Problem Set 2, EE 150/CS 148a, Winter 2025

TAs: Andrew Zabelo and Armeet Jatyani

1. (10 points) Dropout

   (a) (5 points) Dropout Equivalence to ℓ_2 Regularization

   Show that OLS Regression with dropout is equivalent to Ridge Regression by proving that the expected squared loss with dropout is of the following form:

$$E[L(w)] = \|y - f(p)Xw\|^2 + g(p)\|\Gamma w\|^2$$

   where $f(p)$ and $g(p)$ are functions of p, and Γ is a diagonal matrix with the standard deviations of features in data matrix X.

Solution: The expected OLS loss is

$$\mathbb{E}[\mathcal{L}(w)] = ||y - \mathbb{E}[Xw]||^2 + \mathbb{E}[||\mathbb{E}[Xw] - Xw||^2].$$

In dropout, we zero out each weight randomly. Thus, the predictions of a OLS model with dropout can be written as

$$\hat{y} = XMw,$$

where M is a mask matrix of size $D \times D$ where D is the dimension of the weights. M is an indentity matrix with the i-th diagonal value being 0 if w_i is zeroed out for dropout. Thus, the expected loss for OLS with dropout is

$$\mathbb{E}[\mathcal{L}(w)] = ||y - \mathbb{E}[XMw]||^2 + \mathbb{E}[||\mathbb{E}[XMw] - XMw||^2].$$

We know $\mathbb{E}[M] = \frac{1}{1-p}I$, where I is a $D \times D$ identity matrix. This is because each value along the diagonal is an independent Bernoulli random variable with probability of p of being 0 and $1-p$ to be 1.
Computing the bias via linearity of expectation,

$$||y - \mathbb{E}[XMw]||^2 = ||y - X\mathbb{E}[M]w||^2 = ||y - (1-p)Xw||^2 .$$

To compute the variance, we first substitute the expected value of the prediction:

$$\sigma^2 = \mathbb{E}[||\mathbb{E}[XMw] - XMw||^2] = \mathbb{E}[||(1-p)Xw - XMw||^2].$$

Now, we expand the expression inside the outer expected value using its quadratic form:

$$\begin{aligned}
\sigma^2 &= ||(1-p)Xw - XMw||^2 \\
&= ((1-p)Xw)^T((1-p)Xw) - 2((1-p)Xw)^T(XMw) + (XMw)^T(XMw) \\
&= (1-p)^2 w^T X^T Xw - 2(1-p)w^T X^T XMw + w^T M^T X^T XMw.
\end{aligned}$$

Computing the expected value,

$$\begin{aligned}\sigma^2 &= \mathbb{E}\left[(1-p)^2 w^T X^T X w - 2(1-p) w^T X^T X M w + w^T M^T X^T X M w\right] \\ &= (1-p)^2 w^T X^T X w - 2(1-p) w^T X^T X \mathbb{E}[M] w + \mathbb{E}[w^T M^T X^T X M w] \\ &= (1-p)^2 w^T X^T X w - 2(1-p)^2 w^T X^T X w + (1-p) w^T X^T X w.\end{aligned}$$

In the above computation, the last expected value in the second line is simplifies as such because the expected value of the square of a Bernoulli r.v. is the expected value of just the regular Bernoulli r.v.
Since we know $\Gamma = \sqrt{X^T X}$, $||w^T X^T X w|| = ||\Gamma w||^2$. Substituting this in,

$$\sigma^2 = (1-p)^2 ||\Gamma w|^2 - 2(1-p)^2 ||\Gamma w|^2 + (1-p) ||\Gamma w|^2 = p(1-p) ||\Gamma w||^2.$$

(b) (5 points) Dropout Code Create a copy of the Dropout Colab Notebook and rename it to Dropout {first name last name}.ipynb. Follow the instructions to generate a Loss vs. Dropout Rate plot.

