TTIC 31230 Fundamentals of Deep Learning
Midterm Exam Practice Questions

Problem 1. We consider an initialization-only "simple normalization" layer applied to a layer.

$$L' = \texttt{SimpleNorm}(\texttt{L}, \texttt{A}, \texttt{B})$$

```
class SimpleNorm:

    def __init__(self,L,A,B)
        ...

    def forward(self)
        self.value = self.A.value*(self.L.value + self.B.value)

    def backward(self)
        ...
```

Here B should be initialized to $-\mu$ and A should be initialized to $1/\sigma$ but should then be trained independently of measuring μ and σ within batches. (This was previously called "Karpathy Normalization" but it is not actually what Karpathy proposed — Karpathy proposed adding a "simple normalization" immediately after a standard batch normalization so that the network could learn to undo the batch normalization if it wanted to. See lecture 5 of CS231 slide 69. Thanks to Adam Dziedzic for pointing this out).

Assume that A and B have shape (C) where C is the number of channels. Assume that L has shape (B, H, W, C) where B is the minibatch size, H is the height dimension, W is the width dimention, C is the number of channels.

a. write the forward method in `+=` notation.

$$L'[\ldots] \texttt{ += } A[\ldots](L[\ldots] + B[\ldots])$$

(fill in the indices in the above equation).

$$\forall b, x, y, c \quad L'.\text{value}[b, x, y, c] \texttt{ += } A.\text{value}[c] * (L.\text{value}[b, x, y, c] + B.\text{value}[c])$$

b. Write the `+=` notation for the backward method (to L, A and B).

$\forall b, x, y, c$

$$
\begin{aligned}
L.\text{grad}[b,x,y,c] \quad &\texttt{+=} \quad L'.\text{grad}[b,x,y,c]A.\text{value}[c] \\
A.\text{grad}[c] \quad &\texttt{+=} \quad L'.\text{grad}[b,x,y,c](L.\text{value}[b,x,y,c] + B.\text{value}[c]) \\
B.\text{grad}[c] \quad &\texttt{+=} \quad L'.\text{grad}[b,x,y,c]A.\text{value}[c]
\end{aligned}
$$

c. Write the Python/Numpy code for the backward method using appropriate Numpy vector operations.

$$
\begin{aligned}
L.\text{grad} \quad &\texttt{+=} \quad L'.\text{grad} * A.\text{value} \\
A.\text{grad} \quad &\texttt{+=} \quad (L'.\text{grad} * (L.\text{value} + B.\text{value})).\text{sum}(\text{axis} = (0,1,2)) \\
B.\text{grad} \quad &\texttt{+=} \quad (L'.\text{grad} * A.\text{value}).\text{sum}(\text{axis} = (0,1,2))
\end{aligned}
$$

Problem 2. This problem is on momentum. Momentum is traditionally defined in terms of "velocity"

$$
\begin{aligned}
\hat{v}^{t+1} \quad &= \quad \mu v^t + \eta \, \nabla_\Theta \, \ell^t(\Theta) \\
\Theta \quad &\texttt{-=} \quad v^{t+1}
\end{aligned}
$$

In this class we have defined momentum by a smoothing operation on the gradient vector.

$$
\begin{aligned}
\hat{g}^{t+1} \quad &= \quad \mu \hat{g}^t + (1-\mu) \nabla_\Psi \, \ell^t(\Psi) \\
\Psi^t \quad &\texttt{-=} \quad \eta' \hat{g}^{t+1}
\end{aligned}
$$

Assume that $\Psi^0 = \Theta^0$ and that $v^0 = \hat{g}^0 = 0$ and define

$$\eta' = \frac{\eta}{1-\mu}$$

Show by induction on t that we have

$$\begin{aligned} v^t &= \eta' \, \hat{g}^t \\ \Psi^t &= \Theta^t \end{aligned}$$

The base case is given by $\Psi^0 = \Theta^0$ and that $v^0 = \hat{g}^0 = 0$.

