INTRO TO DL (speaker: Razvan Pascanu)
EEML 2020

Can you please point to some documents that discuss these mathematical fundamentals of DNN

Razvan: The slides have links to some papers. I've been somewhat biased when citing, and these are older papers -- there are very valuable follow-ups that I'll leave it to you to find. I'm also not sure of which aspect you are referring to, but I've put a few things here:

- **Linear region: https://arxiv.org/abs/1402.1869**
- **Mode connect: https://papers.nips.cc/paper/8095-loss-surfaces-mode-connectivity-and-fast-ensembling-of-dnns.pdf**
- **Well behaved loss surface: http://arxiv.org/abs/1406.2572 https://arxiv.org/abs/1412.0233**
- **Bad local minimas:https://arxiv.org/abs/1611.06310**
- **Double descent: https://arxiv.org/abs/1812.11118**
- **Lottery Ticket Hypothesis: https://arxiv.org/abs/1803.03635**
- **NTK: https://arxiv.org/abs/1806.07572**
- **Loss of generalization: https://arxiv.org/abs/1910.08475**

Jaspreet Singh

15 today, 10:06am

Are there any set of rules which we should take care of while designing CNN architecture from scratch instead of designing randomly??

Razvan: I would strongly encourage to start from an existing architecture and alter it if you have a decent reason for it. Deeper models tend to learn better solutions, but are harder to train. Increasing width makes learning easier. Skip connections help. Multiplicative/gating seems to be a usually neglected tool in the toolbox (e.g. https://openreview.net/forum?id=rylnK6VtDH, https://arxiv.org/pdf/1910.06764.pdf, https://arxiv.org/pdf/2002.10444.pdf).

Ciprian

14 today, 10:18am

Hello, Răzvan. Do you have any intuition about what type of Nets we can combine in a Multitask Learning problem?

Razvan: Not sure I understand the question, but any architecture can be applied in multitask. I think in multitask learning you want to deal with interference, and maybe unbalanceness (e.g. if a branch has lots of parameters and other doesn't). This unbalance can be fixed by strengthening the signal with some local losses (think of self-supervision or https://arxiv.org/pdf/1611.05397.pdf, https://arxiv.org/abs/1611.03673). For interference, maybe something like a mixture of experts ? Gating and multiplicative interactions seem to help (https://openreview.net/forum?id=rylnK6VtDH , https://arxiv.org/pdf/1910.06764.pdf)

Burak Ercan

14 today, 10:43am

What do you think of transformer based models taking over RNNs for sequence tasks? Can you compare them, and comment where RNNs are still the preferred method?

Razvan: To repeat my answer and expand it a bit. (1) I think transformers will struggle with ordering. E.g. the same events are present in all data points, but there is a simple one position switch or small ordering changes between the sequences. That is because this relies on the positional encoding, and I don't think we know how to encode position. One-hot e.g. doesn't give you the right topology. If you pick a smooth function like cosine and represent position as values on this function, then you might have a resolution problem. I think RNN can deal with this better.

(2) RNN are cheaper at inference! So that is great when you want to deploy big models, and something one shouldn't overlook.

(3) Transformers tend to be tricky to train sometimes (e.g.https://arxiv.org/pdf/1910.06764.pdf) and we have more experience in figuring out how to stabilize learning for RNNs.

That said, if you want to do translation models or many other things, there is no doubt transformers will outperform RNNs. And they are just an amazing architecture that will show us a lot more in the coming years

Martin

11 today, 10:02am

Is somehow the sharpness / flatness of the minimum related to the uncertainty of NNs predictions?

Razvan: More contextual answer. What uncertainty? Are we talking about prediction uncertainty (uncertainty in the outputs) ? Then the connection can be made through the softmax argument. Being certain you need to saturate the softmax, so you might need larger weights, hence larger curvature. But there is not a 1:1 connection there .. and you can saturate the softmax by just

increasing the weights of the top layer, or distributing this among all layers. This will lead to very different outcomes.

Are we talking about the posterior over the weights (bayesian DL perspective). Then yes the connection is even stronger. If you take a Laplace approximation of your posterior, that will give you the Fisher (which is a proxy of the Hessian) whose eigenvalues we are talking about. One would need to discuss the relationship between the Fisher and Hessian which I think there is not enough work on. But it still doesn't change the worry that I have as in, you still don't have a number to call flatness and to compare different solutions against each other .. but maybe I misunderstood the question.

