Lecture 2 Script

Artificial Intelligence as Statistical Learning

Slide 1

- Before we move on to talk more about graph neural networks we need to be more specific about what we mean by machine learning and artificial intelligence.
- Our specific meaning is that ML and AI are synonyms of statistical and empirical learning. These are the subjects we will study in this lecture

Slide 2

- Conceptually, an Artificial Intelligence is a system that extracts information from observations. This is either a downgrade of the meaning of human intelligence or the meaning of artificial intelligence. Depending on how much you choose to buy into the equivalence of the two.
- In either case, I am sorry to disappoint you.
- If you need an example to clarify the difference between observations and information, consider this system.
- In which this image is an observation
- And the intelligence tells you this s your professor, me.
- The names of this process of mapping observations to information vary across communities. Some people call this AI, some call it machine learning, some pattern recognition. My own professional bias is to talk of signals and signal processing.
- I am also used to call observations inputs and to call information outputs. In any event, the important point is that you are given a something, an observation or an input, out of which you want to extract another something, some information, an output. The AI is the engine that does the mapping.

Slide 3

- We denote the observation inputs by **x** and the observation outputs by **y**. They are related by a statistical model in the form a joint probability distribution p(**x,y**). This a relationship that is given to us by the universe, by nature if you prefer it.

- Given that the universe associates inputs **x** and outputs **y** according to the joint distribution p(**x,y**), the design of the AI is simple.

- It should simply predict outputs according to the conditional distribution p of **y** given **x.**

- Or it could predict outputs according to a conditional expectation if we want a deterministic output. As opposed to the random output of the conditional distribution.

- (Empty)

Slide 4

- An important point to stress is that it is unrealistic to expect the AI to be perfect. In general, we expect that nature and the AI may produce different outputs when they presented with the same input. The situation should look something like this.

- Nature relates x and y according to the distribution p(**x,y**).

- An ideal AI would produce this same relationship but a realizable AI will produce outputs **y hat** according to a mapping Phi of **x.** This mapping Phi is the AI.

- Different possible intelligences will map inputs to outputs according to different maps. In order to evaluate the merit of a particular function Phi, we introduce a loss function *ell* of **y** comma **y hat.** The values off this loss function measure the cost, or the loss, of predicting the output **y hat** when the real output produced by nature is **y**.

- An example loss function is the squared 2-norm which measures the Euclidean distance between **y** and **y hat**. This is common in estimation problems.

- Another example is the hit loss which counts the number of components in which the real y and its prediction disagree. If the output **y** is a scalar, this is just an indicator of whether the prediction is correct or not. The hit loss is common in classification problems.

Slide 5

- Irrespectively of its particular form, the loss *ell* of **y, y hat** is for an individual output and an individual prediction associated with an individual input. But we are interested in a range of inputs and outputs. We therefore consider the average of the loss over the distribution p(**x,y**). We are also interested in the best possible AI. We therefore minimize the loss over the choice of map Phi. We end of with this optimization problem in which we average the loss *ell* of **y** Phi of **x** over the nature's distribution p(**x,y**) and choose the best estimator Phi star. This problem is sufficiently important so as to deserve a second look.

- We begin with the predicted output Phi of x. This is done by the AI.

- Then, nature draws the real output y.

- We proceed to evaluate the loss ell.

- And to take expectation over the joint distribution p(**x,y**).

- The optimal estimator, or optimal classifier, or optimal AI.

- Is the function with the minimum statistical average cost over all possible estimators.

- This optimization program is called a statistical risk minimization problem or SRM problem for short.

Slide 6

- The SRM problem leads to what we call learning, or training, which is simply the process of solving SRM.

- Presented with an input x, the candidate AI Phi produces an estimate y hat equals Phi(x)

- Presented with the same input, nature produces output y

- We evaluate the pointwise loss for this prediction-output pair and we take the average over the distribution p(x, y). We then search for the function Phi star with minimum average loss.

- (Empty)

- The outcome of this learning process is the function Phi star with minimum average statistical loss. It is the best possible classifier according to the criteria that WE specify in the loss function ell.

