August 2000

EFFICIENT BACKPROP

A great Paper that looks at issues that you can run into with Back prop and it then presents solutions and trys to explain why these solutions Work.

Authors note that their is no silver bullet for desiging NN'S as their design is problem and data dependent. But there are usefull heuristics.

So the note the bias v. Variance trade off Where bias is the diffruce between the network output across all data sets and the desired Functions output. And Variance is how much the network output varies from dataset to dataset.

obviosly as you train more your bias will go down as you learn the underlying function. But if you train for too long the variance will go up as you learn the noise of this specific dataset. (that's Bad). So the goal is to minimize both Variance and bias.

So they state that their is no guarantee that the network will 1) converge to a good solution 2) converge fast 3) even converge at all.

Now for ways to converge to better Solutions + do it faster

A) Stochastic V.S. Batch learing

- In Batch learnig you comput the Avg gradient across the whole train set + then take a Step in that direction.

But in Stochastic or "online" learing you pick a random sample from X + then Step in the direction of that example. (a more noisy step).

They claim that Stochastic learnig:

1. Often MUCH faster
2. Often Better Solutions are found
3. Can be used for tracking changes.

There is often lots of redundecy in a dataset Because It hopefully has patterns across the data, so on-line does not look at all the examples saving time on the redundency.

The Better solutions occur Because In Batch GD it will converge to the minima of the basin that the weights were initialized in however with on-line it allows it to Jump to new basins because of the noisy steps also on point #3 if the function you are trying to model constantly changes then On-line learing allows you to constatly tak in a new sample & train and to model will slowly change if the underlying model also is changing in real life.

But the very noise that helps SGD also hurts it.
- Because of the noise we can't use Second order methods for SGD
- Because of the noise the Learning may never converge. But Just fluctuate around the local minima

However in practice SGD is prefered

B) Example order
- for Batch this does not matter
- for SGD this does matter
- Networks learn the most from examples that are the most un~~fa~~familiar.

- we can shuffle Data So that Succesive examples are not from the same class

- Present inputs tha cause larger error more frequantly

note be carefull as this perterbs the importance the network places on diffrent types of examples

c) Normalizing Inputs

- If we normalize input values training is ussualy faster

- Also good to decoralate inpust or don't use if they are like $X_2 = 2X_1$ the don't use X_2 as this will slow down learning.

D) Sigmoid

- If not normalized input than Learing can often not happen

E) chossing Target values

- Don't use (-1, 1) for classifiction lables as will cause weight updates to be very small when using sigmoid.
- Also this would mean you get no indication of confidance when Inputs are near the decison Boundry.
- So pick target to maximize the second derivte of sigmoid to fix this problem

F) you don't want weights to be too big and too small as both screw up learning with sigmoid so pick weights from Distribution with mean 0 & $Sd = m^{-1/2}$ where $m = \#$ input features

G) choosing a Learning Rate

- give each weight its own learning rate
- lower layers lower Learning rate
- Higher layers Higher learning rate

E) Momentum can help if loss surface is nonspherical

They also note Adoptive learning rates & Radial Basis Functions.

Now on the topic of gradient decent convergance for some of the theory behind the earlyer tricks.

if α_{opt} = optimal learning rate

then $\alpha < \alpha_{opt}$ = too small of steps

then $\alpha > \alpha_{opt}$ = too large and will occilate but eventually converge

then $\alpha > 2\alpha_{opt}$ = it will Diverge

They then Note several Methods for improving training speed via second order methods often using the Hessian to account for the curvature + there for take a Better learning Step.

They then discuss and present Several Ways to compute the Hesian (or it's aproximate at least)

They then go on to explain Why large eIgenValues are a problem.

After all this disscussion they present methods on how to Impliment this in practice