devices, all of them smart and smart basically means that they're collecting your data, but also that they're trying to optimize algorithms on that data.

So on the one hand, this is good, because it gives us the opportunity to better understand certain processes, training very rich models, voice recognition, all these kind of potentially useful automations are now possible because we collect so much more data.

On the other hand, it also is kind of potentially a nightmare because if you implement it wrongly all your data can be potentially leaked.

The algorithm may perform super unexpectedly in some unseen situations so there are definitely definitely things that you should pay attention to when moving towards this data-driven world.

So the way we will study this kind of concept of these interacting algorithms is in a general optimization setting where our goal is to minimize some function f of x and with x lying in some constraint set curly x and this f is then our objective function.

For simplicity of the analysis that the radical results I will show we usually take to be a Lipschitz function and either convex or strongly convex and what we're then looking for is the minimizer so the optimum which we refer to to as X star.

So given all these kind of potential issues that we might have with this in this data driven world, what could we say are the requirements of a good algorithm.

Well, first thing is robustness.

We want it to be robust to certain unseen datasets.

We don't want it to perform weirdly on situations the algorithm hasn't seen before.

We also want it to be computationally efficient.

And why do I say that.

Well, obviously storing data, processing data, it doesn't come free.

It comes actually at large costs of large data centers, energy, it costs a lot of energy and stuff like that.

So ideally we would also want to make it as computationally efficient as possible using the minimal data that it can while still converging as fast as it can.

And potentially also to simplify this computationally efficient concept, we want it to be trainable in a decentralized manner.

So basically if we can leave the devices, for example, sorry, the data on each local device, this might save us having to store all the data on a central server, having to transfer all the data to a central server.

So if we can train it locally on every device while still converging goods, that would be nice.

So what do we usually do when we go about optimizing some objective function f of x.

Well, we take whatever optimization algorithm we like to take.

we initialize some points of this optimization algorithm somewhere randomly, and then we optimize, and potentially we do this more times.

So what we also could do is exactly do the same thing, take our usual optimization algorithm, again, initialize multiple particles, but while we run the algorithm, we could also let the particles interact with each other.

So all the different agents, particles, whatever you wanna call them, we have initialized, all of them are performing a certain optimization, but at the same time we also allow them to exchange information between all the particles.

Why would we think that this is potentially useful.

Well, if we manage to define the interaction in a useful good way, what could happen is that first of all we are exploring the loss function more effectively.

So because different particles are in different places of our objective function or loss function, it could happen that we are studying this loss function more efficiently, which could in turn result in faster convergence, convergence closer to the optimum, or convergence to a better minimum.

And what I mean with potentially a better minimum could be a more robust minimum.

So a model, a parameters of your model that are more robust to certain outliers in the data, for example.

So this is kind of the underlying idea.

This is why we were thinking, well, maybe interactions could be interesting to study.

We're not the first ones to study interactions.

So just very briefly, I would like to mention some previous work.

Interactions have had benefits when your goal is not necessarily to optimize with more to sample from a certain distribution.

If you define interactions in certain smart ways, you can can achieve exponential convergence to some distribution that you want to sample from.

We call this the invariant measure in this particular case.

Could also lead to reductions in the asymptotic variance, or the variance of how far your samples are from the samples from the actual invariant distribution in terms of their variance.

And also, it has been shown that when optimizing, interactions have been able to converge to more robust, or in this case, I call it flatter, minima in non-convex settings.

This is this third paper where they have also studied elastic averaging SGD, which is kind of a interacting particle optimization method.

You could kind of see it like that.

So there are definitely some signs in the idea that interactions could be useful in the settings that we're gonna be looking at, which is more responsible optimization.

So given this introduction, I would like to now discuss a bit of the algorithm that I'm going to be presenting the results for.

Specifically, I'm not looking at regular gradient descent, but I'm looking at mirror descent.