- Therefore, we can say that we have learned to estimate the outputs y that nature produces. We have learned to predict nature. We have learned to imitate nature.

- Once we have learned the optimal AI Phi star, during execution time we just have to evaluate it. Whenever we are presented with input x, we state that nature intends to produce y as an output.

A Word on Models

Slide 7

- We need to have a talk about models.
- This is because we have seen how to formulate learning as a mathematical program. But this is not a very complete formulation of learning
- In turn, this is because our formulation requires access to models in the form of the probability distribution relating inputs to outputs. And this is easier said than done.

Slide 8

- Indeed, we have rewritten AI as the solution of the SRM problem illustrated in this block diagram.
- But the implementation of this block diagram relies on knowing the probability distribution p(x,y).
- Namely, the implementation of this block diagram requires a model of how x and y are jointly generated.
- If this block diagram does not work without a model, the pertinent question is: Where is this model coming from? There are three possible answers to this question: System's modeling. System's identification. And Machine learning proper

Slide 9

- The first source of a model is to actually build a model. We can do this when we know the laws that relate inputs and outputs. Maybe we know the physics of the problem. Or it's chemistry or it's biology. Maybe we understand how a virus propagates.
- This is not rare. We very often know or can build good models. In this image I illustrate a swarm. We know exactly how these drones respond to motor inputs.

- We are going to talk about learning and we will therefore implicitly disparage models. But do not underestimate the value of models. This is how the vast majority of the technological marvels around you have been designed.

Slide 10

- The second source of a model is system's identification. We do not know the laws governing the system but we acquire data pairs x_q, y_q drawn from nature and we use them to estimate the model.

- You can think of this as learning the probability distribution p(x,y). This is also common. A wireless network is a good example. Your cellphone doesn't know the probability distribution relating the power it transmits to the power received by the base station. But it can measure it. At a more macro level, your service provides can't mode the irradiation of a city. But they can measure it. They show it to their customers on commercials.

- And this is also very powerful too. What we don't design with models built from principles we design with models built from systems identification.

Slide 11

- The third alternative is to bypass the learning of the distribution.

- We go straight to the learning of the estimation map Phi (x). This means that we replace the SRM block in which we use the model to associate outputs to inputs.

- By a Data Samples block. The learning problem now changes from one in which the AI is trying to imitate a model into one in which the AI is trying to imitate observations.

- This is also a very powerful alternative. The technology is still more in the realm of promise than realization. But the promise is real. We have seen impressive transformations in speech processing and computer vision since the mid 2010's.

Empirical Risk Minimization

Slide 12

- We began with a definition of learning in terms of statistical risk minimization. But we have evolved into a definition in terms of what we will see now is empirical risk minimization.
- This is a form of learning that bypasses models
- By trying to imitate observations, as opposed to imitating models.
- Let us formulate this mathematically. Get a pencil. It might get heavy.

Slide 13

- Henceforth, we will use the terms AI and ML to refer to the pipeline illustrated in this block diagram. In this pipeline we learn from data samples.
- Not distributions
- Data Samples
- The AI attempts to imitate input-output pairs we have observed in nature and have incorporated into a training set. Let's look at this mathematically.

Slide 14

- Statistical risk minimization works on a pointwise cost ell of x, Phi of x averaged over the distribution of input-output pairs. It operates on a statistical cost.
- The epistemological value of expectations is that they are well approximated by averages. We can therefore approximate the statistical cost with data.
- To do so we acquire a training set calligraphic T containing capital Q input output pairs (x_q, y_q). These pairs are drawn independently from the distribution p(x,y).

- Given these samples we know that for sufficiently large Q it is possible to approximate the expectation with an average. That is, instead evaluating the statistical risk by taking the expectation highlighted in blue, we evaluate the empirical risk highlighted in red. The empirical risk is an average over the data samples (x_q, y_q).

- That the empirical risk is close to the statistical risk for a given function Phi is just a particular statement of the law of large numbers. We may quibble over the meaning of what it means for the number of samples Q to be large or what it means that the empirical and statistical risk are close. Conceptually, however, the proximity of these two quantities holds under very mild conditions

Slide 15

- One would therefore be very well justified in replacing the statistical risk minimization with an empirical risk minimization problem.

