TTIC 31230 Fundamentals of Deep Learning
Problem set 4
Due Thursday, February 8, 11:59 pm

Problem 1. (10 points) This problem is on initialization (slide 23 of lecture 4). Consider a single unit defined by $u = Wx + b$ where u and v are vectors, W is a $d_u \times d_x$ matrix and b is a bias vector initialized to zero. Assume that each component of x has zero mean and unit variance. Suppose that we initialize each weight in W from a zero mean Gaussian distribution with variannce σ. Consider u as a random variable defined by the distribution on x and the independent random distribution on W.

a) What value of the initialization variance σ gives zero mean and unit variance for each component of u? Show your derivation.

Each component of u can be written as

$$u_i = \sum_j W_{i,j} x_j.$$

If W and x are independent with zero mean then u_i has zero mean.

The variance of a sum of independent random variables is the sum of the variances. So we have

$$\begin{aligned} Eu_i^2 &= \sum_j EW_{i,j}^2 x_j^2 \\ &= \sum_j (EW_{i,j}^2)(Ex_j^2) \\ &= d_x \sigma^2 \end{aligned}$$

Setting this equal to 1 gives $\sigma = 1/\sqrt{d_x}$.

b) Consider $u' = \mathrm{relu}(Wx + b) - b'$. If we want u' to have zero mean and unit variance will this increase or decrease the variance of the random distribution used to initialize W? Explain your answer.

For any random variable y such that $P(y < 0) > 0$ we have that the variance of $\max(0, y)$ will be less than the variance of y. So to keep the variance at 1 we must **increase** the variance σ of the weights $W_{i,j,}$.

Assuming that b is initialized to zero, the appropriate value of σ is $\sqrt{\frac{c}{d_x}}$ with

$$c = \frac{1}{\frac{1}{2} - \frac{1}{2\pi}} \approx 3$$

Problem 2. **(10 points)** This problem is on RNNs. Explain why gating is needed for the highway path (skip connections) in RNNs but not in resnet.

As stated in the notes (slide 44 of lecture 4 and slide 8 of lecture 5), resnet has different parameters for each layer and the final matrix computing the value of the "diversion" computation can use layer-specific weighting for the diversion values. In the case of gated RRNs each time step uses the same parameters and so the weighting of the diversion components must be data-dependent.

Problem 3. **(20 points)** What is the order of the number of floating point operations (serial computer running time) as a function of input sentence length and output sentence length for both the forward and backward pass of sequence to sequence models for machine translation both with and without attention.

Let n be the length of the input sentence and let m be the size of the output sentence.

The number of floating point operations in the forward computation (as a function of input and output sentence length) is proportional to the size of the computation graph. Without attention this is $O(n+m)$. With attention the computation graph includes the computation of the weights $\alpha_{i,s}$ for input position i and output position s. Hence the size of the computation graph is $O(nm)$. The backward computation simply calls the backward method on each node of the computation graph in the reverse order. Hence the size of the number of floating point operations in the backward pass is also proportional to the size of the graph which is $O(n+m)$ without attention and $O(nm)$ with attention.

Problem 4. **(10 points)** Suppose that we adjust the learning rates so that the update of SGD and RMSprop have the same norm — so that

$$||\Phi_{t+1} - \Phi_t||$$

is the same in both cases. Assuming both updates start at the same value Φ_t will Φ_{t+1} be in the same? Explain your answer.

They will in general be in different directions. As a simple example consider the case where Φ.grad has the same value in all components but the variance $s[c]$ is different for different parameters c. In this case we have that the update of SGD is the same in all parameters but the update of RMS prop will be proportional to $1/\sqrt{s[c]+\epsilon}$ for parameter c which gives different updates in different directions. Hence the two updates are not in the same direction.

Problem 5. **(40 points)** Consider SGD applied to the following.

$$W^* = \underset{W}{\text{argmin}}\ E_{x\sim\text{Train}}\ \text{loss}(Wx)$$

where x is a vector and W is a matrix.

a) Write the expression for the SGD update computing W_{t+1} from W_t, x_t, the gradient ∇loss, and with learning rate η.

