Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

Week-1

Lecture -4: Basics of Optimization Problems

So this course is going to be about distributed optimization but before we get into the distributed aspect of optimization, so we will review the basics of optimization right. So basics of convex functions, convex sets, and so on. So just to give you a brief outline of what we are going to look at in this lecture. So we are going to look at optimization problems in general. How do we mathematically formulate optimization problems? Then we are going to look at something called convex sets and convex functions. then there is going to be a discussion about certain operations that preserve convexity.

So, if I give you a set of convex functions can you come up with a new function that is also going to be convex that is based on these original function. So, operations that preserve convexity then first and second-order conditions for convexity. And if you remember in the last lecture we mentioned something about strictly strongly convex functions versus non-strongly convex functions. So, in the same spirit, we are going to look at functions of the which are called strictly convex functions.

And we will look at the importance of studying strictly convex functions and then also strongly convex functions. So, these are going to be the topics of discussion for today's lecture. So, how do we mathematically define an optimization problem? So, typical

$$\min_{x\in\mathbb{R}^n} f(x)$$

optimization problem has this kind of flavor. So, you want to minimize or maximize a particular function f of x, x lies in Rn. So, if I if I give you a problem of this form just this this alone and this this would be called unconstrained optimization right because x can take any values and this is called unconstrained optimization or an unconstrained optimization problem.

But most real-world optimization problems are in fact I mean they have some implicit or explicit constraints right? So, you can specify those constraints and it would be subject to certain constraints of this form. Let us say some function hi(x) less than equal to 0 and they can be multiple such functions. So, let us say there are m such inequalities. So, these are called inequality constraints and then you can also have something called equality constraints. So, for all j in 1 through some r.

$$\text{s.t.} \quad h_i(x) \le 0 \quad \forall i \in \{1, 2, \dots, m\}$$
$$l_j(x) = 0 \quad \forall j \in \{1, 2, \dots, r\}$$

So, this is a typical definition or formulation of a constrained optimization problem. Is this clear? So, h i's are your inequality constraints and l j's are your equality constraints. So, what is f here? Cost or you can say it is an it is basically your objective function. So, f is called objective function. So, this is objective function.

objective function x is your it's called decision variable or optimization variable and in this case x is a x is really a vector right? So, x has n components x1 x2 ..xn. So, basically it is you have n decision variables or n scalar with decision variables because x lies in Rn. So, why did we just mention like can maximization be an optimization problem? So, why did we write this particular form and not maximization? So, you can relate minimization and maximization as. So, minimizing f of x is equivalent to maximizing negative of f of x.

right. So, that is why I mean this this in itself is a general form of mathematical optimization problem. In certain cases when f happens to be convex and h i's or their inequality constraints these functions h i also happen to be convex functions and l j's are linear equality constraints. So, this is called a convex optimization problem ok? So, convex we still do not I mean we still have not introduced what convex functions and convex sets are, but just for now we. So, convex optimization problem is when your f is convex function your inequality constraints h i's are convex functions. and l j's are linear inequalities.

So, we call this convex optimization problem yeah. No, so the function is a scalar function like I mean it is a scalar-valued function right? So, this can be an example can be f of x is let us say x 1 square plus x 2 square. So, you still get just one output. So, this output needs to be minimized. x is a vector here, x is an Rn, right? So, it has n components and those n components are x 1, x 2, x n, ok, yeah.