- Thus, instead of learning the optimal statistical classifier Phi star sub S. Which minimizes the statistical average of the pointwise losses

- We learn the optimal empirical classifier Phi star sub E. Which minimizes the empirical average of the pointwise losses

- Given that the objectives are close for any function Phi, which we have just seen follows from the most basic fact of probability theory, One would think that the optima are also close. That is, that Phi star sub S and Phi star sub E are similar if the number of samples Q is sufficiently large.

- One would be thoroughly disappointed. This is not true.

- The statistical minimizer Phi star S and the empirical minimizer Phi star E, need not be close. If you care to know why we have gone wrong, the formal mistake is in exchanging a limit with a minimization. The minimum of the limit of a sequence is not the same as the limit of a sequence of minima.

- The importance of the why notwithstanding, I am more interested, and I want you to be more interested, in how we go wrong. In particular, I want you realize that the solution to the ERM program is trivial.

- You just need to make the optimal AI copy the output y_q for all of the inputs x_q that appear in the training set. As long as you do that, the pointwise losses *ell* of y_q, Phi x_q

vanish for all q and the empirical risk is null. Since the loss function is nonnegative, this is the minimum possible value for the empirical risk.

- This is just as trivial as it is nonsensical.

- It yields no information whatsoever about observations that are outside the training set. Any function in which Phi of x_q copies the corresponding output value y_q is optimal in the empirical risk. The outputs Phi of x for inputs that are not part of the training set can be arbitrary. They do not appear in the cost. We have learned nothing about what goes on outside of the training set.
-

ERM with Learning Parametrizations

Slide 16

- Learning with data produced a mathematical formulation in terms of empirical risk minimization problems. Alas, it produced a problem that does not make sense.

- The search for a problem that makes sense. Brings us to the concept of learning parametrizations.

Slide 17

- To obtain a sensical ERM problem we require the introduction of a function class C. Instead of searching for an optimal AI over the space of all possible functions, we search over the functions that belong to this class.

- For example, we could select the class of linear functions so that the AI is restricted to produce outputs of the form H times x, for a certain matrix H. If we do that, the parametrized ERM problems reduces to a search for the matrix H that solves an ERM problem in which the map Phi of x is of the form H times x

- Now, the choice of a linear parametrization may be good or bad. It very often is a bad choice. But, regardless, it improves on the non-parametrized ERM, because the solution is at least sensical.

- Good or bad, once we have find the optimal estimator H star, we can used to produce estimates y hat for observations x outside of the training set. Just premultiply the input x by the optimal ERM matrix H star.

Slide 18

- Parametrized empirical risk minimization is sensical in a mathematical sense. If we compare parametrized statistical risk minimization with parametrized empirical risk minimization, their solutions are close if we restrict the class C to functions with sufficient smoothness.

- On the left we have SRM restricted to a function class C and on the right we have ERM restricted to the same function class C. The solutions of these two problems are close for C sufficiently smooth and Q sufficiently large.

- This is the fundamental theorem of statistical learning and it states that parametrized ERM is a valid approximation of parametrized SRM.

- I want you to realize that the introduction of the function class C is not an innocent change. It introduces the complication of having to identify a proper class. But I also want you to realize that this problem is unavoidable. We can't have learning without a function class. ERM is otherwise nonsensical.

Slide 19

- We began by formulating learning as an SRM problem where we learn from models. But this is unusual nomenclature.

- In reality, learning is reserved for parametrized ERM problems that learn from data.

- There are three differences between SRM and ERM

- Because the distribution is unknown. The block in the bottom left corner.

- Is replaced with a training set block that signifies our access to a collection of input-pairs x_q y_q contained in the set T

- Because the nonparametric ERM problem is nonsensical. The block in the upper left corner.

- Is replaced with a block where candidate AI functions are restricted to belong to a function class C.

- And because we are operating with data. Not models. The statistical risks minimization block in the right.

- Is replaced with an empirical risk minimization block where the cost is averaged over entries of the training set.