I don't know if anyone is familiar with mirror descent, but basically, mirror descent is a generalization of projected gradient descent.

So basically, mirror descent is useful in among other the case where you want to sample from some-- sorry, optimize such that your parameters are lying in some constraint sets, currently x.

And the way mirror descent works, very, very briefly, I won't go into all the details.

But basically, mirror descent relies on some kind of mirror map, which is going to help you pass between your constraint set or the primal set, whatever you want to call it, and your unconstrained set or the dual set.

So this passing between the two is done using this mirror map.

And so basically what happens is that we project our particle into the unconstrained set using this gradient of the mirror of the convex conjugate of the mirror map.

Then in this unconstrained slash dual space, this is where we take our gradient step.

So this gradient step is taken in a dual space.

The gradient step itself remains as it usually is.

And then we project this particle z, which was in our unconstrained slash dual space back into our primal space, such that it is also going to be lying in our constrained set curly x.

And this projection is done using this fragment divergence thing, which basically is a different way of measuring the distance or divergence.

So this is briefly mirror descent.

Starting your primal space, project into your dual space, take your gradient step in a dual space and project back into your primal space using this fragment divergence.

Well, here I explain basically again the same things.

So we look at the continuous time version of mirror descent which is given as follows.

So this variable set is updated using this gradient step and then we're doing the projection back into our primal space using this convex conjugate of the mirror map.

An additional technical thing is the assumption that we assume that this guy directly maps into the constraint set so we're not doing this additional projection step.

This is just for simplicity of the analysis.

So that is mirror Now, why is mirror descent useful.

This is basically a standard convergence result of mirror descent for a convex objective function.

So if f is assumed to be convex, then basically you can achieve the following convergence results.

So your, the distance between f at some x, t and the distance between that and your f at your optimizer x star, the integral, the average integral that's called like that of this thing is bounded by this Bregman divergence term and divided by 2t.

So basically what this shows us is first of all, obviously the higher we make capital T, so the longer we run our optimization algorithm, the smaller this term on the right is going to be.

But we also see another thing, because if we would have used a standard gradient descent, this d phi curly X would have been a Euclidean norm.

And so the benefit of mirror maps, of mirror descent is that if we can choose this mirror map in a good way, this d phi of d phi curly X could potentially be smaller than the D5 curly X of the regular Euclidean distance metric.

So this is kind of the underlying point.

If we manage to find a mirror map that works well for a problem, and then what means works well is actually relatively complicated.

But basically, the idea is if you do have a mirror map that fits your problem very well, you might be converging faster than regular projected gradient descents because you have this additional flexibility of choosing a mirror map that suits your problem, as opposed to just using the Euclidean distance to perform the projection.

Okay, so that is that for merely sense, a very short introduction because merely sense is a big topic, but hopefully this just gives them the basic intuition behind the algorithm.

So now the question is, how do we define the interacting algorithm that I keep mentioning, but haven't yet shown.

So now basically what we wanna do when we refer to interacting parts optimization that instead of using independent particles, we want to use interacting particles.

So essentially for our dynamics, this then means the following.

Each of our particles, so each of our z, t, i's, where i is the particle number, and we say we have capital N in total, each of those particles is moving as follows.

It is first of all taking the very standard gradient step into the over the objective function f, so the function that we're trying to optimize.

Then comes the interaction part and here we assume a relatively standard or basic interaction where basically two particles are exchanging their values, so basically just giving their value to each other.

If their aij, which is their interaction matrix, is one.

So if the particles are connected in our underlying graph, aka they are exchanging their values.

Then we have this parameter theta that controls the interaction strength.

So how much do we value that the particles are going to converge to the same value versus how much do we value that they should converge to the objective function f.

And then lastly, we have this noise term.

Now why do we consider noise.

Just because it's interesting, because in settings where you are only able to compute your gradient over, for example, a sub-sample of your data, you kind of have noise in the algorithm as well.