Solution: colab link

2. (10 points) Batch Normalization

(a) (5 points) BatchNorm Backpropagation Given $\frac{\partial L}{\partial Y}$, derive expressions for:

- $\frac{\partial L}{\partial \beta}$
- $\frac{\partial L}{\partial \gamma}$
- $\frac{\partial L}{\partial X}$

Solution:

$$Y_{ij} = \gamma_j \hat{X}_{ij} + \beta_j \implies Y_i = \gamma^T \odot \hat{X}_i + \beta \implies \frac{\partial Y_i}{\partial \beta^T} = 1 \implies \frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{N} \left(\frac{\partial \mathcal{L}}{\partial Y_i}\right)^T$$

$$Y_{ij} = \gamma_j \hat{X}_{ij} + \beta_j \implies Y_i = \gamma^T \odot \hat{X}_i + \beta \implies \frac{\partial Y_i}{\partial \gamma^T} = \hat{X}_i \implies \frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{N} \left(\frac{\partial \mathcal{L}}{\partial Y_i} \odot \hat{X}_i\right)^T$$

$$Y_{ij} = \gamma_j \hat{X}_{ij} + \beta_j \implies Y_i = \gamma^T \odot X_i + \beta \implies \frac{\partial Y_i}{\partial \hat{X}_i} = \gamma^T \implies \frac{\partial \mathcal{L}}{\partial \hat{X}} = \frac{\partial \mathcal{L}}{\partial Y} \odot \begin{bmatrix} \gamma^T \\ \vdots \\ \gamma^T \end{bmatrix}$$

Now, we know that $\hat{X}_{ij} = (X_{ij} - \mu_j)/\sqrt{\sigma_j^2 + \epsilon}$. Since μ_j, σ_j^2 are both functions of X_{ij}, we need to apply quotient rule and chain rule:

$$\frac{\partial \hat{X}_{ij}}{\partial X_{ij}} = \frac{s_j (X_{ij} - \mu_j)' - (X_{ij} - \mu_j) s_j'}{s_j^2}.$$

Here, s_j is $\sqrt{\sigma_j^2 + \epsilon}$. Computing the gradients, we have

$$
\begin{aligned}
\mu_j' &= \frac{1}{N} \\
s_j' &= \frac{1}{2\sqrt{\sigma_j^2+\epsilon}} \cdot \frac{\partial}{\partial X_{ij}}(\sigma_j^2+\epsilon) \\
&= \frac{1}{2s_j} \cdot \frac{\partial}{\partial X_{ij}} \cdot \frac{1}{N} \cdot \sum_{k=1}^{N}(X_{kj}-\mu_j)^2 \\
&= \frac{1}{2s_j} \cdot \frac{2}{N} \cdot \sum_{k=1}^{N}(X_{kj}-\mu_j) \cdot \frac{\partial}{\partial X_{ij}}(X_{kj}-\mu_j) \\
&= \frac{1}{2s_j} \cdot \frac{2}{N} \cdot \left(X_{ij}-\mu_j + \sum_{k=1}^{N}(X_{kj}-\mu_j) \cdot \left(-\frac{1}{N}\right)\right) \\
&= \frac{1}{Ns_j} \cdot (X_{ij}-\mu_j)\,.
\end{aligned}
$$

In the above calculation for s_j', the summation for $k = 1, \ldots, N$ cancels out because the sums of the difference with a mean is 0. The extra $X_{ij} - \mu_j$ term comes from the fact that when $k = i$, the partial derivative of $(X_{ij} - \mu_j)$ w.r.t. X_{ij} is $1 - \mu_j'$.

Thus, for $\hat{X}_{ij}$, the derivative is

$$
\begin{aligned}
\frac{\partial \hat{X}_{ij}}{\partial X_{ij}} &= \frac{s_j(1-\frac{1}{N}) - (X_{ij}-\mu_j) \cdot \frac{1}{Ns_j}(X_{ij}-\mu_j)}{s_j^2} \\
&= \frac{1}{s_j} - \frac{1+\hat{X}_{ij}^2}{Ns_j}.
\end{aligned}
$$

However, we still have to account for the role X_{ij} plays in $\hat{X}_{kj}, k \neq i$ since X_{ij} is used in μ_j, s_j for those terms as well.