Younduk Nam

10 today, 10:09am

Almost all CNN use pooling layers. However, the alpha go paper does not use pooling layers. Is there reason for not using pooling layers?

Pooling is not as popular generally. Resnets don't have them, and most newer architectures don't either. They are throwing information in an *unlearned* way so there is no doubt that is better to learn how to throw information out. The reason they were a must in the past was for computational reasons. Convnets also tend to use a lot of memory .. pooling reduced that too.

Saby

8 today, 9:57am

Why do we prefer to regularize a complex/expressive model instead of directly training a less expressive model?

Because it gives us a lot more slack in terms of finding the exact right family of expressive models. Plus we don't need to handengineer a new family of models of a given expresivity for each task. We take the super expressive family .. same for everything and just regularize it more strongly or less strongly.

Mikhail Belkin will talk about double descent which is some really great work he did. But basically another reason is that in over-parametrized regimes, the usual overfitting behaviour doesn't seem to occur. There is still a lot to be studied here and a lot of open questions, but empirically we see this. And it has something to do with implicit regularization of our tools. So basically we don't need to do anything if we have a very over-parametrized model, which makes everything easy!

Ciprian

8 today, 10:02am

Hello, Răzvan. What is happening if we use 2nd order or 3rd order Taylor Series to express the loss?

Razvan: You mean in deriving SGD? You get a different interactive process for learning. For second order Taylor you get the Newton step where the hessian is damped (this comes from the trust region that you still need).
In my PhD thesis (sorry -- wasn't cool enough to publish anywhere) I did some of these derivations combining second order methods with natural gradient.

You get meaningful powerful gradient-based iterative optimizers. But quite expensive ones. Third order taylor .. is probably intractable to compute for a neural net. The hessian is intractable too so you need to approximate it. You can find different ways of approximating it though.

Anonymous

6 today, 10:28am

Could you elaborate on the intuition behind the curvature (flat + wall) of the loss for RNNs?

Razvan: Yes, sorry. I wanted to cram a lot of material because there are so many cool things to talk about but it probably led to a less enjoyable talk.

The thing that I wanted to highlight is that for RNNs this curvature landscape is a bit less symmetric. I.e. there is one direction that you want to move in, and if you do the curvature explodes. If you move in the other direction (this will slightly lower the eigenvalues of Jacobian) curvature will not change by much. It reflects the lack of symmetry of having the gradients explode only when the eigenvalues go over 1, but when they decrease, the norm of the gradients doesn't change by much. In the paper (https://arxiv.org/abs/1211.5063) I had this discussion about this direction in which things explode potentially hiding other directions in which you can make meaningful progress. Not sure if this answers your question. The slides are almost verbatim taken from that old paper.

Burak Ercan

5 today, 9:57am

We do not find a global optimum with optimisation, which seems bad at first; but it seems to help with generalisation. Is this point of view correct?

Razvan: A few notes:
(1) You don't want the generalization to rely on you not being able to converge -- or converge to the right kind of local minima which we don't even know what it means. That makes the whole thing ill-defined. Ideally you want to have the best optimizer that works on the best regularised loss. So fix regularization, and then employ the strongest optimizer. But of course that is an idealistic view and maybe we can't fix the regularization properly or at least we need to understand better the implicit form of regularizations we have from the optimizer.

(2) There is no theory saying that global minima generalize worse. There is some intuition that in the vicinity of the minima (be it local or global) you might generalise better because you overfit less. But that intuition doesn't say anything about the type of minima. More likely the reverse is true. If the local minima is poor in terms of performance you have a guarantee that generalization will be worse. Test error is bounded from below by the train error. So if the train error is large, you test error can only be larger than that.

Anonymous

5 today, 10:00am

Hi, what do you mean by "Strict the size of the Model"?

Razvan: Sorry -- I might need some context here. Not sure what I was referring to. Maybe you can give more details. Maybe I meant "restrict the size of the model" to control capacity and not overfit?

Novin Shahroudi

5 today, 10:08am

Is what you explained about changing the curvature would have the same effect as having different learning rates per layer or per neuron? If not, what's the diff?