- These is therefore the block diagram of an ERM system. It is made of three components. A training set. A function class. And an empirical risk.

Slide 20

- This has been a little intense. Let's summarize. In this class, we will use machine learning and artificial intelligence interchangeably.
- And both of them will be interpreted as synonyms of empirical risk minimization. This is not entirely accurate. But sufficiently close for our purposes. When in doubt, ERM is the term that I have defined precisely. The other two may mean different things to different people.
- ERM, as shown here, minimizes a loss function averaged over a training set with the AI function restricted to a given class.
- The three components of ERM are
- A dataset T containing pairs that describe an input-output relationship we want to mimic.
- A pointwise loss function ell, that evaluates the fit of predictions Phi of x relative to the actual output y drawn by nature.
- And, most importantly, a function class C which restricts the search space of possible AI maps. Without which, ERM does not make sense.
- To make the parametrization more explicit, consider a p-dimensional parameter H to span the function class. That is, for different values of the parameter H, we instantiate different function Phi belonging to C. We can thus restate the search for an optimal function as a search for the optimal parameter.
- When we look at this formulation of ERM, we realize that the design of an ML or AI system is tantamount to the selection of the function class C.
- For once, what else is in this problem that is left as a choice for the system's designer? The data set is to be acquired, the loss is jus a metric.
- It is the function class that determines the properties of the AI. Because it is the function class that determines how the AI generalizes from the inputs x_q in the training set to inputs x that are not part of the training set.

Stochastic Gradient Descent

Slide 21

- ERM entails solution of an optimization problem. Stochastic gradient descent is the customary method used for the minimization of the empirical risk.

Slide 22

- The minimization problem associated with the training of an estimator is shown here in its parametric form where the range of possible functions Phi is spanned by a parameter H. Our goal is to find the optimal parameter H star.

- In our discussions here it is convenient to define the average loss function capital L of H as the average of the pointwise loss functions.

- With this definition, the training problem is just the minimization of the average loss function L of H.

- The function we want to minimize is an empirical risk. But it's particular form is somewhat besides the point. It is just a minimization. And minimizations can be carried out with the gradient descent algorithm.

Slide 23

- Gradient descent, not surprisingly, uses gradients. The property that gradients have that makes them useful in finding a minimum is that they are perpendicular to the level sets of the loss. Please notice the introduction of notation here. We use g go H to denotes gradients of the loss L evaluated at parameter H.

- In this illustration we illustrate some level sets of a loss function

- Along with a gradient g of H which is, as we have said perpendicular to the level set. The gradient is the rate of change and points outwards of the level set. We depict the negative gradient which points inwards.

- As a consequence of pointing inwards of the level set. The negative gradient points towards the minimum argument H star.

- Not directly, by the way. Towards. Kind of a rough driving direction

- Mathematically, the angle between the negative gradient and the arrow H - H star that points to the minimum, is less than pi over 2. This is because the inner product of these two vectors, the negative gradient and H - H star is positive.

- If the gradient points towards the minimum we can use it in a gradient descent algorithm. We have an iteration index t and an associated parameter value H_t. We evaluate the gradient g of H_t associated with this parameter and scale it with a stepsize epsilon. We then update the parameter value to H_{t + 1} by subtracting the scaled gradient from the current parameter value H_t. Since the negative gradient points towards the minimum, H_{t + 1} is closer to H star than H_t was if the stepsize is sufficiently small.

- Formalizing these observations we can prove that iterates H_t convergence to the optimum parameter H star.

Slide 24

- Up until now we have been talking about L of H without considering its specific form. If we look at the particular form of the empirical cost, we see that the gradient of the average loss is an average of the gradient of the pointwise losses

- Equipped with this specific gradient form, we can rewrite gradient descent for empirical risk as an iteration in which the parameter iterate H_i is updated by subtracting the average of the gradients of the pointwise losses.

- This is all good, but those gradients have turned out very costly to compute. They are an average of Q pointwise gradients. Even if we have a small number of data samples, this can take a lot of computation to evaluate.