$$
\begin{aligned}
\frac{\partial \hat{X}_{kj}}{\partial X_{ij}} &= \frac{s_j(X_{kj}-\mu_j)' - (X_{kj}-\mu_j)s_j'}{s_j^2} \\
&= \frac{s_j\left(-\frac{1}{N}\right) - (X_{kj}-\mu_j) \cdot \frac{1}{Ns_j}(X_{ij}-\mu_j)}{s_j^2} \\
&= -\frac{1+\hat{X}_{kj}\hat{X}_{ij}}{Ns_j}.
\end{aligned}
$$

Combining the derivatives from $\hat{X}_{kj}$ and $\hat{X}_{ij}$ with the values of $\frac{\partial \mathcal{L}}{\partial \hat{X}_{ij}}$,

$$
\frac{\partial \mathcal{L}}{\partial X_{ij}} = \frac{1}{s_j} \cdot \frac{\partial \mathcal{L}}{\partial \hat{X}_{ij}} - \frac{1}{Ns_j}\sum_{k=1}^{N}\frac{\partial \mathcal{L}}{\partial \hat{X}_{kj}} - \frac{\hat{X}_{ij}\sum_{k=1}^{N}\hat{X}_{kj} \cdot \frac{\partial \mathcal{L}}{\partial \hat{X}_{kj}}}{Ns_j}.
$$

Not sure if the TA reading this is looking for the OG Batchnorm paper derivation, so here's the comparision: essentially, in the sum above, the first term is the derivative that comes from the contribution of X_{ij} to every $\hat{X}_{ij}$, the second term is from the contribution of X_{ij} to every time the mean is used to compute an element in column j for $\hat{X}$, and the last term is the contribution of X_{ij} to every time the variance is used to compute an element in column j in $\hat{X}$. To put this into matrix form (as a product of a Jacobian of sorts with the matrix of $\frac{\partial L}{\partial \hat{X}}$, which we will now denote D), notice that the matrix corresponding to the first term is $\frac{1}{s_j} \odot (ID)$, where I is the identity matrix, the second term can be written as $-\frac{1}{Ns_j} \odot (\mathbf{1}D)$, where $\mathbf{1}$ is the matrix of all ones, and the last term corresponds to $\frac{1}{Ns_j} \odot \hat{X} \odot \Big(\mathbf{1} \cdot (\hat{X} \odot D)\Big)$. Thus, our gradient for the input is

$$\frac{\partial \mathcal{L}}{\partial X} = \frac{1}{s_j} \odot (I\frac{\partial \mathcal{L}}{\partial Y} \odot [\gamma^T]) - \frac{1}{Ns_j} \odot (\mathbf{1}\frac{\partial \mathcal{L}}{\partial Y} \odot [\gamma^T]) - \frac{1}{Ns_j} \odot \hat{X} \odot \left(\mathbf{1}(\hat{X} \odot \frac{\partial \mathcal{L}}{\partial Y} \odot [\gamma^T])\right).$$

In the equation above, $[\gamma^T]$ is the $N \times D$ matrix with its rows being γ^T.

(b) (5 points) Batch Normalization Code Implement BatchNorm from scratch and verify that your implementation matches PyTorch's built-in BatchNorm function.

Solution: colab link
Holy shit it actually works I didn't think my formula was right since it was different from the batchnorm paper's.

3. (15 points) Layer Normalization

(a) (5 points) LayerNorm Backpropagation Given $\frac{\partial L}{\partial Y}$, derive expressions for:

- $\frac{\partial L}{\partial \beta}$
- $\frac{\partial L}{\partial \gamma}$
- $\frac{\partial L}{\partial X}$

Solution: There's no change for β and γ since the dimensions for those are still the same.

$$Y_{ij} = \gamma_j \hat{X}_{ij} + \beta_j \implies Y_i = \gamma^T \odot \hat{X}_i + \beta \implies \frac{\partial Y_i}{\partial \beta^T} = 1 \implies \frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{N} \left(\frac{\partial \mathcal{L}}{\partial Y_i}\right)^T$$

$$Y_{ij} = \gamma_j \hat{X}_{ij} + \beta_j \implies Y_i = \gamma^T \odot \hat{X}_i + \beta \implies \frac{\partial Y_i}{\partial \gamma^T} = \hat{X}_i \implies \frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{N} \left(\frac{\partial \mathcal{L}}{\partial Y_i} \odot \hat{X}_i\right)^T$$

$$Y_{ij} = \gamma_j \hat{X}_{ij} + \beta_j \implies Y_i = \gamma^T \odot X_i + \beta \implies \frac{\partial Y_i}{\partial \hat{X}_i} = \gamma^T \implies \frac{\partial \mathcal{L}}{\partial \hat{X}} = \frac{\partial \mathcal{L}}{\partial Y} \odot \begin{bmatrix} \gamma^T \\ \vdots \\ \gamma^T \end{bmatrix}$$