Razvan: Yes and no :). Yes, correcting by the curvature is equivalent to having a per-parameter learning rate. But curvature is a property of the loss function. Is the second

derivative of the loss function. So curvature tells you what are those optimal learning rates (under a second order Taylor expansion) And curvature is a matrix that is NxN so is not an independent learning rate per parameter, but it actually looks at the correlation between gradients entries. But I've been using the word very loosely in my presentation, and I think we need context. But by curvature I tend to mean the Hessian, which says how quickly the gradient change but also how different entries of the gradients are correlated with eachother.

Younduk Nam

5 today, 10:23am

Is it normal to use Batch-Norm in RNN to fight off the exploding gradients?

Razvan: No. One early excuse was the lack of a batch (which was true in many RNN applications). I think LayerNorm (https://arxiv.org/abs/1607.06450) was proposed to be the BatchNorm for RNN. There are a new papers that I haven't yet internalised myself, but that are using normalisation to deal with exploding gradients (e.g. https://arxiv.org/abs/2006.12169). Directly talking about batchnorm, there is also this question of how do you deal with time. Do you keep seperate statistics for each timestep? What if you have different length sequences? Do you average them? There is a nice work from Yann Olivier discussing this: http://www.yann-ollivier.org/rech/publs/pcnn.pdf

Ashok Kumar Pant

4 today, 10:04am

Is there a way besides grid search to add neurons and layers automatically for a given dataset? (edited)

Razvan: Not sure if I parsed the question correctly. Is the question can you modify the model at train time or augment it after training. The answer is yes and there are some interesting works in there (ask me for citations ..).
Or are you saying is there a way of figuring out the right model size for a dataset without training on it? There is no machinery that looks at data and says what is the right model for you. But following the dogma of double descent (see question much higher above), the safe answer is the bigger the better. Pick as large of a model as you can (of course going too far means a lot of wasted compute) and do early stopping if you are really worried or add a bit of regularization.

Also experience helps a lot here. If you start playing with neural nets, and at some point it will be just natural for you based on previous dealing with the domain and what not. Also look at what models other works looking at that domain are using.

Sara

4 today, 10:05am

do you recommend calculating loss through cross validation (kfold)? or train-test split?

Razvan: I think the answer is that k-fold is better, but is not tractable (or at least too expensive given the boost that it gives), so almost everyone uses train-test split. I'm thinking about ImageNet or larger datasets. But even for Cifar 10 with a resnet 50 it can become expensive to do k-fold.
The other note is that sometimes people are a bit sloppy with evaluation, and there are a few datasets that for sure we have been overfitting the test set. The non-disputable case is MNIST.

Jaspreet Singh

4 today, 10:11am

How can we make CNNs equivariance or invariance for problems such as image classification??

Razvan: Would be good to extend the question. There is interesting work on equivariance coming from Max Welling (e.g. with Taco : https://arxiv.org/abs/1602.07576) , but many other cool places. It depends on invariant or equivariant with respect to what property. Depending on that property you have different answers. Some involve moving in Fourier domains. Happy to take this offline as it can become very specific (and I might not know the correct answer either).

Saby

4 today, 10:36am

It seems that Densenet should be the next step after Resnets but somehow Resnets are more popular! Any reason why?

Razvan: I think these things take time. To be honest I still like to think of an MLP or 3 layer conv, LeNet5 style when I want to study things. It might be a question of momentum. Also is not clear

to me Densenet out do Resnet by a sufficiently large margin. But give it a bit more time. IMHO, what I would be most excited about is for us to look into simplifying the models. I think one reason I'm not a vision person is because when it comes to vision you have these very complicated architectures (IMHO) that are there to squeeze most of the performance out. But to me, I'm happy to pay 2-3% performance to have a model I understand better and doesn't include things like batchnorm, which is not really kosher from an optimization perspective. If you look carefully at batchnorm it breaks the proof of convergence for SGD, and in some sense you take each SGD step on a different function (every time you recompute stats, you change the function). That is why I always liked Natural Nets (https://arxiv.org/abs/1507.00210) and I would've wished we had a diagonal version of natural net instead of batchnorm. But that is not what happened -- not that natural net is not well cited, is just not widely used :).

Younduk Nam

4 today, 10:42am

You say gradient is not memorizing, but for overfitted model, it seems like memorizing. Can we understand as weights are memorized and gradient is not?