Slide 25

- To avoid this cost, we use stochastic gradient descent. At iteration t, we select a batch of Q_t samples from the training set. This batch, which we will denote as T sub t is drawn randomly from the data set and is such that the number of samples it contains is much smaller than the number of samples Q in the training set.

- We now define the stochastic gradient hat g as an average of pointwise gradients over the batch set. This is misleadingly similar to the regular gradient. The only difference is

the set over which we sum pointwise gradients. But the sets over which we sum are very different. The gradient set has all of the Q samples. The stochastic gradient set has a batch of Q_t samples which are much smaller in number and chosen at random from the training set.

- Setting aside these differences for a second, we can think of replacing stochastic gradients for the gradients used in gradient descent. Doing that, yield the stochastic gradient descent algorithm. Updates of parameters follow the stochastic gradient hat g. Or, equivalently, they follow the average of the pointwise gradients over the batch set.

- SGD is cheaper to implement because the average is over a smaller number of pointwise gradients. And it is not difficult to see that it will retain the descent property of gradients. In some form.

Slide 26

- The reason why stochastic gradients keep the descent property, in a sense, is that the expected value of the stochastic gradient, with respect to the random choice of batches, is a gradient.

- Here we have again our illustrative function with its level sets and its minimum argument H star.

- We know that gradients point inwards of the level set and therefore in the rough direction of the minimum.

- Stochastic gradients are random. Depending on the batch draw, it may be that the point inwards or it maybe that they point outwards. But more often than not the stochastic gradients point in the right direction. This has to be the case for their expectation to be a gradient. We signify this on the figure with the heat map of stochastic gradients having more mass around the gradient than in the opposite direction.

- Now, if stochastic gradients point in the right direction on average.

- It means that we move towards the optimum more often that not.

- If you want to be more formal about it, the expected angle between the stochastic gradient and the arrow with a head at the optimum is acute on expectation.

- And if you wanted to be completely formal, you would use the angle equation to build a submartingale and prove convergence. This is not difficult. But since this is not a class on optimization, I will not bother you with the details.

Stochastic Gradient Descent Memorabilia

Slide 27

- Our coverage of SGD has been brief and, therefore, incomplete. I covered only because there are a few things I wanted you to understand. And a few things I wanted you to know.

Slide 28

- I wanted you to understand why gradient descent converges. Which is because the gradient points towards the minimum.

- And also why SGD converges. Which is because the stochastic gradient points in the right direction on average. These two facts are true for any convex function.

- I also wanted you to know that for the empirical risk, the cost of computing stochastic gradients is much smaller than the cost of computing gradients. Because we take an average of pointwise gradients over a batch set that contains a number of samples that is much smaller than the number of samples in the whole training set.

- This is the difference between night and day. Between a method that works and one that does not.

Slide 29

- Besides these fundamental points and, despite our lack of proper coverage, there are some other remarks I want to make.

- When we speak of convergence of SGD, we speak of a peculiar form of convergence. As the iteration index grows, the limit infimum of the distance between H_t and H star is smaller than a quantity whose order is epsilon divided by the square root of the batch size Q_t.

- Therefore, we do not converge exactly. We have a limit infimum. Not a limit. This means that iterates H_t approach the optimal argument H star and hover around it.

- The size of the hover region is proportional to the stepsize epsilon. To get closer to the optimum we can decrease the stepsize. This comes at the cost of slower convergence. More iterations are needed to approach the limit.

- The size of of the hover region is also inversely proportional to the square root of the batch size. We can get close to the optimum by increasing the batch size. This triggers larger computational cost per stochastic gradient. More time is needed in each iteration.

- Since I am talking about convergence, I also want to clear a potential misconception. For large batch size Q_t, the stochastic gradient hat g and the regular gradient g are close.

- This is true. But this fact is not needed for convergence. In fact, SGD converges even if the batch size is just one. That is, even if we select individual samples from the training set at each step.

- Convergence follows because the stochastic gradient points in the right direction on average. And if a random mistake is made at a certain iteration and we move away from the optimum, this mistake is more likely than not to be corrected in the next step. There is a certain relentlessness to descent methods. We are, almost literally, pushing a ball downhill. The difficult task is to push the ball uphill. There is a lot we can do a descent method that will not preclude its convergence. The convergence time and the hover region, of course, may be affected.