As for X, since it's analagous to the transposed version of batchnorm, if we sum from $j = 1 \ldots D$, we should have the correct formulas for layernorm backprop. Notice that we also commute any matrix multiplications since we want it over the D dimension this time, not the N dimension.

$$\frac{\partial \mathcal{L}}{\partial X} = \frac{1}{s_i} \odot (\frac{\partial \mathcal{L}}{\partial Y} I \odot [\gamma^T]) - \frac{1}{Ns_i} \odot (\frac{\partial \mathcal{L}}{\partial Y}\mathbf{1} \odot [\gamma^T]) - \frac{1}{Ns_i} \odot \hat{X} \odot \left((\hat{X} \odot \frac{\partial \mathcal{L}}{\partial Y} \odot [\gamma^T])\mathbf{1}\right).$$

(b) (5 points) Layer Normalization Code Implement LayerNorm from scratch and verify correctness against PyTorch's built-in LayerNorm.

Solution: colab link

(c) (5 points) BatchNorm LayerNorm Puzzle Bob has access to Y from BatchNorm. How many elements in Z (LayerNorm output) does Bob know with certainty if X is symmetric?

Solution: All of them. If the matrix is symmetric, then the mean and variance across the batch and feature dimensions are the same. Thus, layernorm, which is analagous to doing batchnorm on the transpose of a matrix, is the same.

4. (40 points) Regularization, Optimizers, and Augmentation in MNIST

(a) (10 points) Regularization

1. Plot the distribution of weights of a newly initialized model. Let X be a random variable that is the value of a parameter chosen at random from our uninitialized model. Write an expression for the PDF of X. Why is it beneficial for the standard deviation of the initial distribution of a layer to decrease as layer width increases?
2. Plot the distribution of weights of a model trained using SGD. Qualitatively, what is the new distribution of weights?
3. Now plot the distribution of weights of a model trained using SGD and weight decay 0.01. Qualitatively, what changed about the distribution? In theory, why should this help a model's ability to generalize to unseen data?
4. ℓ_2 regularization on weights (not to be confused with ℓ_2 regularization on model outputs) and weight decay are often used interchangeably, but they are in fact distinct methods. Prove that for standard SGD, ℓ_2 regularization and weight decay are equivalent.
   Weight decay update:
   $$w_{t+1} \leftarrow w_t - (\nabla_w L + \lambda_{\text{weight decay}} w_t)$$
   ℓ_2 regularized update:
   $$w_{t+1} \leftarrow w_t - (\nabla_w L)$$
   where
   $$L = \lambda_{\ell_2} \sum_i w_i^2$$
5. Use your solution from part (d) to run SGD with ℓ_2 regularization with the ℓ_2 lambda that would make it equivalent to SGD with weight decay 0.01. How does the distribution of weights compare to part (c)?

Solution: colab

The distribution of new weights is the sum of multiple uniform distributions. The pdf is roughly

$$p(x) \approx \begin{cases} 0.03 & \text{if } x \in (-0.5, 0.5) \\ 0.14 & \text{if } x \in [-0.1, -0.05] \cup [0.05, 0.1] \end{cases}.$$

It makes sense for the the standard deviation of the initial distribution of a layer to decrease as layer width increases to make sure the variance of the proceeding layer has a similar variance.

The distribution of weights trained with SGD is normal. In theory, this makes sense since naturally occuring data generally follows a uniform distribution.

ℓ_2 regularization and and weight decay are the same because the gradient of the sum of the squares of the weights is just the current weights times 2. Thus, the $\lambda_{\ell_2} = \frac{1}{2}\lambda_{\text{weight decay}}$.