Razvan: we're talking about different things. What I said is that for an RNN, the fact that gradients flow from h_t to h_{t-k} does **not mean that you remember or memorized or you can recover h_{t-k} which is the problem that LSTM and orthogonal RNN are trying to solve to increase their memory capacity. Also in DL we usually make a distinction between memorizing in the weights vs memorizing in the hidden state (of for e.g. an RNN). I know in neuroscience there is no such distinction. I'm also not saying that having gradients flow means you haven't memorized. I'm saying having gradient flow is not sufficient. Is not even necessary to be honest. I can give you a simple RNN that memorizes the first observation but no gradient flow.**

So the memorization and weights is a different story. Maybe you can ping me if you want to discuss this further .. or leave a comment on the doc.

Burak Ercan

4 today, 10:49am

Hi Razvan. Can you compare your Relation Networks with Graph Networks? Is RN a special case of GN? When to use which?

Razvan: Yes. In my view GNs are a super class, everything else is a particular choice. And when I say GN I don't mean a specific model, so if you say use a GN I would need to make a choice (e.g. is it a GraphConvNet, is it a GraphAttentionNet, is it an Interaction net, is it Yujia's original GGNNs if I got that name correctly, etc.). There is not a lot of benchmarking done between different variants. But sometimes it is obvious which ones are best. I think Petar is a good person to talk

about this. But e.g. in interaction net the message function is more powerful (and expensive) than GraphConvNets. So if messages are complex, and you don't want to have many layers of GCNs, then interaction nets are better. MPNN (message passing NN, which are very similar to Interaction nets) are, I think, known to be better for program execution. If you work with a large number of nodes, you probably want GCN or Graph Attention Net for scalability. By the way, for me, transformers are graph nets. They follow the formalism perfectly.

Anonymous

3 today, 9:48am

Can I assume that for supervise learning the order of data is presented matter? If True is there any way to fix the network instead of reordering the data?

Razvan: The order of the data has an impact. We sample I.I.D data which minimizes this impact. The ordering matters because of this tug-of-war dynamics -- which I don't know if I manage to convey the message, but I think it is a very powerful way of understanding learning. Is there a way of solving this? There are people working on this :). I would argue that continual learning is about this. It is about finding an alternative way of doing credit assignment, by removing this requirement of being IID. And this can be solved at a macro level or micro level. Macro level is more like modular nets and stuff like that. If you really want a citation, look up progressive nets, packnets, and a mixture of experts. For microlevel I think Elastic Weight Consolidation can be repurposed to do this. A paper that I really like (though is just on arxiv) is this Selfie-boost paper that does that: https://arxiv.org/abs/1411.3436

Amin Honarmandi SHandiz

3 today, 10:08am

For GAN network, how can we understand that if generator fool discriminator?

Razvan: One way of thinking of GAN is as a multi-agent game. You have two players, a generator and a discriminator. And the game is for the discriminator to figure out if an image is coming from the generator or not. And the generator is trying to fool the discriminator. This whole thing is differentiable and you can backprop through. Not sure if this answers your question. Leave me a comment if not.

Anonymous

3 today, 10:08am

In ReLU if the x < 0, then the activations are set to zero. We practically ignore the data. Why does it not hurt performance?

Razvan: We're not ignoring it .. or at least I wouldn't use that word. You can think of ReLU in different ways. If you have different ReLU off for a particular region of the space, then the on-ReLUs decide what is the linear response. So you turn a unit off to change the linear response of the model, and the way you get interesting functions is by using different units in different part of the space. You lose a feature of the data point if any non-zero projection of it goes to a unit which is off -- which is extremely unlikely in the models that we use. The other way to think is: if you consider each unit as a particular feature detector. You consider 0 as the feature was detected. Anything negative means the feature is not there, so that is information in itself, and all features that are not there make sense to be 0.

Lastly .. the big worry of ReLU was that you will have many units that are off, and you kill off gradients. E.g. in an extreme case you turn all units off so then you can learn since no info about the input goes forward. For the typical initialization we use (He or Xavier) and for the model sizes we use (more than 50 or 100 hidden units) the likelihood of that is 0. And this is what we realised empirically. This is like the worry with local minima. But if you change initialization (or data -- which normally is normalized in some meaningful interval like 0-1 or -1 to 1) you can break learning: https://arxiv.org/abs/1611.06310

Alex

3 today, 10:15am

Is batch normalisation really necessary to train ResNet's?