Here we have that ∇loss is a vector of dimension d where W has shape (d,k) and x has dimension k.

$$W^{t+1}[i,j] = W^t[i,j] - \eta \nabla \text{loss}[i] x_t[j]$$

b) Suppose that we define $\tilde{x}_t$ by doubling the first input feature.

$$\begin{aligned} \tilde{x}_t[0] &= 2x_t[0] \\ \tilde{x}_t[j] &= x_t[j] \ \text{ for } j \neq 0 \end{aligned}$$

and simultaeously making a compensating change in W_0

$$\begin{aligned} \tilde{W}_0[i,0] &= W_0[i,0]/2 \\ \tilde{W}_0[i,j] &= W_0[i,j] \ \text{ for } j \neq 0 \end{aligned}$$

Consider SGD running from W_0 on x_0, x_1, ... and SGD running from $\tilde{W}_0$ on $\tilde{x}_0$, $\tilde{x}_1$, ... Show that these are not equivalent — they do not produce the same sequence of vectors $W_t x_t$ (consider the first update).

Let W and x be as defined in part (a) and let $\tilde{W}$ and $\tilde{x}$ be the modifications of W and x as defined in part (b). Since $\tilde{W}\tilde{x} = Wx$ we have that ∇loss is the same in both cases.

$$\nabla_W \text{loss}[i]\tilde{x}_0[j] = \nabla_W \text{loss}[i](2x_0[j]) = 2\nabla \text{loss}[i] x_0[j]$$

This gives

$$\begin{aligned} \tilde{W}_1[i,0] &= \tilde{W}_0[i,0] - \eta \nabla \text{loss}[i]\tilde{x}_0[0] \\ &= \frac{1}{2}W_0[i,0] - 2\eta\nabla \text{loss}[i], x_0[0] \\ &= \frac{1}{2}W_1[1,0] - \frac{3}{4}\eta\nabla \text{loss}[i] x_0[0] \end{aligned}$$

This gives that $\tilde{W}_1[i,0] \neq \frac{1}{2}W_1[i,0]$ and hence $\tilde{W}_1[i,0]\tilde{x}_1[0] \neq W_1[i,0]x_1[0]$.

c) Explain why part b implies that SGD is sensitive to the choice of feature units (pounds vs. kilograms).

The change in scaling in part (b) can correspond to a change in units. A change in units can cause a change in the SGD algorithm even when the weights are adjusted to be in inverse units.

d) Define an SGD update on $\tilde{W}$ that maintains the invariant $\tilde{W}_t[i,0] = W_t[i,0]/2$.

$$\begin{aligned} \tilde{W}_{t+1}[i,j] &= \tilde{W}_t[i,j] - \eta[j]\nabla \text{loss}[i] x_t[j] \\ \eta[0] &= 1/4 \\ \eta[j] &= 1 \ \text{ for } j > 0 \end{aligned}$$

This gives $\tilde{W}_1[i,0] = \frac{1}{2}W_1[i,0]$ which will be maintained as an invariant.

e) Define

$$\begin{aligned} \sigma[i] &= \sqrt{E_{x\sim\mathrm{Train}}\,(x[i]-\mu[i])^2} \\ \mu[i] &= E_{x\sim\mathrm{Train}}\,x[i] \end{aligned}$$

Define an SGD update involving $\sigma[i]$ such that the resulting sequence of vectors W_0x_0, W_1x_x, W_2x_2, ... remains the same when we rescale the components of x and inverse scale the initial components of W_0. Compare your "units independent" update to RMSProp.

$$\tilde{W}_{t+1}[i,j] = \tilde{W}_t[i,j] - \frac{\eta}{\sigma[j]^2}\nabla\mathrm{loss}[i]x_t[j]$$

Here σ will be proportional to the scaling of $x[j]$ which is analogous to the factor of 2 in parts (b) and (d).

Note that σ has the same units as x and W has inverse units of x so that Wx is dimensionless.

If ∇loss has units of bits then the learning rate η should be in inverse bits. Hence the learning rate would be sensitive to whether we ln or $\log_2$ in the definition of the loss.

If the gradient vector ∇loss is constant over time then the RMSProp update becomes

$$\tilde{W}_{t+1}[i,j] = \tilde{W}_t[i,j] - \frac{\eta'}{\sigma[j]+\epsilon}\nabla\mathrm{loss}[i]x_t[j]$$

which seems similar but is different.

Problem 6. (10 points) Explain why the more general formulation of the free lunch theorem on slide 26 of lecture 7 implies the special case on slide 24. (Assume that the compression of h is a null terminated byte srtring. Note that the set of all null-terminated byte srtings is prefix-free.)

For any prefix-free code $C(x)$ we can consider a probability distribution on the code words defined by flipping a coin to set the first bit, then flipping a coin to determine the second bit, and so on until we have reached a code word. This defines a probability distribution on code words with the probability of a code word C being exactly $2^{-|C|}$where $|C|$ denotes the number of bits in C. Hence the code defines a distribution on x with $P(x) = 2^{-|C(x)|}$. Inserting this distribution into the probability form of the free lunch theorem gives the code length form of the theorem.