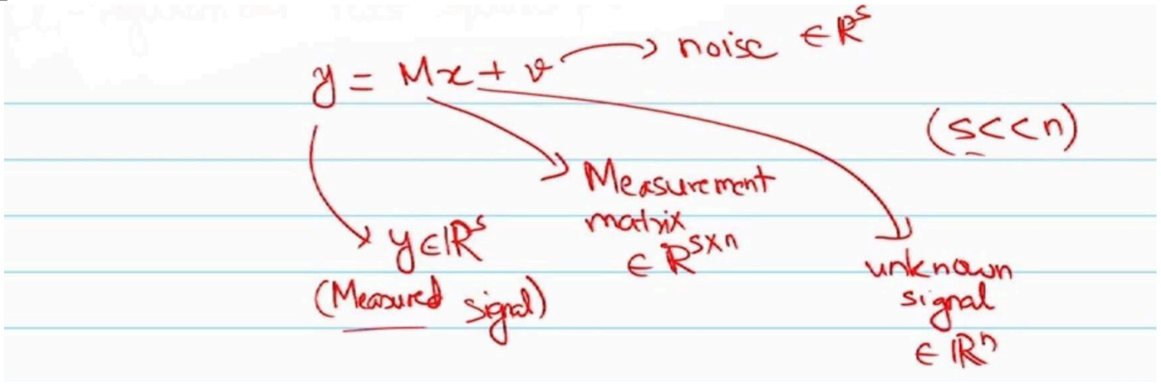
Right now we are just going to be studying optimization first and once we have a good handle on how to pose optimization problem, then we will move into designing distributed optimization algorithm. So, if domain of f on which this function f is defined, let us say this turns out to be Rn whole of Rn. So, then this particular set, so x in Rn such that hi of x less than equal to 0. lj of x equal to 0 for every i,j if i,j defined in this set over here.

$$\underline{\mathcal{X}} = \left\{ x \in \mathbb{R}^n : \ h_i(x) \le 0, \ l_j(x) = 0 \quad \forall i,j \right\}$$

So, this particular this particular set is called. Yeah, feasible solution set right, it is a feasible solution set ok right? So, any feasible point. So we are trying to find the point at which f get minimized, but then any feasible point would be defined by the set of points for which this these constraints are satisfied right? So this is your feasible solution set within this feasible solution set we are trying to find an x star which basically minimizes your function f of x ok. And there are several algorithms through which you can do that I mean they are first-order primal-dual algorithms first-order primal-dual methods, then there are second-order methods as well which uses the Hessian information. So, how many of you have worked with neural networks? So, you must be aware of something called LBFGS for instance as one of the optimizers or Ada Hessian right? So, Ada Hessian would be an example of a second-order method which uses Hessian information or Newton's method that that is again an example of second-order method.

First-order method would be where which basically just simply relies on the gradient and does not use the Hessian of the function. So, those are those come under first-order method. So, we will look at these algorithms in the latter half of the course or at least in the subsequent lectures, but I mean to find an optimal solution or in this feasible solution set we use several algorithms. So, let us look at few examples of like common optimization problems and we will also try to relate this to how this these examples can be viewed in a distributed setting too ok.

So, an example. So, L1-regularized least squares problem. So, what do we have it what do we have here? So, you have some measurement y in some let us say y in Rs ok? You have some measurement y. So, y is your measurement from measured signal M is your measurement matrix which is known, it may be so in some s cross n and v is some noise. we have a device that measures some noisy signal x and gives some output y. So, x is something that we are trying to estimate based on the measured signal y and v is some noise and this noise is also in Rs and this x is an unknown signal that we are trying to measure or we are trying to estimate rather and this is in Rn and typically in such problems we assume that s is much much smaller than n.



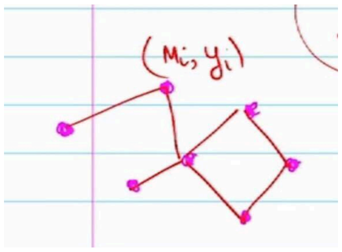
so what we can measure is a I mean so this measured signal y it belongs to a Rs which so s you assume s to be much much smaller than m. So, if have you seen this problem in any context? So, have you heard of things like sparse recovery or compressive sensing? So, the idea is we want to find out x right. So, we want to I mean noise is something that we do not know anyway. So, we want to minimize we want to find x in Rn such that based on our measurement y and the measurement matrix n. So, this difference is minimized right we assume that noise is anyway small.

$$\min_{x \in \mathbb{R}^n} \|\underline{y} - Mx\|^2$$

So, this is what we want to find an x which minimizes this difference, but if you have a scenario where s is much smaller than n. So, this is if your signal is a sparse basically then we add something called L1 penalty constraint. So, only the relevant components of x is what I mean basically are going to be picked up most of the entries in x are going to be 0 and this is when we add this L1 penalty here. So, this kind of optimization. So, this is an example of an optimization problem without the constraints.