Slide 30

- Another fact that I want to stress is that I have grossly understated the complexity of functions that are not convex

- All of the plots I have shown illustrate convex functions. All of the comments I have made apply to convex functions. And all of the results I have explained hold for convex functions. As highlighted by this illustrative set of level sets.

- In reality, many problems involve functions that are not convex. Germane to us, this is the case of neural networks, including their convolutional versions like CNNs and GNNs. In functions that are not convex the level sets look like the ones illustrated in this sketch. And in addition to a global minimum like H star, we also have local minima like H dagger.

- In this non-convex functions, a gradient may move the iterate towards.

- The global minimum H star

- Or a local minimum like H dagger, of which there can be several

- This will depend on the initial condition. Thus, depending on this arbitrary choice we may end up at the optimum H star, as we want. Or a local minima like H dagger. Which is not what we want.

- This said, it may be that when we implement our SGD algorithms for neural networks we end up at a local minimum, but we won't care.

- We will pull the grand ostrich. Bury our heads in the sand and implicitly assume that H dagger is optimal. This has been observed to be good enough for neural networks and there is a lot of frenetic research activities trying to show that local minima in ERM problems with neural networks are as good as, or at least not far from, the global minimum.

Slide 31

- Finally, I would be remiss if I didn't point out that stochastic gradient descent is not a great algorithm. Just the one we have.

- Among other undesirable properties, convergence speed and convergence itself are very sensitive to the choice of parameters

- Getting SGD to converge and getting it to do so at a reasonable speed requires trying different step sizes, and different batch sizes. May be we even have to try different initial conditions.

- And it could be, indeed, it will often be, that small changes to any of these parameters have a large effect on convergence. In a word SGD is a finicky algorithm. This is one of the reasons we will use packages to implement SGD. To use pre-developed software that reduces the fastidious task of trying parameters.

The Importance of Learning Parametrizations

Slide 32

- We will close this lecture with a discussion on the importance of selecting the right learning parametrization.
- We have seen that artificial intelligence reduces to empirical risk minimization and that in ERM all we have to do is choose a learning parametrization
- We will illustrate with some examples that this is not an easy choice. The parametrization controls generalization outside of the training set and it can make or break an AI system
- When all is said and done, the parametrization is a model of how outputs are related to inputs. And, as is always the case of models, they have to be an accurate representation of nature.

Slide 33

- In reality, data is gathered from nature. To illustrate the effect of learning parametrizations in AI, we will generate our own fake data that follows models that we specify.
- One of them is a linear model in which inputs x are related to outputs y according to a linear transformation.
- Determined by a known matrix A of proper dimensions.
- We also add a white Gaussian noise term that is chosen independently of the input x and has zero mean.
- The other is a nonlinear model where we post process this same linear transformation with a sign function.

Slide 34

- Given that we know the models we can compute the statistical risk minimizer. As in our first attempt to build an AI

- For instance, if we use the squared 2-norm loss to measure the pointwise mismatch between AI estimates and actual outputs
- We end our with the SRM problem in which we average the pointwise loss over the data distribution.
- Using the given linear model and taking derivatives it is ready to determine the optimal AI function
- Which is nothing else but the very same linear transformation that appears in the model
- This is a trivial example, but I am using it to illustrate the point that the AI mimics nature. Literally in this case. The optimal SRM estimator is the model itself.

Slide 35

- But in they course we are more interested in cases where the model is unknown. Thus, even though we know the models of our fake data, assume that we don't know them.
- Instead, we have access to Q data pairs x_q comma y_q which we lump in a training set T.
- We hypothesize a linear parametrization Phi of x equals H times x. This is not the model, which we denoted by A times x. Just a hypothesis on how outputs are related to inputs. An assumption on what is a good function class for this machine learning problem.
- Using this function class we end with the ERM problem
- In which the pointwise loss is averaged over the training set. Not over the distribution. Which we are assuming, as would be the case in applications, that it is unknown.
- This is a problem we can solve with stochastic gradient descent.
- The particular form of which we show here for reference. But this specific expression for SGD is not important for our forthcoming discussions
- A more important observation is that the linear parametrization we have chosen can be used irrespectively of what the actual model is. It does not matter how inputs are related to outputs in the actual set. We are free to hypothesize a linear relationship wherever we please.

