mathbb{R}^m$

- Matrix inversion in $\hat{\theta} = (\boxed{XX^\top})^{-1}Xy$ expensive to compute $O(n^3)$

$XX^\top \in \mathbb{R}^{n \times n}$

  - Gradient descent $\hat{\theta}^t$.

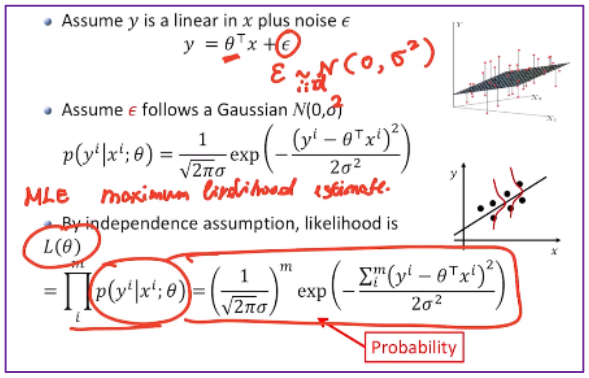
$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t \boxed{+ \frac{\alpha}{m}\sum_i^m (y^i - \hat{\theta}^{t\top}x^i)x^i}$$

Optimization

24 / 27

Okay, and now we want to be able to write this in a more compact way. So, we're going to define a data matrix here. And what is this x? Let me extend it. So, first of all, remember x i is our eyes data point. So, remember this x^i, the first entry is going to be 0. Then x^1 1 , another x^1. And this is the I is from 1 to n. So, this is my twin data point and my x to find here is a matrix. And the matrix is defined by i. Put x^1, x^2, and you should understand each of this is actually a vector. So, this x is a matrix of dimension m by m. Okay, as you can see as I illustrate, is very important to keep track of the dimension of your matrix and vectors to make sense out of it. And what is my Y? Each of these Y is my response variable, that's the price. So, even if you're dealing with math, don't forget the meetings, sometimes it will be helpful for you to understand these are the prices. So, y is going to be a vector $y^1 \dots y^m$. So, this transpose, if I write it this way, this is a row vector, and I put a transpose here; it's going to create a column vector. In our class, all the vectors are column vectors. So, this is a m-dimensional vector. So, after this, my gradient expression here. This is my gradient expression. You can validate using linear algebra can be written more compactly into this expression where you can see this involves x, y, and this $XX^T\theta$. Why this is helpful because now if I want to solve this equation equal to, I can write a closed-form expression again using linear algebra. So, theta now is equal to this. Okay, great. So, I have a closed-form expression of very clean, nice compact. And if I look at this, what does this involve? Basically, this involves x, y. This is a cross coherent covariance between X a feature in the response. And then this is the covariance for the x itself. Okay. After then, if you take a step back after looking at this and say, is this really a nice expression solution I wanted to have? There's one issue visit, which is actually if you solve this close from splashing, you have to solve the inversion of this matrix, XX^T. If we consider the complexity of that, remember what is your X? X is an n by m dimensional matrix. XX^T, is going to be a matrix of dimension n by n. And n is the number of features. If you have a lot of features, this matrix is going to be pretty large. In fact, the computational complexity of inverting a n-by-n matrix, is going to be n cube. Wow. So, we have a lot of features this is a pretty expensive step to solve. Why actually increments? You realize this may not be the best solution. In fact, not just because the computational complexity, but also, when you invert this matrix, if the variables are correlated, this matrix tend to be not invertible, so, it's going to create trouble. You will end up having the incorrect data to the solution, unlike what you're expected to be. To solve this problem, an alternative approach to use so-called the gradient descent. As you can see now, we're getting back to optimization again, of saying, if I were to solve this optimization problem, instead of solving directly, we can solve it indirectly, using this iterative algorithm. You can see now, instead of solving it once, you can start with some initial guess something, and then you start to improve it, by adding this part involves data, this is the gradient. Actually, here we gradient to improve it. This is getting back to optimization.

Recap, you can see what has happening from beginning to the last of what I just talked about. You have a practical problem. Now we start to build a model, in this case, very simple linear model with some probability assumption. Then we represented more compactly introducing matrix-vector notations and reducing problem for model fitting in solving optimization problem, and solved optimization problem becomes a separate issue and evolving its linear algebra techniques and actually consideration while you talk about how do we address computational complexity and possibly numerical issue that involves our equations and incrementation programming skills.



