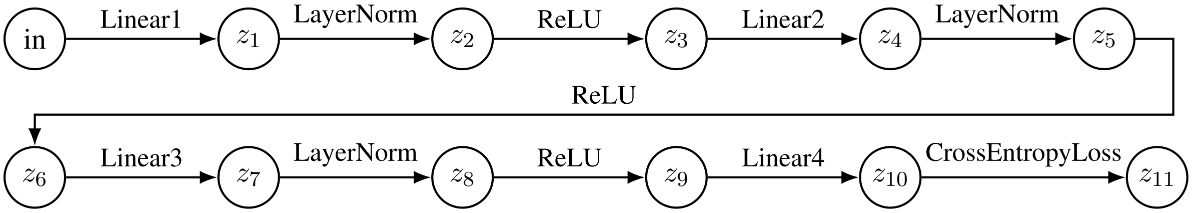
Data 188 Introduction to Deep Learning
Spring 2026 Eric Kim

Discussion 04

This discussion will cover normalization and regularization.

1. Model Size

Consider the following classification model architecture:



Suppose we know some of the dimensions of the model parameters as follows:

- `in.shape = [batchsize=4, in_feats=52]`
- `Linear1(in_feats=52, out_feats=_________)`
- `Linear2(in_feats=128, out_feats=256)`
- `Linear3(in_feats=_________, out_feats=20)`
- `Linear4(in_feats=_________, out_feats=_________)`
- `num_classes=1000`

(a) Fill in the missing dimensions above. Additionally, label the diagram above with the dimensions of the intermediate activations (z_1 through z_{10}) and output (z_{11}).

(b) Which term is called the "logits"? How do we transform logits to class probabilities?

(c) Let `batchsize=n` and let K be the total size of all intermediate activations. What is the total number of elements in all intermediate activations?

(d) Let C be the total number of model parameters. How would you compute this? (You don't need to do the arithmetic, but you should understand how to compute C on an exam.)

(e) Recall that GPU memory is a limited resource, and assume that all `ndarrays` (e.g. activations/parameters) are stored in GPU memory. When training (or inferencing) with large `batchsizes`, do activations or parameters take up more memory?

2. Layer Normalization

Recall that *layer normalization* (LayerNorm) is a normalization technique that normalizes across the feature dimension (e.g. the rows) of its input X, where X has shape `[batchsize, dim_feat]`:

$$\text{LayerNorm}(X) = \frac{X - \mu}{\sqrt{\sigma^2 + \epsilon}} \odot \gamma + \beta$$

where μ and σ^2 are the mean and variance of each **row** of X (shape `[batchsize, dim_feat]`), ϵ is a small constant to prevent division by zero, γ and β (both shape `[dim_feat]`) are learnable parameters that scale and shift the normalized output, and $\odot$ denotes element-wise multiplication.

(a) Perform LayerNorm(X) given the following (assume $\epsilon = 0$ for simplicity):

$$X = \begin{bmatrix} 4 & 4 & 8 & 8 \\ 0 & 2 & 2 & 8 \\ 1 & 1 & 1 & 5 \end{bmatrix}, \quad \gamma = [1, 2, 3, 4], \quad \beta = [0, -1, 1, 0]$$

(b) Notice that in the previous subpart, `batchsize = 3`. If we instead had `batchsize = 1`, can we still take the LayerNorm? Why or why not?

3. Batch Normalization

Recall that *batch normalization* (BatchNorm) is a normalization technique that normalizes across the batch dimension (e.g. the columns) of its input X, where X has shape `[batchsize, dim_feat]`:

$$\text{BatchNorm}(X) = \frac{X - \mu}{\sqrt{\sigma^2 + \epsilon}} \odot \gamma + \beta$$

where μ and σ^2 are the mean and variance of each **column** of X (shape `[batchsize, dim_feat]`), ϵ is a small constant to prevent division by zero, γ and β (both shape `[dim_feat]`) are learnable parameters that scale and shift the normalized output, and $\odot$ denotes element-wise multiplication.