For the induction case we have

$$\begin{aligned} v^{t+1} &= \mu v^t + \eta \nabla_\Theta \, \ell(\Theta)@\Theta^t \\ &= \mu(\eta' \hat{g}^t) + \eta \nabla_\Psi \, \ell(\Psi)@\Psi^t \\ &= \eta' \left(\mu \hat{g}^t + \frac{\eta}{\eta'} \nabla_\Psi \, \ell(\Psi)@\Psi^t \right) \\ &= \eta' \left(\mu \hat{g}^t + (1-\mu) \nabla_\Psi \, \ell(\Psi)@\Psi^t \right) \\ &= \eta' \hat{g}^{t+1} \end{aligned}$$

And this implies $\Psi^{t+1} = \Theta^{t+1}$

The semantics of μ and η' seem clearer in the gradient smoothing formulation. In particular, we expect the optimum value of η' to be near the optimal value of η when momentum is not used. We also expect that μ and η' are more nearly conjugate in meta-parameter search (we can optimize μ and η' independently).

Problem 3. Adam uses the equations.

$$\begin{aligned} \hat{g}_i^{t+1} &= \beta_1 \hat{g}_i^t + (1-\beta_1) \left(\nabla_\Theta \, \ell^t(\Theta) \right)_i \\ s_i^{t+1} &= \beta_2 s_i^t + (1-\beta_2) \left(\nabla_\Theta \, \ell^t(\Theta) \right)_i^2 \\ \Theta_i^{t+1} &= \Theta_i^t - \frac{\eta}{\sqrt{s_i^{t+1}} + \epsilon} \; \hat{g}_i^{t+1} \end{aligned}$$

Suppose that for each scalar parameter Θ_i we have that $(\nabla_\Theta \ell^t(\Theta))_i$ is either 1 or 0 and the fraction of time that it is 1 is ϵ. If we want s_i^t to be a reasonable estimate of $\mathrm{E}\left[(\nabla_\Theta \ell^t(\Theta))_i^2\right]$ what is a reasonable value of β_2 as a function of ϵ? Explain your answer.

In order get an accurate estimate of the average value of $(\nabla_\Theta \ell^t(\Theta))_i^2$ we will need to average over a number of examples where the expected number of cases where $(\nabla_\Theta \ell^t(\Theta))_i$ is 1 is significant. Assuming that the update equations are applied per batch we will want to average over a number k of batches satisfying say

$$\epsilon b k \geq 5$$

or

$$k \geq \frac{5}{\epsilon b}$$

avergaging over k batches corresponds to

$$\beta_2 = 1 - \frac{1}{k}$$

or

$$\beta_2 = 1 - \frac{b\epsilon}{5}$$

Problem 4. An intuitive prior on numbers is the log-uniform prior where $\log_2 |\Theta_i|$ is taken to be uniformly distributed between, say, -10 and 10. A log-uniform prior corresponds to a density

$$p(\Theta_i) \propto 1/|\Theta_i|$$

This correspinds to the following regularizer.

$$\Theta^* = \operatorname*{argmin}_\Theta \ell(\Theta) + \lambda \sum_i \ln |\Theta_i|$$

Assuming that we only consider positive value of Θ_i (to make things simpler) give the conditions on $\partial\ell/\partial\Theta_i$ for a local optimum (or rather a stationary point — a point where the derivative of the objective function is zero).

$$\frac{\partial \ell}{\partial \Theta_i} + \lambda \frac{1}{\Theta_i} = 0$$

Problem 5. This problem is on complex-step differentiation. Suppose that Θ is a parameter tensor and that we have computed Θ.grad using complex arithmetic

at the parameter setting $\Theta + i\epsilon\Delta\Theta$ where $\epsilon = 2^{-50}$. Write the expression for $H\Delta\Theta$ where H is the Hessian in terms of the complex value for Θ.grad.

$$H\Delta\Theta = \frac{Im\left[\Theta.\text{grad@}(\Theta + i\epsilon\Delta\Theta)\right]}{\epsilon}$$

Problem 6. This problem concerns initialization. Consider a unit defined by a simple inner product of a weight vector and previous units followed by a simple normalization and then a nonlinearity.

$$y = \sigma(W \cdot x)$$

or in component notation

$$\begin{aligned} y &= \sum_i w_i x_i \\ z &= \sigma(a(y+b)) \end{aligned}$$

a. Assume that the x_i are independent and have mean μ_x and variance σ_x^2. Assume that the the w_i are initialized independently to have mean μ_w and variance σ_w^2. If we want $a(y+b)$ to have zero mean and unit variance, how should we initialize a and b?