Razvan: Depends on what you mean by train. To get SOTA performance. I think at the moment yes, though I think we can fix that. Getting good performance. I guess no. As I mentioned, the paper from Soham De and Sam Smith looks at this https://arxiv.org/pdf/2002.10444.pdf , but there are other works as well. I think give it a year and we will have papers explaining how to get SOTA without batchnorm :)

Younduk Nam

3 today, 10:16am

How many ResNet Blocks requires batch-norm? Is there any other good alternative to batch-norm that you know?

Razvan: Look up Soham De and Sam Smith looks at this https://arxiv.org/pdf/2002.10444.pdf , though again there are other works. Is just I don't want to look for citations :)). I think you have a batchnorm layer per block, but I might be wrong. The placing of batchnorm is also important, though this is probably well understood now.

Olivia

3 today, 10:16am

How to assure the training is not learning to discriminate based on some specific feature that might still be correlated to multiple other inputs?

Razvan: Great question. I guess you are hinting towards correlation vs causality. Neural nets learn correlations .. that is what SGD does. How to fix it? (1) Pre-process the data to remove spurious features you don't want to latch on. (2) Add inductive biases, or domain knowledge in defining the architecture, or altering SGD such that it penalizes it if it latches on a particular feature, or on any feature too strongly. (3) **More data, that will show the model that the correlation discovered doesn't hold always and leads to worse performance.**

Dealing with biases in models and causality are cool topics that I always wanted to get into, I just need to somehow find more time :) -- which feels a hard thing to do for me at the moment.

caius.debucean

3 today, 10:19am

How do you know what kernel size is suitable for a layer? And what is the effect of smaller/bigger kernels?

Razvan: A good question. I would ask this to vision people as well (e.g. Viorica/Carl). I think the intuition was at some point that smaller kernels are better (they just lead to more memory consumption and more compute). I guess at the end of the day it depends on the data. What kind of structure you need to discover. If you have a small kernel size you need deeper models to be able to look in higher layers and a large chunk of the input. So there is a tradeoff between computational cost, learnability (deeper models can be harder to optimize), and what the data really needs. I don't have a formula though. Either intuition (3x3 seems a safe bet always). Or hyper-tuning.

Novin Shahroudi

3 today, 10:25am

Can you state what are the x and y axes on the dynamical system view of the exploding gradients slide (the bottom-left plot)? not able to interpret the plot.

Razvan:

Sorry .. those all were meant to be illustrative. Y is the output of the sigmoid unit. X is the value of the bias. The weight w is fixed. All details are here: https://arxiv.org/abs/1211.5063

Anonymous

3 today, 10:41am

You mentioned that a solution to exploding and vanishing gradients is gradient clipping. What is gradient clipping and how to perform it?

Razvan: Sorry .. I wished I would have had the time to present things better. Or remove some slides. It was too much to cramp and little time to prepare. Gradient clipping only resolves exploding gradient, not gradient vanishing. It is widely used everywhere, from old LSTM papers to any new work that you are aware of. Gradient clipping says predefine a threshold T. Every time the gradient norm is larger than T, normalize the gradients to have the norm T.

There are many ways to do gradient clipping. They all work equally well. You can clip the norm of the gradient as described above. To me is the more mathematically sound approach. You can clip entries of the gradients if their magnitude is too large. You can clip the magnitude of the gradient as you backprop it through the model. You can replace gradients that have the norm too large by small random noise :). They all work. The threshold is not very hard to tune either, in many frameworks it is a constant that everyone uses. It is a very robust solution. I don't find it super elegant. A more in depth discussion here: https://arxiv.org/abs/1211.5063 or ping me for more details.

Anshul

2 today, 10:00am

Is the piecewise linear nature of DL function approximation related to why optimization of first order expansion of loss works?

Razvan: No. Or at least no if I understood your question correctly. You take 1st order Taylor if you want to derive SGD (which is a 1st order gradient based method). Using a higher order Taylor expansion means you can have a larger trust region because you have a better approximation of the function. So you can take larger steps (e.g. Newton method). If anything piecewise linear makes things a bit less nice as gradient optimization methods like things smooth. Piecewise linear functions are not smooth. However the loss is an average over many examples, so that smooths out the function further.

Riccardo De Feo

2 today, 10:17am

Could you please share again the physics/statistical mechanics references hinted at during the first half of the lecture?