So then the question is what is the influence of this additional noise if we have it in our scheme.

Now the thing I do want to mention here is like I said this is the most basic, basic, basic form of the interactions that we could define.

In the future I would like to look into more complicated ones because you could make this function that defines the interactions between two particles as involved as you want to make it.

This is just a simple one where they are exchanging their values if their interaction matrix, if the value at aij is non-zero.

Okay, so in the remainder I will work with a vectorized form of these dynamics.

So basically just very briefly you just stack all of your particles in this bold Z thing.

Then you define your graph Laplacian.

So to replace your interaction matrix to vectorize that part we define this graph Laplacian.

L and then we define the scurly L as the Kronecker product with a identity matrix.

And then these are basically the dynamics that we have, just easy to work with.

Basically, this V is our stacked gradient, this is our interaction, the LZ term is our interaction term, and then then we have the browning motion, just a stacked browning motion.

All right, well, this is basically all the stuff that I already said, but this is the way the interaction works is that if they are connected to particles, so if aij is not zero, then they are exchanging their values.

Noise, I also briefly mentioned why we consider this, why we want to consider additional noise.

Just to go into a little bit more detail on this, noise is interesting.

Well, first of all, as I said before, these inexact gradient estimates, so when you're using stochastic gradient descent, your noise, your gradient is going to contain additional noise.

Now this noise, by the way, is not fully additive and not as simple as we have it in this term, like the noise that you get from stochastic gradients are actually a bit more involved but just for simplicity we say additive browning motion.

Also noise can arise from errors in your communication graph so imagine for example that your particles or your agents are actually smartphones and they're all trying to collaboratively train some model.

It could be that certain phones are turned off at certain times so basically this is corrupting kind of your communication channels.

So this is also where the noise could come from.

But you might also want to actually add noise into your algorithm, which is the idea of this differential privacy, which basically says that if you add the right distribution and the right amount of noise, the model that you train might actually be more secure, so not leak information about the data it was trained on.

So again, in our case, very simplistic noise, just additive Brownian motion, which is the most simple one you can consider.

All right, so this was it for the relatively long introduction as to why interactions and how do interactions look.

Let's look into the first benefit that interactions might give us.

And the first benefit is this idea of variance reduction.

So if we make a relatively long proof for the convergence of interacting stochastic mirror descent and if we assume that our underlying objective function is convex, now basically this is the result that we get.

Now what do we see here.

Basically for this first term on the left is as usual our distance between f at some particle x_ti and f at the optimum x star.

Again, averaged over time.

Then the first term on the left is our standard convergence term.

Remember we also had this term just when we studied regular mirror descent, without interactions and without more than one particle.

This term has this Braggman divergence term here, which shows us the benefit of mirror descent, that we have this additional flexibility in choosing the right mirror map so that this term could be smaller than if we were just using projected gradient descents.

And this term is decreasing with time as I also said before.

And then come the more interesting part.

So basically we have this thing where we see that the noise, so our sigma squared, is divided by two times n times this Hessian of the mirror map term.

And basically this term is also one of the more interesting ones, because essentially what this shows us is that the variance, so the effect of the noise, could be reduced by a factor of capital N if we are using more than one particle.

Now this only happens if the remaining two terms, so this one and this one, are sufficiently small, because because otherwise you just lose out on this effect again.

And so what are these two terms.

They're basically fluctuation terms.

So this sets with like this swivel thingy on the top, basically says how far is each particle from the particle average value.

So basically how far away is one particle from the average of the system.

So basically if we can also guarantee that these fluctuation terms are bounded, then what is the benefit of interacting mirror descent.

Well, the benefit is that this noise term can be reduced by a factor of capital M, AKA we would be able to converge closer to the actual optimum of our function because this thing is gonna be smaller if this thing is small.

Okay, this is basically everything that I already said.

So let us look into this fluctuation term.

So remember that I said, if the fluctuation can be made sufficiently small, then we are getting this effect of variance reduction.