This problem should have given the dimension of w and x. Say the dimension is d. Then we have

$$\begin{aligned} b &= -\text{E}\left[\sum_{i=1}^{d} w_i x_i\right] = -d\mu_w\mu_x \\ \frac{1}{a} &= \sqrt{\text{E}\left[\left(\left(\sum_i w_i x_i\right) - d\mu_w\mu_x\right)^2\right]} \\ &= \sqrt{\text{E}\left[\left(\sum_i (w_i x_i - \mu_w\mu_x)\right)^2\right]} \\ &= \sqrt{\text{E}\left[\sum_i (w_i x_i - \mu_w\mu_x)^2\right]} \quad \text{other terms are zero} \end{aligned}$$

$$= \sqrt{d\mathrm{E}\left[((\mu_w + \epsilon_w)(\mu_x + \epsilon_x) - \mu_w\mu_x)^2\right]}, \quad \epsilon_w = w - \mu_w, \quad \epsilon_x = x - \mu_x$$

$$= \sqrt{d\mathrm{E}\left[((\mu_w\epsilon_x + \mu_x\epsilon_w + \epsilon_x\epsilon_w)^2\right]}$$

$$= \sqrt{d\mathrm{E}\left[\mu_w^2\epsilon_x^2 + \mu_x^2\epsilon_w^2 + \epsilon_x^2\epsilon_w^2\right]} \text{ other terms are zero}$$

$$= \sqrt{d(\mu_w^2\sigma_x^2 + \mu_x^2\sigma_w^2 + \sigma_x^2\sigma_w^2)}$$

When I wrote this problem I did not realize the answer was so complex. Exam problems will be simpler.

b. If a and b are initialized so that $a(y+b)$ has zero mean and unit variance, is there any advantage to using Xavier initialization on w? Explain your answer.

y will have zero mean and unit variance after initialization whether or not we use Xavier initialization on w. However, Xavier initialization has an interaction with SGD in this case which is dificult to analyze. The first part of the answer would receive full credit and the second point would get extra credit.

Problem 7. Consider a highway path update of the form

$$L_{i+1} = F_i * L_i + (1 - F_i) * D_i(L_i)$$

where F_i is a parameter (is independent of the problem instance) rather than being computed from L_i. In a Resnet-like CNN the diversion $D_i(F_i)$ uses different parameters for each i. Assume that F_i can also be a different parameter for each i. Do you think this is a reasonable CNN architecture? Explain your answer (your explanation is more important than your answer).

The intended answer was that this would be a reasonable architecture provided that the diversions have independent parameters at each layer (as in Resnet). This is not a reasaonable architecture for an RNN where the diversions at the different layers have the same parameters. In the RNN case all time steps are presumed to be doing the same update and the forget gate must be value dependent.

Problem 8. Suppose that at training time we construct a mask μ on the parameters so that

$$\begin{cases} \mu_i = \frac{1}{2} & \text{with probability } \alpha \\ \mu_i = 1 & \text{with probability } 1 - \alpha \end{cases}$$

Then at train time we do

$$
\begin{aligned}
y_i &= \text{Relu}\left(\sum_j W_{i,j}\ \mu_j x_j\right) \\
\Theta &\texttt{ -= } \nabla_\Theta\ \ell(\Theta, \mu)
\end{aligned}
$$

Give a corresponding weight scaling rule for computing y_i at test time for this "half drop out" training algorithm.

We want the expectation of $\sum_j\ W_{i,j}\ \mu_j x_j$ at test time to match that at train time. This works out to be

$$y_i = \text{Relu}\left(\left(1 - \frac{\alpha}{2}\right)\sum_j\ W_{i,j}\ x_j\right)$$