- But f course. This learning parametrization will work well only when it matches the unknot model. This is the most important point we have to make about ERM. The importance of matching the parametrization to the unknown model. We will therefore elaborate with three examples.

Slide 36

- Consider data generated with a linear model. We are using dimensionality 100 for inputs and outputs and we operate with 1,000 entries in the training set.

- Our ERM problem uses a linear learning parametrization

- As shown on the plot on the left of your screen, the iterates of the SGD trajectory reduce the loss. We are therefore succeeding at solving the ERM problem. And the loss to which we converge is small. Thus, we are learning an AI that approximates the model well within the training set.

- But we must recall that during live operation we operate outside the training set.

- The plot on the right shows testing of the SGD trajectory on a different set. We see that the loss is also reduced. Thus, we not only succeed in solving the ERM problem and reducing the loss to a small value. We also succeed at learning to operate outside the training set.

- There is no mystery here. The model is linear. The parametrization is linear. Thus, the parametrization learns the model

Slide 37

- As an example of a different nature, consider the sign model with the same dimensions and the same number of samples.

- We are still using ERM with a linear parametrization as before.

- And when we look at the SGD trajectory we see that loss is again reduces. We are succeeding at solving the ERM problem. This is just a property of SGD. SGD is working. As we know it should.

- However, if we look at the value of the loss to which we are converging, we see that the loss is high. We are converging to an AI that, however optimal, is not that good. We are solving ERM but we are NOT learning.

- When we look at operation outside the training set the situation is just as bad. There is no reason to expect successful testing out of failed training. Miracles do not happen.

- There is not mystery here either. The model is NOT linear. The parametrization is linear. The parametrization does not learn the model.

Slide 38

- Out third example goes back to the linear model. We retain the dimensions of the input and output data but reduces the number of samples to Q equals one hundred.

- We still learn with a linear parametrization

- We see that SGD reduces the loss, because SGD succeeds at solving ERM. And, since the model and the parametrization are matched, the loss converges to a small value. We are learning to predict outputs within the training set.

- But when we operate outside of the training set during live operation.

- The loss is not reduced by much. We are failing to learn to operate outside the training set.

- In this case there is a little bit of a mystery. The model is linear and the parametrization is linear. So how come we are not learning the model? The answer to this mystery is the complexity of the model relative to the amount of data that is available. We know that if we increase the amount of data in the training set we will succeed in reducing the loss outside the training set. We saw this in our first example.

- But when we reduced the number of samples in the training set we ended up with insufficient data. There is not enough data in the training set to learn the model.

- The very important observation to make here is that there is never enough data. Thus, our models not only have to be matched. The have to be sufficiently simple that they can learn with amounts of data that, however large, are always insufficient.

Slide 39

- The point for us to remember as we move forward is that machine learning is model free but not really model free after all.

- It is true that ML does not require a model relating inputs x to outputs y. This was the whole motivation for introducing ML.

- In the examples we considered here we do not need to know the matrix **A**

- But we do need to know a class of functions to which the model belongs. At the very least some rough hypothesis of which parametrizations are adequate for the problem of interest.

- In the examples here we succeeded at learning when our hypothesis that the model is linear matched the reality of the input-output model. We failed when model and parametrization were mismatched.

- But we not only need our parametrization to match models. We need them to be of sufficiently limited complexity so that they can operate with insufficient data. Data is always insufficient when the problem dimensions are large.

- This is where we leverage structure using convolutional architectures such as CNNs and GNNs. We will start doing so in the next lecture.

Time Estimates for Recordings

- It takes about 2.4 minutes per page.
- It takes about 3.8 minutes every 2 pages
- A script that covers about 7 minutes takes about 3 pages.