But if not, then we might even be worse off than if we were to just use one particle.

But luckily under certain assumptions, we can actually show that also the interaction term can be made sufficiently small.

And specifically, we have this following result that the unexpected value, our average of the squared fluctuation term in the dual norm, that's what this star is for, is first of all decreasing with time.

This is what this first term shows us.

And second of all, if we make the interaction strength large enough, so this theta that I had, let me go back briefly, that I had here, if this theta is sufficiently large, then note that also this second term, so the one with the decay divided by theta times the sigma squared, can also be made sufficiently small because if we increase theta, this term decreases.

So what this says is that the first benefit of interacting merely sense could be that if your function is sufficiently convex, ideally even strongly convex, and you choose your interaction strength to be sufficiently large, then your variance could be reduced compared to running just one or independent particle.

So this is the first benefit of interactions that they may reduce your variance, AKA allow you to converge closer to the actual optimum of your objective functions, so the X star.

Now, this all looks nice and everything, but I do wanna mention that this analysis is not perfect.

The reason it's not perfect is exactly for this mirror map thing, because basically what I implicitly assume when I derive the results that I showed is that not just the objective function F is strongly convex, but actually that dysfunction currently V, which is defined as a composition between your, the gradient of which is defined as a composition between your gradient of F and the gradient of the mirror map.

So this makes things a little bit less pretty in a sense that I have to make an additional assumption that this composition of the two functions is actually strongly convex while ideally we would only like to assume that our function F is strongly convex.

But on the other hand, it could also be actually a benefit.

Because if you choose again this mirror map in the right way, you might actually be able to make this function currently V-convex even if your f is not.

But this is something that I haven't been able to show up to this point.

So this is just kind of a side note of the fact that it's all not very perfect yet.

OK, so that is that.

Let's look into a numerical example, and specifically here a linear regression setting.

So basically we're trying to minimize this Wx minus b, the distance of that in L2 norm squared, such that this curly x is lying in some constrained set.

Specifically, we choose it to be the simplex, the unit simplex.

And then to make it a little bit more exciting, we also say that this matrix W has a relatively high condition number so that the problem is ill conditioned.

So to make the optimization not as trivial as it might have been otherwise.

And what do we see.

Well, these are some of the results that we would get for this particular setting.

On the left, you see the actual convergence of in your loss function value.

So you see the blue line is regular stochastic mirror descents.

The orange line is stochastic mirror descents interacting with 10 particles, and the green line is with 100 particles.

And basically what you see here is that both our orange line and our green line are actually able to converge closer, lower in this loss function value, meaning that they are indeed, as we would have expected, able to converge closer to the optimum.

This is on a logarithmic scale, so this is why it might not look as impressive, but basically the point is that they are able to converge closer to the optimum.

And then on the right, we also plot the histogram of the samples that we get after a certain point in our initialization.

And also here we see that our interacting particles have less variance, so they're more centered as opposed to the stochastic mirror descent or the interacting with few particles.

So kind of confirms the results that we showed theoretically that indeed if we define interactions and assume that the interactions are sufficiently strong, then yes, we could potentially converge closer to the optimum.

All right.

So this is a bit of a more interesting problem.

It is this kind of traffic assignment problem.

So basically-- well, let me just explain it kind of briefly.

The idea of traffic assignment is to compute the optimal path between two nodes in your graph.

So we just assume we have a certain graph and we wanna see how do we get most optimally from A to B.

This problem, if you define it in a certain way, you can show it's this convex optimization problem and you will have a simplex constraint on your thing you're optimizing over.

So your X, this one I will skip because it's just a lot of technicalities as to how we actually define this problem.

But basically, two results.

This first one is interesting because it actually shows us the benefits of mirror descent as opposed to gradient descent.

So gradient descent, remember, uses the Euclidean L2 norm to make projections, so to project back into our constraint set, which in this case is the simplex.

