TTIC 31230 Fundamentals of Deep Learning, Autumn 2021

Exam 1

Problem 1: (25 pts). This problem is on softmax, entropy and cross entropy. Let the variable i range over the non-negative integers (0 to ∞). Consider the following softmax distribution where $\beta > 0$ is an inverse temperature parameter.

$$P(i) = \operatorname*{softmax}_i \,[-\beta i]$$

(a) Give an expression for $P(i)$ in terms of i without using a softmax or infinite sum. You can use the equality that for $0 < \lambda < 1$ we have

$$\sum_{n=0}^{\infty} \lambda^n = \frac{1}{1-\lambda}$$

Solution:

$$P(i) = \frac{e^{-\beta i}}{\sum_{i=0}^{\infty} e^{-\beta i}} = \frac{e^{-\beta i}}{1-e^{-\beta}}$$

(b) Solve for the entropy of this distribution. You can use the equality that for $0 < \lambda < 1$ we have

$$\sum_{n=0}^{\infty} n\lambda^n = \frac{\lambda}{(1-\lambda)^2}$$

Solution:

$$\begin{aligned}
H(P) &= E_i\;[-\ln P(i)] = \sum_{i=0}^{\infty} P(i)\;\left(-\;\ln \frac{e^{-\beta i}}{(1-e^{-\beta})}\right) \\
&= \sum_{i=0}^{\infty} P(i)\;\left(\ln\left(1-e^{-\beta}\right) + \beta i\right) \\
&= \left(\ln\left(1-e^{-\beta}\right)\sum_{i=0}^{\infty} P(i)\right) + \sum_i \frac{e^{-\beta i}}{1-e^{-\beta}}\;\beta i \\
&= \ln\left(1-e^{-\beta}\right) + \frac{\beta}{1-e^{-\beta}}\;\;\sum_{i=0}^{\infty} i(e^{-\beta})^i \\
&= \ln\left(1-e^{-\beta}\right) + \frac{\beta}{1-e^{-\beta}}\;\;\frac{e^{-\beta}}{(1-e^{-\beta})^2} \\
&= \ln\left(1-e^{-\beta}\right) + \frac{\beta e^{-\beta}}{(1-e^{-\beta})^3}
\end{aligned}$$

(c) Consider a one-hot population distribution defined by $\mathrm{Pop}(k) = 1$ and $\mathrm{Pop}(i) = 0$ for $i \neq k$. What is the cross-entropy $H(\mathrm{Pop}, P)$ and KL-divergence $KL(\mathrm{Pop}, P)$? What is the cross-entropy $H(P, \mathrm{Pop})$ and the KL divergence $KL(P, \mathrm{Pop})$?

Solution:

$$\begin{aligned}
H(\mathrm{Pop}, P) &= E_{i\sim\mathrm{Pop}}\left[-\ln P(i)\right] \\
&= -\ln P(k) \\
&= \frac{e^{-\beta k}}{1-e^{-\beta}} \\
\\
KL(\mathrm{Pop}, P) &= H(\mathrm{Pop}, P) - H(P) \\
&= \frac{e^{-\beta k}}{1-e^{-\beta}} - \left(\ln\left(1-e^{-\beta}\right) + \frac{\beta e^{-\beta}}{(1-e^{-\beta})^3}\right) \\
\\
H(P, \mathrm{Pop}) &= KL(P, \mathrm{Pop}) = \infty
\end{aligned}$$

Problem 2. (25 pts) 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 $h_\ell[I]$.

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

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. Backpropagation 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. (25 pts) 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. (25 pts) The self-attention in the transformer is computed by the following equations.

$$\begin{aligned}
\mathrm{Query}_{\ell+1}[k,t,i] &= W^Q_{\ell+1}[k,i,J]L_\ell[t,J] \\
\\
\mathrm{Key}_{\ell+1}[k,t,i] &= W^K_{\ell+1}[k,i,J]L_\ell[t,J] \\
\\
\alpha_{\ell+1}[k,t_1,t_2] &= \operatorname*{softmax}_{t_2}\left[\frac{1}{\sqrt{I}}\,\mathrm{Query}_{\ell+1}[k,t_1,I]\mathrm{Key}_{\ell+1}[k,t_2,I]\right]
\end{aligned}$$

Notice that here the shape of W^Q and W^K are both $[K,I,J]$. We typically have $I < J$ which makes the inner product in the last line an inner product of lower dimensional vectors.

(a) Give an equation computing a tensor $\tilde{W}^Q[K,J,J]$ computed from W^Q and W^K such that the attention $\alpha(k,t_1,t_2)$ can be written as

$$\alpha_{\ell+1}(k,t_1,t_2) = \operatorname*{softmax}_{t_2}\left[L_\ell[t_1,J_1]\tilde{W}^Q[k,J_1,J_2]L_\ell[t_2,J_2]\right]$$

For a fixed k we have that $W^Q[k,I,J]$ and $W^K[k,I,J]$ are matrices. We want a matrix $\tilde{W}^Q[k,J,J]$ such that the attention can be written in matrix notation as $h_1^\top \tilde{W}^Q h_2$ where h_1 and h_2 are vectors and $\tilde{W}^Q$ is a matrix. You need write this matrix $\tilde{W}^Q$ in terms of the matrices for W^Q and W^K. But write your final answer in Einstein notation with k as the first index.

Solution: This is easier to do in vector-matrix notation for a fixed k. But it can also be done entirely in Einstein notation:

$$\begin{aligned}
&= \operatorname*{softmax}_{t_2}\left[\frac{1}{\sqrt{I}}\,(L_\ell(t_1,J_1)W^Q[k,I,J_1])\;\;(W^K[k,I,J_2]L_\ell(t_1,J_2))\right] \\
&= \operatorname*{softmax}_{t_2}\left[\frac{1}{\sqrt{I}}\sum_{j_1,j_2,i} L_\ell(t_1,j_1)\;W^Q[k,i,j_1]\;W^K[k,i,j_2]\;L_\ell(t_1,j_2)\right] \\
&= \operatorname*{softmax}_{t_2}\left[\sum_{j_1,j_2} L_\ell(t_1,j_1)\;\left(\frac{1}{\sqrt{I}}\sum_i W^Q[k,i,j_1])\;\;(W^K[k,i,j_2])\right)\;L_\ell(t_1,j_2)\right] \\
&= \operatorname*{softmax}_{t_2}\left[\sum_{j_1,j_2} L_\ell(t_1,j_1)\;\tilde{W}^Q[k,j_1,j_2]\;L_\ell(t_1,j_2)\right] \\
&= \operatorname*{softmax}_{t_2}\left[L_\ell(t_1,J_1)\;\tilde{W}^Q[k,J_1,J_2]\;L_\ell(t_1,J_2)\right]
\end{aligned}$$

$$\tilde{W}^Q[k,j_1,j_2] \;=\; \frac{1}{\sqrt{I}}\,W^Q[k,I,j_1]W^k[k,I,j_2]$$

(b) Part (a) shows that we can replace the key and query matrix with a single query matrix without any loss of expressive power. If we eliminate the key matrix in this way what is the resulting number of query matrix parameters for a given layer and how does this compare to the number of key-query matrix parameters for a given layer in the original transformer version.

Solution: The original version uses $2(I \times J)$ key-query matrix parameters for each head. If we use only the single query matrix we use J^2 parameters for each head. These are the same for $I = J/2$.