$$\min_{x \in \mathbb{R}^n} \|\underline{y} - Mx\|^2 + \|x\|_1$$

So, there are no constraints here we want it is an unconstrained minimization problem and it is it is very common in domains like a print applications like compressive sensing or image processing and this particular L1 penalty is added to ensure sparsity. So, this ensures sparsity. So, if your signal is sparse, if you know that only a certain entries are going to be non-zero, every other entry is largely going to be 0, then you add this L1 penalty right? When you train neural networks we add an L1 constraint as well right sometimes on the weight parameters because we just want to make sure that the weights which are relevant are going to be non zeros rest of the weights are going to be zeros right and this is pretty much the same thing this is like a mean square loss and plus the L1 constraint okay. So, this is an example of an L1 regularized least squares problem okay.



So, how would this look in a distributed setting? So, let us say I have a network now ok.

Something like this ok. It is a network and every node or every agent in the network measures this x, but I mean basically they are trying to measure x they have their own measurement matrix. Let us say that ith node has a measurement matrix Mi M sub i and it ends up measuring signal or the measured signal is yi right. The equivalent optimization problem for or in the distributed case would be we would want to minimize there are n

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{m} \|\underline{y_i} - \underline{M_i} x\|^2 + \|x\|_1$$

agents in the network, we would want to minimize let us use some other n instead of let us say there are m agents in the network. So, let us assume there are m agents in the network. So, every agent has its own measurement matrix M sub i and it for the same unknown signal x it ends up measuring y sub i right because it is going to have its own model here.

There is going to be v sub i noise in its own measurement. So, this is what we want to minimize. as a team objective right. So, we would want to come up because it is it is the same signal that every agent every sensor is trying to measure, it is the same x that every agent is trying to measure. So, we want to arrive at a common x, but then you want to at the same time you want to make sure that every agent's individual sort of error this residual error is minimized right? So, the measurement matrix Mi and the measured signal yi they are not known to central agent some central entity.

And the question is how do you solve such problems? Yeah. All these agents are independent and they have their own like I mean let us say they have their own sensors and they have their own measurement matrix. You can also think of this problem in terms of let us say you are trying to find a solution to Ax equal to b, but then every agent can only see a bunch of rows. every agent has its own A sub i right in that large matrix every agent can only see a bunch of rows and but then it is the same x that is shared across every agent and that is what they need we need to figure out right. So, if I if every agent tries to solve its own problem everyone would get its own estimate of x i what x i would look like right.

So, in a distributed like and another way to sort of pose this problem is. So, minimize x1, x2,..., xm sum i equal to 1 to m y minus Mi xi norm square plus xi norm is subject to x1 equal to x2 equal to x n right.

$$\min_{x_1, x_2, \dots x_m \in \mathbb{R}^n} \sum_{i=1}^{m} \|y_i - M_i x_i\|^2 + \|x_i\|_1 \quad \text{s.t.} \quad x_1 = x_2 = \dots = x_m$$