And as we see, it converges just a whole lot slower.

But mirror maps, if we choose the right mirror map, merely sense is able to converge a lot faster.

So this is just validating what I said in the beginning that there is benefit in this additional flexibility of choosing the mirror map.

And let's look now at interactions.

Basically the same kind of histogram as I showed in the first example.

But here we see that the more particles we use, the closer together they're gonna lie, thus the smaller the variance is.

So again, validating our theoretical results that we can converge closer to the optimum and closer to each other.

And then the last example, which might be interesting from the stochastic gradient descent perspective.

As I also mentioned sometime in the beginning, basically when you are optimizing over some objective function that consists of a sum of other objective functions.

So let's say a sum of the functions over your other data points.

If you have a very, very large amount of data, you might wanna say, I'm not going to compute this big sum every time.

I might just compute it only a subset of this sum.

This is fine, but then you would add additional noise into your algorithm because obviously your gradient is going to vary in each iteration that you have.

So let us see what is the value of interactions in this particular setting.

So the reason I mentioned this explicitly is because here I'm no longer adding in this Brownian motion noise in every step of the iteration.

I just assume that the noise is implicitly coming from the fact that we're only computing the gradient over a subset of the data and not the whole data set.

So this makes the noise that we're having in this algorithm a bit more complicated since it's not just going to be additive Brownian motion potential.

So yeah, this is basically explaining the same thing.

we again consider this linear regression model where we now compute the gradient over this subset of the data where this curly S is the subset of the data or subset of the objective functions.

And then this is our objective function, sorry, this is our optimization algorithm.

So every particle is getting updated by this step in the direction of the negative gradient computed over some subset of the data plus, as usual, our classic interaction term.

So again, noise is implicit in these gradients.

So what are the results.

Well, these are the results and specifically the figure on the left is nice because it basically shows us that interactions work.

So what we see here is this blue line is the setting where we have, where we are computing the gradient over a batch size of 10.

So using 10 subsets of the objective function to compute the gradient over.

and we're not using any interacting agents, so we're just using one independent particle that is not exchanging anything with any other particles.

And then we have the green and the red lines, which are the, well, let's say they're both kind of the full batch setting.

So in both cases, we're computing the full gradient.

So this comes at the computational cost that we are now computing the gradient over all of our data samples instead of just this batch of 10.

This obviously converges better.

So the green and the red converge closer to the optimum than the thing with the blue one where we just have a lot of noise, but we're not doing anything else.

And then we have the orange line, which is the most interesting one, because this is where we still assume that we're computing the gradients over a smaller batch size, but we're also using interactions between our particles.

And specifically, we're selecting 10 particles and we're letting them interact.

And interestingly, this is able to converge more or less the same as the red or the green line, even maybe a little bit faster if you see that it's kind of below the green and the orange line.

So again, validating our fuel that interactions is an alternative to computing the gradient over a larger subset of your data.

Specifically, we can still use the small subset of the data that we're using, the small batch size.

But if we define these interactions, then basically we can still converge closely to the optimum.

So that is that.

We can also look at this from a different perspective.

So as I said in the very beginning, when I was talking about this related literature that is out there, interactions have been used also in the setting of sampling.

So basically in sampling, the goal to sample from some invariant distribution.

And ideally, interactions would then help you also converge closer to or better or faster to samples from this invariant distribution.

So we can also study the performance of our interacting stochastic mirror descent algorithm from the perspective of this invariant distribution.

So instead of asking the question of how fast are are we going to converge to the optimum f of x star.

We could also ask the question is, how soon are the particles going to be samples from the invariant measure.

And I will explain in a bit why this is kind of the same question that you're asking.

So we know the invariant distribution that our algorithm is going to converge to if we sample sufficiently long.

Specifically, this invariant distribution looks as follows.

So basically it is this one over some normalization constant z times the exponential of minus two over our noise squared multiplied by first of all, the sum over our objective functions, or actually remember that this curly V was the composition of our objective function in our mirror map because we're working with mirror descent here.