- Assume y is a linear in x plus noise ϵ

$$y = \theta^\top x + \epsilon$$

$\epsilon \underset{iid}{\sim} N(0,\sigma^2)$

- Assume ϵ follows a Gaussian $N(0,\sigma^2)$

$$p(y^i|x^i;\theta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(y^i-\theta^\top x^i\right)^2}{2\sigma^2}\right)$$

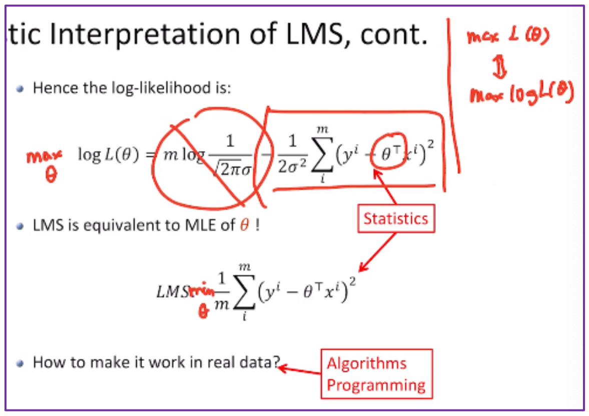
MLE maximum likelihood estimate.

- By independence assumption, likelihood is

$$L(\theta) = \prod_i^m p(y^i|x^i;\theta) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^m \exp\left(-\frac{\sum_i^m\left(y^i-\theta^\top x^i\right)^2}{2\sigma^2}\right)$$

Probability

Probabilistic Interpretation of LMS: Finally, why this is squared, here is the answer. Why do we solve this square as a way to go here. That's actually coming from the assumption we made initially about these errors. We made assumption that these errors follow the distribution of this. The errors are IID independent, identically distributed as normal and variable with 0 mean and variance squared, and If I write down the problem of finding the optimal theta, given observations by an x, then I will solve a so-called maximum likelihood estimation (MLE) problem. Maximum likelihood estimate. Basically, the likelihood function for here is going to be because the theta are independent, the product of the likelihood of a sample and each of this p, the likelihood for each sample is equal to this expansion, is basically the question of the normal distribution.



tic Interpretation of LMS, cont.

$\max L(\theta) \Updownarrow \max \log L(\theta)$

- Hence the log-likelihood is:

$$\max_\theta \log L(\theta) = m\log\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2}\sum_i^m\left(y^i-\theta^\top x^i\right)^2$$

Statistics

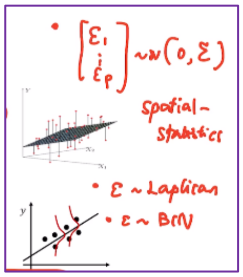
- LMS is equivalent to MLE of θ !

$$LMS_{\text{train}}\ \min_\theta \frac{1}{m}\sum_i^m\left(y^i-\theta^\top x^i\right)^2$$

- How to make it work in real data?

Algorithms Programming

You can see if I now simplify, first I'm going to take a log of my likelihood function that gives you the log-likelihood function. After some simplification, you can see this is why it looks like. My goal is to find the theta to maximize the likelihood function, by the way, there's another trick here. If you want to maximize L theta as equivalent of maximizing the log of L theta. After taking log, this is giving you the same solution. But this case, it's going to simplify the problem because before you have the product of this piece. Now I've taken log, it becomes the sum of these terms. Now becomes very simple. If you look at what's involved in this log-likelihood function, that basically was this term that doesn't depend on theta. I can ignore the constant term. And this term, is what I care about because theta is in here, that's my argument, and also, have the data. By looking at this, you can see solving maximum likelihood estimate of your favorite parameter, reduced to finding its square solution. Maxima in theta of this expression of the likelihood function, is the same as solving, minimizing the theta of this particular objective function. That's the second term of the log-likelihood function. Great, so, I have to find statistics mean explanation, interpretation inside of my least square problem. It's basic coming out of that. Besides this interesting observation, also, shows you that this can help you to extend your least square method. If for example, your data are not independent, and they are correlated, how do I deal with that? That basically can also, be extended, if you allow instead of your errors are independent identically normal, you could make them a joint distribution so, you collect all the errors together, and you could make assumption that this follows a multivariate normal with some covariance matrix. This is a very common trick in spatial data modeling. So, you can extend it. What if your errors are not normally distributed? What if they are say some bizarre distribution like Laplacian distribution? Or maybe they are binomial, they are binary, and so, and so. These are all reasonable errors. If you, for example, for your application, you followed different distribution, that you can follow the same machinery , derive maximum likelihood, and then you can see in the end becomes a different kind of least square problem to solve.



- $\begin{bmatrix}\epsilon_1\\ \vdots\\ \epsilon_p\end{bmatrix} \sim N(0,\Sigma)$

Spatial-Statistics

- $\epsilon \sim$ Laplican
- $\epsilon \sim$ BrN

My point here is flat mark. You can see solving a machinery problem was involved in here. We will start with a real problem facing the creative part is how do I build a mathematical model from that, and then when you solve a machinery task, you will see these two coming together from probability, linear algebra, optimization statistics coming together. Also, I would make incrementation. I think that's a good point to end. What is the questions about? I have briefly touched upon, what's going to be covered. What are some examples, modeling applications and, hinted upon the topics been covered here. The last example is really an instance I want you to understand. We want to train you to get this skill of solving machine learning problems, how to approach it, and how to understand how these different pieces fit in together to solve this overall problem. It's going to be some mathematics, some modeling, some statistics optimization, and also, implantation, in code and data analysis. I'll come together. All right, That's it for today. Thank you.