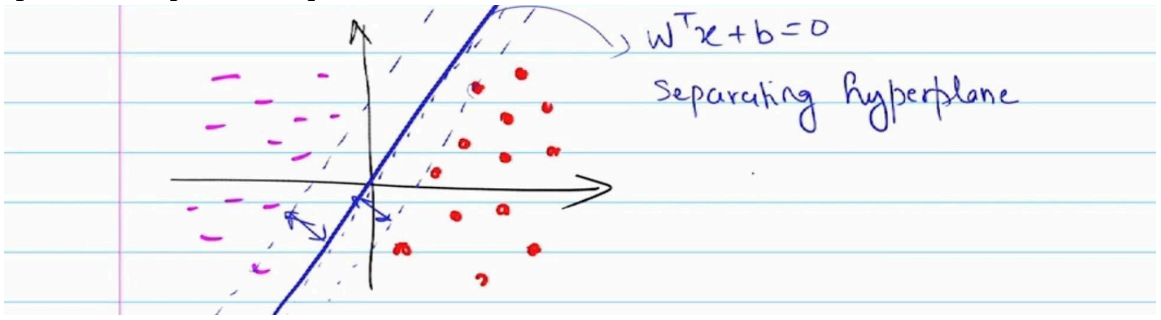
So, it is the same problem but now it's this problem is being expressed as a distributed optimization problem right? Now everyone would have their own xi. They would try to solve their own local optimization problem, but they would also try to run some kind of consensus on what they see and what their neighbors see and based on that they would try and make sure that this all the measurements or all the like all the estimates of the unknown signal x i's they eventually turn out to be same or consistent across all the agents and this becomes a distributed like basically writing the same centralized problem in a distributed manner. Now, but then it comes with an additional constraint here. right. So, the same constrained optimization problem which was a centralized optimization problem the unconstrained optimization problem which was centralized optimization problem becomes a constrained optimization problem because now we are trying to write in a distributed manner ok.

A sensor measures yi and has its own measurement matrix M sub i. So one dot in a graph is one sensor. So, if there is no like a let us say these sensors are not joined right.

So, there is. So, this edge indicates that this particular sensor can communicate with the neighboring sensor. So, there is no communication happening whatsoever, then I mean they would only know their y sub i, they would only know their Mi and they would end up computing their own xi. There is no way for them to know what everyone else is measuring. So, so there is no way for them to recalibrate what the overall x looks like right. So, let us say you have multiple temperature sensors installed in this room. Every temperature like every let us say sensor measures its own estimate of what the temperature is going to look like. But the overall room temperature may be different from what it has measured right. But by being able to communicate with the neighboring sensors you can get a sense of what everyone is measuring and then try and get to a common value. So, this basically edge indicates communication between different agents.

In the previous lecture we looked at a very similar problem where we had four different temperatures being measured right. And then everyone was sharing their own estimate of what temperatures would look like and the goal was to arrive at a common temperature based on, but then in this case we are not just like sharing there is a xi's is one thing, but at the same time they are also solving the optimization problem. So, it is not just about arriving at a consensus value, but it is also every time you are also trying to solve this particular optimization problem too right? So, it combines both optimization as well as consensus ok.

Another example is SVM. So, what is the full form of SVM? Support vector machines. Support vector machines right. So, we will try and formulate SVM as an mathematical optimization problem. again I mean not in a distributed not necessarily in a distributed setting, but we will as of now our emphasis is more towards being able to frame an optimization problem right.



So, let us say you have a set of so you have a set of two or maybe your two classes right and then your goal is to find a separating hyperplane not necessarily through origin your goal is to find a separating hyperplane that basically separates these two classes right and there can be multiple such hyperplanes right this is not unique I mean I can if this is let's say this is one of these separating hyperplanes that separates positively labeled class from the negatively labeled class you can also look at equivalent let us say a hyperplane which looks something like this right, slightly displaced from and this is also another example of a separating hyperplane that separates these two classes. So, in SVM, so what do we do in SVM then? I mean because the answer is not unique right, if this is a separating hyperplane maybe this is also a separating hyperplane. Yeah, so we want to find a hyperplane which maintains the maximum margin and what is maximum margin. So, basically we look at something called supporting vectors. So, from this separating hyperplane you look at the points on either sides such that I mean basically the corresponding parallel hyperplane kind of simply just touches one of the points or the nearest point to that particular hyperplane and you want to maximize this distance.

So that way you maintain a maximum robustness in like given this separating hyperplane you maintain maximum robustness on either side right. So if I am going to perturb this hyperplane a little bit if I instead of choosing this if I had chosen this particular hyperplane to be the separating hyperplane by just perturbing it little bit to the right-hand side I would have ended up incorrectly classifying this particular point. But that is I mean we want to maintain maximum sort of margin from either side and the in SVM the goal is to basically find this kind of separating hyperplane equation would be w transpose x plus b equal zero ok. So, this is the separating hyperplane ok.