But so basically some of these kind of objective functions curly V.

And then additionally, we have this term that comes from the fact that we have defined additional interactions.

So if we would have not have had interactions, we also would have just have had the summation over the objective function, but because we have defined also interactions, we get this additional kind of like a regularization term maybe so we have the Z times the graph Laplacian times Z again.

So why did I say that basically studying the convergence speeds to this invariant measure, could be the same as asking the question of how fast are we going to converge to the optimum.

Well, basically because finding the modes of this invariant distribution, this would mean that we have to solve the following optimization problem.

The mode is basically the z star where our distribution, so this thing w, so the thing that is inside our exponential, is going to be small enough, so as small as possible.

So, and this, if we just look at the way this W thing is defined, is the sum of this V and the sum of this interaction term.

So basically finding this mode of the invariant measure, so around where are our samples mostly coming from, is the same as saying that this Z star is the minimizer of our objective function V.

So to summarize this, what this means is that if the samples have converged to samples from the invariant measure, they are lying around this mode of the invariant measure, and this mode of the invariant measure is in turn the actual minimizer of our objective function.

So this is kind of the way these two things are related and why you could also use the invariant measure to understand the convergence.

Why would we want to do this.

Why would we care about this invariant measure.

Didn't we already prove convergence.

Blah blah.

Yes, we did.

But the idea is that also studying it from the sampling perspective could potentially even give us better convergence rates or convergence under looser assumptions on the objective function and things like that.

So this is the potential value of also looking at it from this different perspective.

And there's this whole bunch of theory, which I really won't go into here, but basically it shows that if your objective function, or sorry, actually your sum of the objective function plus your interaction, so this w thing, if its Hessian is positive semi-definite, then you can obtain exponentially fast convergence to the invariant distribution.

So the distribution of your particles at time t is going to converge exponentially fast to the invariant distribution that we have at infinity.

So you have this exponential convergence result essentially.

We can replicate this result for our particular setup and basically show that in this case of mirror maps interacting mirror descent, you can also converge exponentially fast to these samples from the invariant measure.

I think I will leave it at that.

You can also use the invariant measure to study more things like how far away are we actually from the mode, so how for our samples going to be from the modes and stuff like that.

But I just want to mention this just so you know there's also this other perspective of looking at convergence which from this invariant measure perspective which maybe could be interesting.

So then I come to the last part of the things that I want to show in this particular talk which is another potential benefit of interactions which is this concept of flatter minima.

So in the beginning as well, I said that given that we're using all these different smart devices, all of which are training certain algorithms and using certain data and giving you certain suggestions as outputs, we have to understand the performance of these algorithms on kind of unseen data.

So these in other words amounts we have to make sure that the algorithms and the models that we have are also going to be robust and they're going to perform as we want them to perform on unseen unknown situations.

So, then I come into this concept of flat minimum what I mean when I say that basically if you were to look at deep learning neural network stuff.

You could see that your loss function is something that is relatively complicated, probably even more complicated than this picture shows, because this picture is just in three dimensions, while in essence your neural network is probably a lot more high dimensional.

A thing here to note here is the fact that the loss surface is very non-convex, and also we have different minima with different properties.

And this kind of heuristic idea that I think is still relatively popular in deep learning, at least for sure was popular a few years back, is that flat minima, so minima that are more flat as opposed to very sharply changing their value in your loss surface space, that they will also generalize better.

First time this was proposed was very, very long ago in this paper on flat minima, but it has also been studied relatively recently still.

Why would those slats minima also correspond to better generalization slash better robustness.

The underlying idea is that if the input data changed a little bit, so if you're inputting slightly different data into your model, you don't want your loss function to change and not wanting your loss function to change would mean that you are not changing the output of your neural network too much, which in other words means that you're flat in your loss surface space because you can be in different values of your inputs, but still kind of have the same value of your loss function AKA your actual function output.