Note that both the element-wise multiplication and sum with β broadcasts across the batch dimension.

(a) Perform BatchNorm(X) given the following (assume $\epsilon = 0$ for simplicity):

$$X = \begin{bmatrix} 2 & 2 & 1 & 1 \\ 2 & 5 & 4 & 1 \\ 5 & 5 & 10 & 10 \end{bmatrix}, \quad \gamma = [1, 2, 3, 4], \quad \beta = [0, -1, 1, 0]$$

(b) Notice that in the previous subpart, `batchsize = 3`. If we instead had `batchsize = 1`, can we still take the BatchNorm? Why or why not?

4. Dropout

Recall that *dropout* is a regularization technique where each input unit has a probability p of being set to zero ("dropped out") during training to prevent overfitting. There is also a variant of dropout called *dropout with correction* where we scale up the units that aren't zeroed out by $\frac{1}{1-p}$.

In this problem, we'll perform dropout on a toy example. Suppose we have an input vector $x = [2, 6, 4, 4, 5, 3, 4, 4]$, weight vector $w = [1, 1, 1, 1, 1, 1, 1, 1]$, output $y = w \cdot x$ (dot product), and dropout probability $p = 0.75$. Additionally, assume that dropout results in the first 6 values being dropped (set to 0).

(a) Compute y_{drop}, the output of regular dropout (without correction).

(b) Compute y_{corr}, the output of dropout with correction.

(c) Compute y_{orig}, the output without any dropout (i.e. the original output).

(d) Observe that $||y_{\text{corr}}||$ more closely matches $||y_{\text{orig}}||$ than $||y_{\text{drop}}||$ does. Why is this desirable?

(e) Dropout is an example of a layer that has different behavior at train vs. test time. What other layer also has different behavior for train vs. test time?

5. ℓ_2 Regularization

Recall that ℓ_2 regularization is a technique used to prevent overfitting by adding a penalty term to the loss function that encourages the model parameters to be small. The ℓ_2 regularization term is defined as:

$$\ell_2(\theta) = \lambda \cdot ||\theta||_2^2$$

where θ represents the model parameters, $||\theta||_2^2$ is the squared ℓ_2 norm of θ (i.e. the sum of the squares of the parameters), and λ is a hyperparameter that controls the strength of the regularization.

Consider the following optimization problem with ℓ_2 regularization where θ is the model parameters, ℓ_{ce} is cross-entropy loss, h is the hypothesis function (e.g. output logits of a neural network), X is the input data, and y is the true labels:

$$\arg\min_\theta \ell_{ce}(h(X), y) + \lambda \cdot ||\theta||_2^2$$

(a) When λ is very large, which of the following θ values are likely to be the result of the above optimization?

(a) $\theta = 0$
(b) $\theta = 1$
(c) $\theta = 1000000$
(d) $\theta = \theta^*$, where θ^* is the result of solving: $\arg\min_\theta \ell_{ce}(h(X), y)$

(b) When λ is very small (e.g. $\lambda = 0.0000001$), which of the following θ values are likely to be the result of the above optimization?

(a) $\theta = 0$
(b) $\theta = 1$
(c) $\theta = 1000000$
(d) $\theta = \theta^*$, where θ^* is the result of solving: $\arg\min_\theta \ell_{ce}(h(X), y)$

(c) Suppose we solve $\theta^* = \arg\min_\theta \ell_{ce}(h(X), y)$. Suppose we then add an ℓ_2 regularization term with a "moderate" value of λ such that $\theta = \arg\min_\theta \ell_{ce}(h(X), y) + \lambda \cdot ||\theta||_2^2$.

i. How would you expect $||\theta||_2$ to compare against $||\theta^*||_2$?

ii. How would you expect θ to compare against θ^* when comparing **training** dataset metrics (e.g. **classification accuracy**)?

iii. How would you expect θ to compare against θ^* when comparing **test** dataset metrics (e.g. **generalizability**)?