## TTIC 31230 Fundamentals of Deep Learning, winter 2019

## Backpropagation Problems

**Problem 1:** Consider the following softmax.

$$\begin{aligned} Z[b] &= \sum_j \exp(s[b,j]) \\ p[b,j] &= \exp(s[b,j])/Z[b] \end{aligned}$$

An alternative way to compute this is to initialize the tensors $Z$ and $p$ to zero and then execute the following loops.

for $b, j$ $\quad Z[b]$ `+=` $\exp(s[b,j])$

for $b, j$ $\quad p[b,j]$ `+=` $\exp(s[b,j])/Z[b]$

Each individual `+=` operation inside the loops can be treated independently in backpropagation.

(a) Give a back-propagation loop over `+=` updates based on the second loop for adding to $s$.grad using $p$.grad (and using the forward-computed tensors $Z$ and $s$).

**Solution**: For $b, j$ $\quad s.\mathrm{grad}[b,j]$ `+=` $p.\mathrm{grad}[b,j]\exp(s[b,j])/Z[b]$

(b) Give a back-propagation loop over `+=` updates based on the second equation for adding to $Z$.grad using $p$.grad (and using the forward-computed tensors $s$ and $Z$).

**Solution**: For $b, j$ $\quad Z.\mathrm{grad}[b]$ `-=` $p.\mathrm{grad}[b,j]\exp(s[b,j])/Z[b]^2$

(c) Give a back-propagation loop over `+=` updates based on the first equation for adding to $s$.grad using $Z$.grad (and using the forward-computed tensor $s$).

**Solution**: For $b, j$ $\quad s.\mathrm{grad}[b,j]$ `+=` $Z.\mathrm{grad}[b]\exp(s[b,j])$

**Problem 2:** Show that the addition to $s$.grad shown in problem 1 can be computed using the following more efficient updates.

for $b, j$ $\quad e[b]$ `-=` $p[b,j]p.\mathrm{grad}[b,j]$

for $b, j$ $\quad s.\mathrm{grad}[b,j]$ `+=` $p[b,j](p.\mathrm{grad}[b,j] + e[b])$

**Solution**: The updates for problem 1 can be written as

$$
\begin{aligned}
\text{for } b \quad Z.\text{grad}[b] &= \sum_j -p.\text{grad}[b,j]\exp(s[b,j])/Z[b]^2 \\
&= \left(\sum_j -p[b,j]p.\text{grad}[b,j]\right)/Z[b] \\
&= e[b]/Z[b]
\end{aligned}
$$

$$
\begin{aligned}
\text{for } b,j \quad s.\text{grad}[b,j] &= p.\text{grad}[b,j]\exp(s[b,j])/Z[b] + Z.\text{grad}[b]\exp(s[b,j]) \\
&= p.\text{grad}[b,j]\left(\exp(s[b,j])/Z[b]\right) + e[b]\left(\exp(s[b,j])/Z[b]\right) \\
&= p[b,j](p.\text{grad}[b,j] + e[b])
\end{aligned}
$$

This formula shows how hand-written back-propagation methods for "layers" such as softmax can be more efficient than compiler-generated back-propagation code. While optimizing compilers can of course be written, one must keep in mind the trade-off between the abstraction level of the programming language and the efficiency of the generated code.

**Problem 3.** Consider the following set of `+=` statements defining batch normalization where all computed tensors are initialized to zero.

For $b,j$ $\mu[j]$ `+=` $\frac{1}{B}\,x[b,j]$

For $b,j$ $s[j]$ `+=` $\frac{1}{B-1}\,(x[b,j]-\mu[j])^2$

For $b,j$ $x'[b,j]$ `+=` $\frac{x[b,j]-\mu[b]}{\sqrt{s[j]}}$

Give backpropagation `+=` equations for computing $x.\text{grad}[b,j]$, $\mu.\text{grad}[j]$, and $s.\text{grad}[j]$ from $x'.\text{grad}[b,j]$.

**Problem 4.** The equations defining a UGRNN are given below.

$$
\begin{aligned}
R_t[b,j] &= \tanh\left(\left(\sum_i W^{h,R}[i,j]h_t[b,i]\right) + \left(\sum_k W^{x,R}[k,j]x_t[b,k]\right) - B^R[j]\right) \\
G_t[b,j] &= \sigma\left(\left(\sum_i W^{h,G}[i,j]h_t[b,i]\right) + \left(\sum_k W^{x,G}[k,j]x_t[b,k]\right) - B^G[j]\right) \\
h_{t+1}[b,j] &= G_t[b,j]h_t[b,j] + (1-G_t[b,j])R_t[b,j]
\end{aligned}
$$

(a) Rewrite this using `+=` loops instead of summations assuming that all computed tensors are initialized to zero. You will need to define two additional tensors for the inputs to the activation functions.

(b) Give `+=` loops for the backward computation of tensor gradients starting from $h_{t+1}.\text{grad}[B,J]$. You can write the derivatives of the activation functions tanh and $\sigma$ simply as $\tanh'$ and $\sigma'$ respectively.