So then why am I mentioning all of this.

Well, because potentially interactions could also help you in this idea of robustness.

Specifically, what I showed before is when we were looking this invariant measure, we showed that with interactions, our invariant measure was actually optimizing this function.

So the sum of our objective functions plus this kind of regularization term, x times interactions times x.

Well, if we did not have any interactions, it would just be optimizing this thing.

So basically just your f of x.

So what you could say from And this is obviously very heuristic because I don't have any very rigid proofs for this.

But basically this could potentially show that in non-convex settings, in convex settings this term doesn't matter as we saw before because we were still converging to this, we are still looking for the same optimum.

But in non-convex setting, this regularization term might potentially be doing something to smooth out your very non-convex functions so that you're gonna converge to a minimum which may be flatter and thus more robust.

So let's look at a very basic toy example or relatively basic toy example.

This Mueller Brown objective function is just one non-convex function that is still relatively easy to understand the structure of.

It has several saddle points and several locominima.

And then the question is what are we converging to in this objective function if we are using interactions.

Well, this is the answer which may not tell you a lot, but I will explain.

On the left picture you basically see the optimization performance with 10 particles where we do not use any interaction.

So we initialize 10 particles in different places over our last surface and we let them all converge according to the value of the gradients at their location.

So what this also results to is that they're all converging to different local minima or saddle points that are local minima, sorry.

But because we're not imposing any interactions, they're all converging into different places that are potentially closer to where they started from or whatever.

And on the right, we actually have a picture where we did impose interactions.

So we imposed a sufficiently high interaction strength, forcing all the particles to converge to the same location.

And what is actually interesting about this picture is that the minima that they all converge to, so this thing around zero and zero points five, is also the one which has the property of being the flattest one among all these minima that we have here.

So this gives us a little bit of intuition into the fact that yes, interactions could potentially also help us in finding a model configuration or converging to a certain set of parameters of our model that is potentially more robust to outliers.

Yeah, that's that.

So basically already the final slide, the conclusions.

What we wanna do when we define this kind of interacting agents algorithms is we wanted to solve the problem of having algorithms that are robust, that are computationally efficient and that are trainable in a decentralized manner.

Now this last thing, this decentralized manner, I didn't touch point as much in this presentation, but we also have a bunch of results that show that if we define the interactions in the right way, also in a setting where each particle, node, agent, whatever is optimizing based on their own local data set, so a decentralized optimization, that interactions in the right way can also help you converge a lot faster to the optimum.

So yeah, what we propose, so what is this framework that we say, that we call the interacting particle algorithm consisting of the standard gradient step, an interaction term, and then the standard or relatively centered additional additive Brownian motion term.

And some results that I showed in this talk is that this algorithm can result in more robustness.

This was part of this flat minima story.

Well, decentralized, I don't think I really showed that as much in the sense that you can train it in a decentralized manner, but every particle in my talk was optimizing the same objective function and not different versions of the objective function.

But that is also possible, like I said.

And it can result in exponentially fast convergence.

So converge to the optimum, in some instances exponentially fast in others linearly fast.

So all in all interacting particle optimization algorithms could be a promising direction for further research and for defining this general set of algorithms that satisfies all these three properties that we would like it to satisfy.

And that's it.

- Thank you.

That was a lot of information.

The flow is open for questions.

Maybe I can start with one.

So I don't recall exactly, but in your interaction term, there was a matrix A, which was supposed be double stochastic.

Is this the only requirement on the matrix or how do we choose it.

Yeah, I think to make a lot of the proofs work, I do have to assume it's doubly stochastic.

What other assumptions do we have on A.

I think this is the main one.

But the interesting part of the matrix A is also how do we define it in such a way that we don't require too many particles to interact with each other because obviously interactions also means computational power is spent on those interactions but still converging close, converging fast and that's something that we're looking into now with like the one of the works that we're working on at this point.

