TTIC 31230 Fundamentals of Deep Learning, 2020

Problems For Trainability.

Problem 1. This problem is on initialization. Consider a single unit defined by

$$y = f(W[I]x[I] - B) = f\left(\left(\sum_i W[i]x[i]\right) - B\right)$$

where B is initialized to zero and f is an activation function such as a sigmoid or ReLU. The vector x is a random variable determined by a random draw of a training example. Assume that the components of x are independent and that each component has zero mean and unit variance. Suppose that we initialize each weight in W from a distribution with zero mean and variance σ^2 and that the distribution is symmetric about zero — (the probability that $w[i] = z$ equals the probability that $w[i] = -z$). For example, $x[i]$ might be distributed as a zero-mean unit-variance Gaussian. Consider $y = \sum_i W[i]x[i]$ as a random variable defined by the distribution on x and the independent random distribution on W. Recall that the variance σ^2 of a sum of independent random variables is the sum of the variances and the variance of a product of zero mean independent random variables is the product of the variances.

(a) What value of σ for $W[i]$ gives zero mean and unit variance for y if the vectors $w[I]$ and $x[I]$ have dimension d? Show your derivation.

Solution: Let σ^2 be the variance of $x[i]$. We then have that the variance of $\sum_i W[i]x[i]$ is $\sum_i \sigma^2 = d\sigma^2$. Setting $d\sigma^2$ equal to 1 gives

$$\sigma = \frac{1}{\sqrt{d}}$$

(b) For a sigmoid activation function what is the mean of u.

Solution: We are given that the probability that $W[i] = z$ is the same as the probability of $w[i] = -z$. This implies that for a given value of $x[i]$ we have that the probability that $w[i]x[i] = z$ equals the probability that $w[i]x[i] = -z$. This further implies that, for a given value y, the probability that $\sum_i w[i]z[i] = y$ equals the probability that $\sum_i w[i]x[i] = -y$. So the input to the sigmoid is distributed symmetrically about 0. Since the sigmoid function is itself symmetric about 0, we get that the expected value of the output of the sigmoid is its value at zero which is 1/2.

(c) For a sigmoid activation function is the variance of u larger than, equal to, or smaller than the variance of y?

Solution: The variance is smaller. To show this it suffices to show that the slope of the sigmoid function is everywhere less than 1. The slope is largest at the input zero. The sigmoid function is

$$f(z) = \frac{1}{1+e^{-y}}$$

The slope is

$$f'(y) = \frac{e^{-y}}{(1+e^{-y})^2}$$

which equals $1/4$ at $y = 0$.

(d) What is the largest possible variance of the output of a sigmoid?

Solution: The larest valriance occurs when $y = \infty$ with probability $1/2$ and $y = -\infty$ with probability $1/2$;-). Tn this case $f(y)$ is 0 with probability $1/2$ and 1 with probability $1/2$. Which gives a variance of $1/4$.

Problem 2. Consider a regression problem where we want to predict a scalar value y from a vector x. Consider the L-layer perceptron for this problem defined by the following equations which compute hidden layer vectors $h_1[I], \ldots, h_L[I]$ and predictions $\hat{y}_1, \ldots, \hat{y}_L$ where the prediction $\hat{y}_\ell$ is done with a linear regression on the hidden vector $h_\ell[I]$.

$$\begin{aligned}
h_0[i] &= x[i] \\
&\vdots \\
h_{\ell+1}[i] &= \sigma(W^{h,h}_{\ell+1}[i,I]h_\ell[I] - B^{h,h}_{\ell+1}[i]) \\
\hat{y}_{\ell+1} &= W^{h,p}_{\ell+1}[I]h_{\ell+1}[I] - B^{h,y}_{\ell+1} \\
&\vdots \\
\text{Loss} &= \sum_{\ell=1}^{L}(y - \hat{y}_\ell)^2
\end{aligned}$$

Each term $(y - \hat{y}_\ell)^2$ is called a "loss head" and defines a loss on each prediction $\hat{y}_\ell$. Note, however, that there is only one scalar loss minimized by SGD which is the sum of the losses of each loss head.

(a) Explain why these multiple loss terms might improve the ability of SGD to find a useful L-layer MLP regrssion $\hat{y}_L$ when L is large.

Solution: SGD on deep networks with the loss term only occuring at the final layer is not generally effective because the lower layers never get meaninful gradients. Placing loss functions near the lower layers will cause the lower hidden layers to have meaningful gradients and produce informative features.

(b) As a function of L (ignoring the dimension size I) what is the order of run time for the backpropagation procedure. Explain your answer.

Solution: It is $O(L)$ — linear in L. Backpropagation loops over the assignments of the program and takes time proportional to the size of the program. Back-propagation over the final sum of losses produces a gradient for each prediction $\hat{y}_\ell$ which can be used as we back-propagate over the earlier assignments.