Razvan: Yes. Bray and Dean 2007 https://arxiv.org/abs/cond-mat/0611023 , Fydorov and Williams 2007 https://arxiv.org/abs/cond-mat/0702601

Note that these results don't apply directly to neural nets. Our work https://arxiv.org/pdf/1406.2572.pdf just asks whether empirically we observe similar behaviour as predicted to physical systems because at a high level you can make some parallels. https://arxiv.org/abs/1412.0233 tries to make this connection more concretely, but needs to make rather strong assumptions.

2 today, 10:38am

In NN regression, do we always have to normalise our data before training, even if that affect the loss badly? Which normalisation technique do you recommend?

Razvan: Input normalization is useful. Most initialization schemes (including He and Xavier) work under some assumptions about the data distribution. Not normalizing it (particularly if you don't have other tricks upstream like batchnorm) will hurt learning. You can think of it differently. When you initialise the network you shatter the space into many small regions (i.e. the initial piecewise linear function). There are going to be some regularities of this shattering (i.e. the regions will be centered around zero becoming larger if you move far away). When you have the data, you want that data to touch different regions. If your data is scaled wrongly, you can end up with all data points on the same region, making learning hard. Example. You can have most data to be large negative numbers. That will drive all relu on the first layer to be off, and you will stop learning as no signal will flow. So NN behaves well if initialized properly and data is scaled properly. Of course stronger optimizers plus other things can make this less of an issue.But I've never seen normalisation to hurt. But maybe if you have something specific in mind. By tradition I like to make the data to be between -1 and 1.

Anonymous

1 today, 9:53am

so, we should predict structure which we are expecting? (linear, 2-nd order etc.)?

Razvan: Not sure I understand the question. You mean when I was showing the polynomial fitting. No we don't have to. We take a super class like large NNs and regularize them. But yeah matching the family of function to he right ones (the ones with the right complexity) should help to generalize and classical ML does that. E.g. if the relationship is assumed to be linear, you do linear regression and account for the difference as observational noise.

Anonymous

1 today, 10:00am

if the problem is convex, finding the global optimum point is possible

Razvan: Sure. But NN are not convex. And convex models are not as powerful from a representational perspective to learn complex functions.

Tis

1 today, 10:02am

is there any special Deep Learning techniques to solved chaos and complex in forecasting economic issue?

Razvan: You can use NN to model dynamical systems. Actually it is becoming quite a trend now. E.g. https://arxiv.org/abs/1806.07366 But I don't know what you mean by solving chaos. That seems like a tall order. They can be more or less successful at capturing complex temporal relationships. But forecasting economy is more (in my view) complex than mimicking some known dynamical system. And you have all kinds of other issues, like a low data regime as well. I'm no expert though and this feels like a domain where you need to have some expertise to know what you are doing.

Amaka

1 today, 10:03am

For regularisation, how do you determine the degree of polynomial to use

Razvan: That was an illustrative example. In general you can't. Unless you have domain knowledge. But that is why we don't use polynomials to fit data. They do not exhibit this tendency to generalize like neural networks, so you need to find the right polynomial.

For neural networks, because of all the implicit regularization and learning you can just pick a really large network.

Amin Honarmandi SHandiz

1 today, 10:06am

how can i understand about loss function? which one is better to use?

Razvan: I skipped that part. If you take a probabilistic interpretation of learning, then for any type of data, and output layer there is a *natural* choice given by the probabilistic interpretation of the net. But typically regression you use mean square error. Softmax you use negative log likelihood. These choices are natural because they come from interpreting the output of your model as belonging to some distribution (e.g. for regression is Gaussians, and the KL between guassians will naturally lead to MSE error). If you look up Bishop's book on machine learning it has the derivation. If you ping me (and insist) I might be able to find the derivation from some other slides.

Anonymous

1 today, 10:09am

Could you please describe the way you decide minimum number of observations for each class for neural net to learn?

Razvan: Not sure what you mean. Neural networks usually operate in the large data regime. There is no way of estimating exactly the number of data points that you need to train a neural net of a given size. Not to mention that it matters a lot also how this parameters are used (e.g. is it a narrow deep net, or a wide shallow one). Maybe you can estimate the order of magnitude of the dataset (i.e. is it thousands, is it tens of thousands, is it millions for the entire dataset). But maybe as a high level point of view, you can successfully train neural nets where you have less datapoints than parameters. But again, what do we mean by training. What kind of loss in performance are we ok of losing by reducing the parameters. I'm also not talking about ;semi-supervised, or transfer learning or meta-learning or anything fancy like that. That will be a very different discussion.