So, for a point xi. So, xi's are these points and yi's are going to be labels of these points. So, these are let us say the labels are plus 1 here and the labels here are minus 1. So, for any point for which w transpose xi plus b is greater than 0. So, this means that the label yi

$$W^T x_i + b > 0 \Rightarrow y_i = +1$$
$$W^T x_i + b < 0 \Rightarrow y_i = -1$$

should be plus 1 and if this is less than 0 then you have you assign a label minus 1.
ok alright. So, what is the distance of a point? So, we want to maximize this particular distance right? We want to maximize in some sense we want to maximize the minimum distance. So, we want to maximize the minimum distance of a point from the separating hyperplane. that is our objective right in order to maintain maximum margin we want to maximize the minimum distance. Is this clear? So, what is the distance of a point from the hyperplane? Let us say I choose a point xi.

So, what is the distance? Here. So, di is equal to modulus of w transpose xi plus b

$$d_i = \frac{|W^T x_i + b|}{\|w\|}$$

divided by norm of w ok. This is the distance of a point from the hyperplane. and we want to maximize the minimum distance. So, the minimum distance let us say d is minimum over all points. So, the thing is I mean while this is fine I mean there is still some level of ambiguity right because you can make your w and b arbitrarily small or arbitrarily large and this is I mean this does not fix the scale issue right.

So, I mean you do not you still do not get a unique solution to it. So, we look for points such that the minimum distance this this is basically d times for the minimum distance d times norm of w that is equal to 1. So, that fixes the scale problem why again let us look at this particular distance right. So, d is what we are trying to maximize But if I look at this particular problem I can make my w and b arbitrarily small or arbitrarily large because it is also normalized by norm of w I mean there is no unique solution to it right? There are multiple solutions for which let us say if w and b are solutions 2w and 2b are can also be solutions and so on.

$$d = \min_i \frac{|w^T x_i + b|}{\|w\|} \qquad d\|w\| = 1$$

So, there is no unique solution to it right. So, in order to fix that scale issue we want to look for minimum this distance or the this d. this minimum distance point such that d times norm of w is equal to 1 ok. Is this clear? So, this is just to fix the scale problem, fixes scale. We care about the unique. So, if you look at just look at the d here right definition of d. So, it is w transpose x i divided by norm of w right. So, everything if you scale up or scale down. So, if w and b are solution 2w and 2b are also solution right. So, there is no way for you to find what is your I mean in some sense there is no unique solution to this problem. So, the problem is not well posed rather. What we are trying to find xi's are the points which are inputs and we want to make sure that for the w and b that we are going to find out first of all these make sense. So, w transpose xi plus b greater than 0 if that is true then yi should be plus 1. So, these should make sense and at the same time the minimum distance should be maximized. And now since the problem is not well-posed at least in terms of uniqueness we want to fix the scale. And in order to fix the scale we choose this particular criteria. So, having said that. So, what do we want to do? We want to maximize the minimum distance right maximize with respect to.

$$\max_{\{w,b\}} d = \max_{\{w,b\}} \frac{1}{\|w\|} = \min_{\{w,b\}} \frac{1}{2}\|w\|^2$$

So, what are my decision variables or optimization variables w and b is what we are trying to find. we want to maximize d and because of this this particular constraint over here this is equivalent to maximizing 1 over norm w right which is equivalent to minimizing half norm w square ok. If we want to maximize 1 over norm w, it is equivalent to minimizing norm w or norm w square. Minimizing norm w which is same as you minimize norm w square which is again there is a reason why we want to work with the quadratic functions because it is much they are much nicer to optimize and we are going to look at some glimpses of it towards the end of today's lecture that working with the. So, these functions they belong to something called the class of strongly convex functions and it is much easier to work with strongly convex function.

