TTIC 31230 Fundamentals of Deep Learning, 2020

Problems For Trainability.

In these problems, as in the lecture notes, capital letter indeces are used to indicate subtensors (slices) so that, for example, $M[I, J]$ denotes a matrix while $M[i, j]$ denotes one element of the matrix, $M[i, J]$ denotes the ith row, and $M[I, j]$ denotes the jth collumn.

Throughout these problems we assume a word embedding matrix $e[W, I]$ where $e[w, I]$ is the word vector for word w. We then have that $e[w, I]^\top h[t, I]$ is the inner product of the word vector $w[w, I]$ and the hidden state vector $h[t, I]$.

We will adopt the convention, similar to true Einstein notation, that repeated capital indeces in a product of tensors are implicitly summed. We can then write the inner product $e[w, I]^\top h[t, I]$ simply as $e[w, I]h[t, I]$ without the need for the (meaningless) transpose operation.

Problem 1. 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) 7 pts 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) 7 pts 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) 11 pts 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 2. Gated CNNs
A UGRNN is defined by the following equations.

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

Modify these to form a data-dependent data-flow CNN for vision — an Update-Gate CNN (UGCNN). More specifically, give equations analogous to those for

UGRNN for computing a CNN "box" $L_{\ell+1}[b,x,y,j]$ from $L_\ell[b,x,y,i]$ (stride 1) using a computed "gate box" $G_{\ell+1}[b,x,y,j]$ and an "update box" $R_{\ell+1}[b,x,y,j]$.

$$\Phi = (W^{L,R}_{\ell+1}[\Delta x, \Delta y, j, j'], B^R_{\ell+1}[j], W^{L,G}_{\ell+1}[\Delta x, \Delta y, j, j'],\ B^G_{\ell+1}[j])$$

Solution:

$$\begin{aligned}
R_{\ell+1}[b,x,y,j] &= \tanh\left(\left(\sum_{\Delta x,\Delta y,j'} W^{L,R}_{\ell+1}[\Delta x,\Delta y,j',j] L_\ell[b,x+\Delta x,\ y+\Delta y,\ j']\right) - B^R_{\ell+1}[j]\right) \\
G_{\ell+1}[b,x,y,j] &= \sigma\left(\left(\sum_{\Delta x,\Delta y,i} W^{L,G}_{\ell+1}[\Delta x,\Delta y,i,j] L_\ell[b,x+\Delta x,\ y+\Delta y,\ i]\right) - B^G_{\ell+1}[j]\right) \\
L_{\ell+1}[b,x,y,j] &= G_{\ell+1}[b,x,y,j] L_\ell[b,x,y,j] + (1 - G_t[b,x,y,j]) R_t[b,x,y,j]
\end{aligned}$$

Problem 3. RNNs
Below are the equations defining the update cell of the UGRNN which takes as data inputs $h[B,t-1,I]$ and $x[B,t,K]$ and produces $h[B,t,J]$.

$$\begin{aligned}
R[b,t,j] &= \tanh\left(W^{h,R}[j,I]h[b,t-1,I] + W^{x,R}[j,K]x[b,t,K] - B^R[j]\right) \\
G^h[b,t,j] &= \sigma\left(W^{h,G}[j,I]h[b,t-1,I] + W^{x,G}[j,K]x[b,t,K] - B^G[j]\right) \\
h[b,t,j] &= G^h[b,t,j]h[b,t-1,j] + (1 - G^h[b,t,j])R[b,t,j]
\end{aligned}$$

Here I have written the gate as G^h to emphasize that it is a gate used to define a convex combination of the h_{t-1} and x_t inputs to h_t. Modify these equations to use a second gate G^R which gates inputs to R so that the input to the activation function (threshold function) producing $R[b,t,j]$ is a convex combination of

$$W^{h,R}[j,I]h[b,t-1,I] - B^{h,R}[j]$$

and

$$W^{x,R}[j,K]x[b,t,K] - B^{x,R}[j]$$

where the weighting in the convex combination is given by your new gate $G^R[b,t,j]$. This gated RNN is similar to, but different from, a GRU which also has two gates.

Solution:

$$
\begin{aligned}
G^R[b,t,j] &= \sigma\left(W^{h,GR}[j,I]h[b,t-1,I] + W^{x,GR}[j,K]x[b,t,K] - B^{GR}R[j]\right) \\
R[b,t,j] &= \tanh\left(\begin{array}{l} G^R[b,t,j]\ (W^{h,R}[j,I]h[b,t-1,I] - B^{h,R}[j]) \\ +\ (1-G^R[b,t,j])\ (W^{x,R}[j,K]x[b,t,K] - B^{x,R}[j]) \end{array}\right) \\
G^h[b,t,j] &= \sigma\left(W^{h,Gh}[j,I]h[b,t-1,I] + W^{x,Gh}[j,K]x[b,t,K] - B^{Gh}[j]\right) \\
h[b,t,j] &= G^h[b,t,j]h[b,t-1,j] + (1-G^h[b,t,j])R[b,t,j]
\end{aligned}
$$

Problem 4. RNN parallel run time. Consider an autoregressive RNN neural language model with $P_\Phi(w_{t+1}|w_1,\ldots,w_t)$ defined by

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

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

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

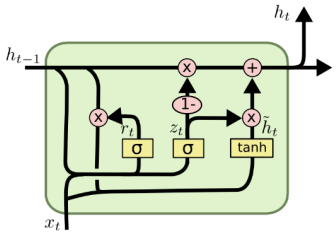
What is the order of run time as a function of sentence length T for the back-propagation for this model run on a sentence $w_1,\ldots,w_T$? Explain your answer.

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.

Problem 5. Translating diagrams into equations. A UGRNN cell for computing $h[b,t,J]$ from $h[b,t-1,J]$ and $x[b,t,J]$ can be written as

$$
\begin{aligned}
G[b,t,j] &= \sigma\left(W^{h,G}[j,I]h[b,t-1,I] + W^{x,G}[j,K]x[b,t,K] - B^{G}[j]\right) \\
R[b,t,j] &= \tanh\left(W^{h,R}[j,I]h[b,t-1,I] + W^{x,R}[j,K]x[b,t,K] - B^{R}[j]\right) \\
h[b,t,j] &= G[b,t,j]h[b,t-1,j] + (1 - G[b,t,j])R[b,t,j]
\end{aligned}
$$

Modify the above equations so that they correspond to the following diagram for a Gated Recurent Unit (GRU).



[Christopher Olah]

Solution:

$$
\begin{aligned}
G_1[b,t,j] &= \sigma\left(W^{h,G_1}[j,I]h[b,t-1,I] + W^{x,G_1}[j,K]x[b,t,K] - B^{G_1}[j]\right) \\
h'[b,t,j] &= G_1[b,t,j]h[b,t-1,j] \\
G_2[b,t,j] &= \sigma\left(W^{h,G_2}[j,I]h[b,t-1,I] + W^{x,G_2}[j,K]x[b,t,K] - B^{G_2}[j]\right) \\
R[b,t,j] &= \tanh\left(W^{h,R}[j,I]h'[b,t,I] + W^{x,R}[j,K]x[b,t,K] - B^{R}[j]\right) \\
h[b,t,j] &= (1 - G_2[b,t,j])h[b,t-1,j] + G_2[b,t,j]R[b,t,j]
\end{aligned}
$$