So how can we define the straight up between speed of converges and computational power and how can we define the interactions in the most optimal way.

And do you have intuition about, so I suppose it should be sparse, right, in some sense.

Yes, sparse, but and also you would like to still, I mean, the intuition is basically you would want the particles to exchange their values as fast as they could.

So you don't want to have kind of clusters in this graph, because then it would take you longer, more iterations for all the particles to have learned from each other, if we can say like that.

Yeah, there are different ways that you could optimize over this this framework.

Yeah, yeah.

Okay.

And in the, you made a link between sampling problems and optimization.

And so as far as recall in sampling problems, there is this, the big question is rapid mixing, showing that the microchain will visit the whole space.

Do you have anything like this here or is this counted from complexity.

Well, this is a very good point.

And I think it also relates to the point of like the choice of the interaction matrix because, for example, one of the papers that we're looking at for how can we choose this interaction matrix is exactly this, it basically says, how do we optimize the interaction matrix to achieve optimal mixing of the underlying Markov chain.

It's a paper by Boyd, Stephen Boyd, I think.

So it is very, very much related to everything that we're studying.

Yeah.

Okay, we still have time for questions.

Can I ask something.

It's about the mirror descent, I'm not so familiar with that.

But how do you choose your phi function.

Is it very related to your dataset or is it quite general.

I'm wondering how flexible your network actually is once you've chosen such a function.

So I think the question of choosing your function phi is a very complicated one because there are a few settings where people kind of know how to choose your phi.

So for example, if you have this simplex constraint, they kind of say that a certain function phi is always going to work well.

But it's not very easy.

Like if you have a problem, so let me explain it differently why I think this this function phi is interesting is because, I don't know if you're familiar with this concept of preconditioning, but basically preconditioning means that you're pre-multiplying your gradients with some additional matrix.

For example, in the Newton dynamics, this matrix is the Hessian of your objective function.

And if your problem is super ill conditioned by preconditioning with this Hessian, you can converge faster, actually in one step for quadratic problems.

And so the interesting thing with mirror descent is you can exactly rewrite it as preconditioned dynamics, but the preconditioner is gonna be your Hessian of the mirror map.

So this is kind of why there is value in studying different choices of the mirror map, but it hasn't really been done much.

Like I say, there are a few choices of the mirror maps that we know work in a few settings if you also choose the other parameters in the right way.

And that's kind of it.

I did.

So on the one hand, I see potential in it because of these links with preconditioned grading descent.

There are also links with pre-mani and grading descent if you've ever heard of that.

So it seems super valuable, but there isn't really a rigorous way of choosing it in a wide set of problems.

So this is on the one hand why it's interesting to me, but on the other hand, also why it's at this point, relatively useless if you go beyond a certain class of problems.

Is there evidence that the choice of such a function is somewhat general or is it is there evidence for the opposite.

So the choice of the function it really depends on I mean the story behind mirror maps is that if you have a certain objective function and a certain constraint set there might be a perfect mirror map that matches the geometry of your objective function and matches is the geometry of your constraint set.

So this theory is there.

If you have a certain kind of geometry underlying your optimization problem, so both your objective function plus your potential constraint set, then yes, there is evidence that choosing the right mirror map is beneficial.

But as I said, this evidence only exists in, I think, limited cases.

So how to choose it in any real world data set, let's say.

there is no guidance to doing that.

So it might exist, but I don't know.

Yeah, but once you choose it for one problem and you have similar problems, how well is the map for those similar problems, but slightly different.

Because you want to use your network for problems that are different from each other.

That's the more general approach, I would say.

Yeah, so I would say that if you did find a function that works well on a certain geometry of your objective function and potential constraint set, it will work the same or equally well on problems of similar geometry and constraint sets, geometry of objective function and constraint sets.

This is what I can say about that.

So it should work if the the underlying geometry of your problems is the same.

.