So, this is your objective function. Now, what are the constraints? Because of d times norn w equal to 1 we want to maximize d. we just now write everything in terms of w I mean. We still have to work with these two constraints though and these constraints will get fixed accordingly because now we know that the minimum distance is going to be at least 1.

$$\text{st.} \qquad y_i(w^T x_i + b) \geq 1$$

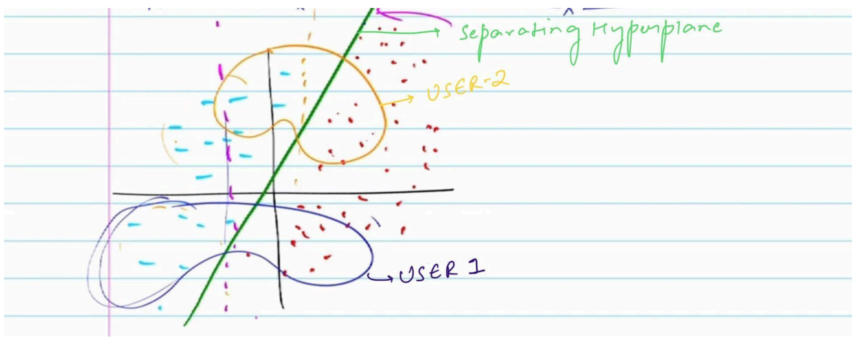
right. So, is there a neat way to write these constraints yi w transpose xi plus b this should be greater than equal to 1. Because the minimum distance we know is now going to be 1 and if this is positive yi is plus 1 if this is negative yi is negative 1. So, the product of these are always going to be positive and because the minimum distance is plus 1. So, here we are using this particular definition of d right. So our final optimization problem for SVM is minimize w,b half norm w square such that yi w transpose xi plus b is greater than 1, okay.

$$\min_{\{w,b\}} \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \quad y_i(w^T \underline{x_i} + b) \geq 1 \qquad \forall\, i \in \{1, 2, \ldots, m\}$$

 and these yi and xi are your data points right. So, xi is your given data point and yi is the corresponding label for that which is going to be either plus 1 or minus 1. So, when you try to fit a separating hyperplane in case of SVM you are given some training data points of the form (xi, yi) on those training data points. So, this is true for every i let us say you have m data points for every i in 1,2, … m ok. So, this these constraints need to be. So, you have m inequality constraints and these basically this is also going to be and you have m set of points in your training set. Is this clear? Now having learnt how to pose math like given a problem how to come up with a mathematical formulation for the corresponding optimization problem.

 Let us look at few more relevant concepts which are convex sets and convex functions. yeah as of now I mean we are not looking at any algorithm we are just trying to study the basic concepts of optimization in general. So, I mean initially we would start with centralized optimization and eventually, we would move to. So, for this for instance let me get back a bit. So, let us say the same SVM problem now why is it why does it become challenging when we try to solve it in a distributed manner.

 So, let us say I give you these set of points So these are positively labeled points and then at the same time I give you some negatively labeled points and looking at these points it appears that a good like separating hyperplane a candidate for good separating hyperplane should look something like this right. Now assume that this data is distributed across multiple users. So maybe the user one it has access to these set of points. And if user 1 ends up finding his or her own separating hyperplane, just based on these points, what would be the hyperplane it would end up finding? It would look something like this. which is way different from the centralized or the hyperplane that we are really looking



for right.

So every agent depending on their data distribution they would end up finding hyperplanes which look very different from the hyperplane that we eventually want to solve for. And this becomes a challenge when we again when we try to solve it completely locally. So that means this should also involve let's say I end up finding up like one particular hyperplane based on my data. Then I'm going to exchange some information with my neighbor to get a sense of what kind of hyperplane he or she is being able to find. And based on that, I'm going to maybe course correct myself and then try to find a better hyperplane and so on.

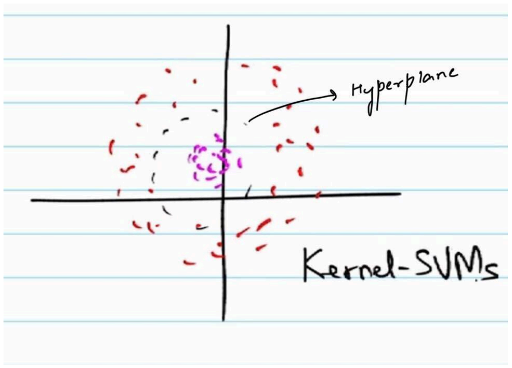
And eventually the same thing would be done by every other agent in the network. And they would try and arrive at this particular centralized solution. So the thing, the challenge with the, when you have data distributed across multiple users is, let us say for instance I mean so you understand that we are looking for this green separating hyperplane right depending on the data distribution, but if I look at the individual data distribution between like different agents. So, agent 1 for instance has these set of points. Now, for these set of points a corresponding separating hyperplane would be would look something like this.