Volviane

1 today, 10:20am

What is a non parametric model?

Razvan: Very good question. I hope it is somewhat clear what a parametric model is. A non-parametric model typically relies on the training set to construct the output. Maybe an example will help. Something like k nearest neighbour classifier. Where you take x, you look for the k-nearest neighbours in the training set by some predefined distance measure (e.g. euclidean distance but it can be something more funky) and you average the labels of those neighbours to produce the output for the input that you have. Or simply return the label to the closest neighbour

from the training set. The advantage of these approaches is that they need no learning. It usually relies on some search from the training set. So they can incorporate a new observation right away. While a neural net needs to do many updates before the function it learns responds to a new observation. The downside is that these methods are not as robust as parametric models. They do not generalize as well. People have been thinking of ways to combine these two classes of models, to have the fast (1-shot) acquisition of new information of non-parametric, but the robustness of parametric models.

Alessio Brini

1 today, 10:25am

What are some examples of compression functions we can use to make the combination operator in RNN more expressive?

Razvan: Typical RNNs (and I mean old school ones, not LSTMs) are using h_{t+1} = sigmoid (W h_t + Ux + b). This is a generalised linear function. It is a convex function and it is not a universal approximation.

Something better could be the LSTM update, which includes some extra gating which does increase expressivity. Or better an MLP where you get something like:

h_{t+1} = sigmoid (W_2relu(W_1 h_t + Ux + b_1) + b_1)

This will make the compression function a 1-layer MLP. It could be a 2-layer MLP. It could be a resnet-50 if you want :).

Just a note. I like this a lot as an idea, but it hasn't been proved popular so use this at your own risk. I think it should help. I was playing with this kind of ideas here: http://arxiv.org/abs/1312.6026

Anonymous

1 today, 10:35am

Why bother about RNN if LSTM is designed specifically to remember and forget things?

Razvan: Maybe I could have skipped RNNs for the sake of time .. I've felt like I barely got to explain all I wanted to do. But I do feel that without understanding RNNs you will not understand LSTMs. Ignoring the basics will hurt your understanding later on. That is why I wanted to say why backprop starts from the top. Because there are a lot of amazingly good people out there, but who actually don't understand why backprop is not just the chain rule, or how it works. Is just a

call of tf.grad or what not. And that kind of understanding is good to run things, but is not enough to understand and be able to comfortably play with models.

Another note. LSTM doesn't solve forgetting. They enlarge how far we can see in the past from say 20-50 steps to 200-400 steps (depending on how complex the data is). That is why transformers now are so popular, because they can expand beyond the 200 limit of LSTMs.

Anonymous

1 today, 10:37am

if initialization is super important we usually have only limited access to it in most DL frameworks?

Razvan: Two things. Initialization is super important to understand learning and how things work. Changing the seed or toying with initialization a little bit (e.g. scaling a bit the sampling interval) will not help you get better numbers. Libraries are built to efficiently run models and get numbers and apply DL to different things. And it makes sense to hide the initialization since that is not what a practitioner needs to play with.

So they are hidden since you don't need to hypertune them. They are only useful to understand learning dynamics of neural nets (and the learning process). But even then, most of the time you want to understand gradient dynamics under normal conditions (and people include the initialization into those normal conditions).

Alessio Brini

1 today, 10:40am

Is there a relationship between Glorot and He init with respect to the choice of the layer activation?

Razvan: Glorot is derived I think for both ReLU and tanh. He is meant for ReLU. He approximates better how the variance of the input will affect the variance of the output. So He is an improvement over Glorot in the sense that it is derived from a tighter bound for ReLU.

Anonymous

0 today, 9:42am

could you differentiate matrices & vectors

Razvan: Sure .. a lot of things are differentiable. E.g. Even computing the inverse or eigendecomposition can be a differentiable process. But yeah most of the time continuous things are differentiable.

Sara

0 today, 10:03am

I've over-fitting problem; can be seen from the loss error increasing after some number of neurons,but can't show that in plot of training-testing, how to show?

Razvan: Not sure I got the question. Maybe ping me.