(c) Rewrite the above MLP equations to use residual connections rather than multiple heads. There are multiple correct solutions differing in minor details. Pick one that seems good to you.

Solution:

$$\begin{aligned}
h_0[i] &= x[i] \\
&\vdots \\
\tilde{h}_{\ell+1}[i] &= \sigma(W^{h,h}_{\ell+1}[i, I]h_\ell[I] - B^{h,h}_{\ell+1}[i]) \\
h_{\ell+1}[i] &= \tilde{h}_{\ell+1}[i] + h_\ell[i] \\
&\vdots \\
\hat{y} &= W^{h,y}[I]h_L[I] - B^{h,y} \\
\text{Loss} &= (y - \hat{y})^2
\end{aligned}$$

Problem 3. Consider a bottleneck multi-layer perceptron (MLP) with residual connections defined as follows where N_{bottle} is smaller than $N_{\text{in}} = N_{\text{out}}$.

$$\begin{aligned}
\tilde{L}_\ell[n_{\text{bottle}}] &= \text{ReLU}(W^{b,1}_\ell[n_{\text{bottle}}, N_{\text{in}}]L_\ell[N_{\text{in}}] - B^{b,1}_\ell[n_{\text{bottle}}]) \\
\hat{L}_\ell[n_{\text{out}}] &= \text{ReLU}(W^{b,2}_\ell[n_{\text{out}}, N_{\text{bottle}}]\tilde{L}_\ell[N_{\text{bottle}}] - B^{b,2}_\ell[n_{\text{out}}]) \\
L_{\ell+1}[n] &= L_\ell[n] + \hat{L}_\ell[n]
\end{aligned}$$

(a) What is the number of multiplications done by this network as a function of $N_{\text{in}} = N_{\text{out}} = N$, N_{bottle} and the number of layers L (including the input layer)? Under what conditions does this give fewer multiplications than the standard MLP with one matrix between layers?

Solution: The number of multiplications is $2NN_{\text{bottle}}(L-1)$. For a standard MLP (with no botleneck) the number of multiplications is $N^2(L-1)$. The bottleneck layer has fewer multiplications for $N_{\text{bottle}} < N/2$.

(b) We now consider introducing a multiplicative constant γ into the residual connection.

$$L_{\ell+1}[n] = \gamma(L_\ell[n] + \hat{L}_\ell[n])$$

If the network is initialized such that each response of $L_\ell[n]$ and $\hat{L}[n]$ has zero mean and unit variance, and are assummed to be independent, what value of γ gives that $h[\ell+1, j]$ has zero mean and unit variance.

Solution: $1/\sqrt{2}$

(c) The main advantage of a stack of residual connections is that there is direct additive path from the loss to each layer of the stack, including the input layer. Give a reason why the introduction of the constant $\gamma < 1$ as in part (b) might be damaging to the optimization of the lower layers of the residual stack.

Solution: When we introduce $\gamma < 1$ as in (b) the gradient update on the bottom layer is reduced by γ^{L-2}. This could harm the learning along the direct connection between the loss and the first layer of the network.

Problem 5. RNN run time. Consider an autoregressive RNN neural language model with $P_\Phi(w_{t+1}|w_1, \ldots, w_t)$ defined by

$$P_\Phi(w_t|w_1, \ldots, w_{t-1}) = \operatorname*{softmax}_{w_{t+1}} \; e[w_t, I]h[t-1, I]$$

Here $e[w, I]$ is the word vector for word w, $h[t, I]$ is the hidden state vector at time t of a left-to-right RNN, and as described above $e[w, I]h[t, I]$ is the inner prodcut of these two vectors where we have assumed that they have the same dimension. For the first word w_1 we have an externally provided initial hidden state $h[0, I]$ and $w_1, \ldots, w_0$ denotes the empty string. We train the model on the full loss

$$\begin{aligned}
\Phi^* &= \operatorname*{argmin}_\Phi E_{w_1,\ldots,w_T \sim \text{Train}} \; -\ln P_\Phi(w_1, \ldots, w_T) \\
&= \operatorname*{argmin}_\Phi E_{w_1,\ldots,w_T \sim \text{Train}} \; \sum_{t=1}^{T} -\ln P_\Phi(w_t|w_1, \ldots, w_{t-1})
\end{aligned}$$

What is the order of run time as a function of sentence length T for the backpropagation for this model run on a sentence $w_1, \ldots, w_T$? Explain your answer.

Solution: The backprogation takes $O(T)$ time (not $O(T^2)$). The model consists of $O(T)$ objects each of which performs a single forward operation and a single backward operation. As the backpropagation procedes more of the loss terms in the sum over t get incorporated. It should be noted that RNNs take time linear in the sequence length even on parallel hardware. The transformer, on the other hand, takes constant time in parallel independent of the length of the sequence. This is a major advantage of the transformer.