which is way different from what we want, what we are aiming for right. And let's say another user, maybe another user has access to these set of data points. So they will end up finding this to be their separating hyperplane. So everyone's separating hyperplane looks different from the centralized solution. And it really depends on the data points that they have access to. Now if everyone just solves their own optimization problem and do not care about the optimization problem that others are trying to solve or the data points that others may have, this becomes a challenge to find, to basically come up with a good answer.

Just because this particular user hasn't seen points in this domain, it does not mean that this particular hyperplane is going to solve his or her own like optimization problem in future times as well right. Let us say tomorrow at certain point they get access to this particular data point may be a data point somewhere over here. This would be classified as a positively labeled point even though it should have been classified as a negatively labeled point right? So, the more the users you have in the network And the more the variety of the data is there in the network, I mean, if you are able to solve the centralized optimization problem, that would be a good or a very good estimate, like that would be a better candidate than trying to solve this problem locally based on a very small set of data, right? And that is a challenge when we work in distributed settings. So if everyone starts solving their problems locally, everyone would have their own estimates of what the separating hyperplane would look like. And this would be very different from an ideal solution. So not only they need to solve this optimization problem, they also need to exchange information or communicate with their neighbors so as to get a sense of how everyone else is doing. And based on that, they would try and cooperatively sort of try to arrive at this particular centralized objective. right because as of now everyone's hyperplane looks very different from what this should look like right. And again this is another challenge with distributed optimization problem when you do not have a centralized entity how do you guarantee that you end up solving the centralized objective function without I mean without all the agents having access to their neighbor's data or their neighbors neighbors data or the data of everyone else is in the network ok. So, usually then you allow some amount of error and you want to minimize that error, but this is this is the case when.

So, just to reiterate the question is in an exceptional case it may happen. So, for instance even over here let us say what if one of the data just one of the data point ends up being over here right. and in that case we cannot find a separating hyperplane that separates all the positively and the negatively labeled data points. So, in that case we I mean the optimization problem is somewhat different, but you want to minimize the number of mismatches or misclassifications. Yeah, yeah. So, if you are looking at this version of the problem where you I mean this where you have a hard constraint.

So, this is I mean you can then go to look at soft test frames, but if you have a hard constraint of this form then I mean such these kind of scenarios are not I mean you cannot handle. There are other ways I mean let us say I mean this is just one of the examples, but I mean you can also have scenarios of this form where you have let us say all the positively labeled data points in distributed in an outer periphery and maybe negatively labeled data points somewhere over here. So, no matter what kind of hyperplane you end up drawing you would never be able to separate these points right. So, in that case I mean ideal instead of the hyperplane should look something like this



So, we use something called kernel SVMs for solving such problems. you would try and project these points in in a higher dimensional in an abstract higher dimensional space where these points are linearly separable draw a hyperplane there and then sort of project it back. So, using kernels. So, we use something called kernel SVMs, but yeah I mean there are cases where you may not be like this is a very simplistic scenario just to motivate you how to formulate mathematical optimization problems, but I mean there are settings where you obviously cannot separate all points perfectly and in that case you would obviously have to allow some amount of error right. Just when you train a neural network for instance I mean you never get 100 percent kind of accuracy right, you always get some maybe even with data sets like MNIST which are relatively I mean nowadays at least relatively simpler I mean the accuracy is still around 99.5, 99.6 percent right. there is always some misclassification and that is fine and sometimes those misclassifications can also be due to the fact that a particular data point has been incorrectly labeled when it should have been classified it is being classified correctly, but the true label that was assigned to it that was incorrect right. So, you cannot avoid such scenarios. Thank you very much.