(b) (15 points) Optimizers:

1. Derive the bias correction term $\frac{1}{1-\beta^t}$ for the Adam optimizer and show that it is necessary to un-bias the estimates for the first and second moments.
2. Explain why adding an ℓ_2 regularization penalty to the loss is problematic when using SGD with momentum. Discuss its effect when using RMSProp and Adam optimizers.
3. Explain how AdamW correctly penalizes weights in contrast with Adam.
4. Run the following and report the validation accuracies. Which two methods perform best? Why do the others perform worse?
   - Adam, default
   - Adam, ℓ_2 lambda=0.005
   - Adam, weight decay=0.01
   - AdamW, ℓ_2 lambda=0.005
   - AdamW, weight decay=0.01

Solution: The formula for the moment are

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 m_{t-1} + (1-\beta_2) g_t^2 \\ &. \end{aligned}$$

Computing the expected value of the moments,

$$\begin{aligned} \mathbb{E}[m_t] &= \mathbb{E}[\beta_1 m_{t-1} + (1-\beta_1) g_t] \\ \mathbb{E}[v_t] &= \beta_2 m_{t-1} + (1-\beta_2) g_t^2 \\ &. \end{aligned}$$

As the model becomes more stable and $g_i \to \mathbb{E}[g]$, the Adam update becomes
F

Using Adam with no weight decay or regularization worked best, but using AdamW with weight decay worked the best since AdamW accounts for momentum when using weight decay. AdamW does this by performing the weight decay updated and the momentum update separately, not by backpropagating the weight decay term as a gradient of the loss. Thus, the gradient from the L2 loss is not factored in when it computes the momentum.

(c) (15 points) Data Augmentation:

1. Implement methods for common image transformations: rotations, translations, blurring, scaling, shear, and Gaussian noise addition.
2. Display the effects of each augmentation using one example for each digit (0-9). Ensure augmented images remain recognizable.
3. Evaluate augmentation effects on performance:
   - Train a model on the small dataset and note validation accuracy.
   - Apply different augmentation strengths and train models to report:
     - Augmentation that decreases accuracy by $\leq 1\%$.
     - Augmentation that decreases accuracy by $1-3\%$.
     - Augmentation that decreases accuracy by $\geq 3\%$.

4. Investigate rotation range impact by setting $(-180, 180)$ and analyze misclassifications via a confusion matrix.
5. Develop a method to randomly apply augmentations to scale the dataset to 50,000 images. Train models and report validation accuracy for:
   - Small dataset (5,000 images).
   - Small dataset copied 10 times (50,000 images).
   - Augmented dataset (50,000 images).
   - Original dataset (50,000 images).

Solution: colab

5. (25 points) Word2Vec

(a) (10 points) Implement CBOW Implement and train the Continuous Bag of Words (CBOW) model on the Shakespeare dataset.

Solution: colab link

(b) (15 points) Semantically Similar Clusters Find clusters of words that exhibit semantic similarity based on trained embeddings.

Solution:

- **my** (1.0), **his** (0.34), **your** (0.33), **thy** (0.31), **lawrence'** (0.28), **our** (0.27), **the** (0.26), **sicilian** (0.26), **hastings'** (0.24), **her** (0.23)
- **brother** (1.0), **nunnery** (0.26), **son** (0.24), **sheepshearing** (0.22), **brother's** (0.21), **crutch** (0.21), **sons** (0.21), **beck** (0.2), **cow** (0.2), **daughter** (0.2)
- **slay** (1.0), **kill** (0.27), **submit** (0.27), **expose** (0.25), **railed** (0.25), **fade** (0.25), **practise** (0.25), **dardanius** (0.24), **guide** (0.24), **according** (0.24)
- **shall** (1.0), **will** (0.28), **blithe** (0.26), **longed** (0.26), **should** (0.25), **if't** (0.23), **may** (0.23), **can** (0.22), **would** (0.22), **tumbled** (0.22)
- **people** (1.0), **ground** (0.28), **forum** (0.26), **members** (0.24), **multitudes** (0.23), **ambassadors** (0.22), **wing** (0.22), **disciplines** (0.22), **lights** (0.21), **complaint** (0.21)
- **senator** (1.0), **bout** (0.26), **fisherman** (0.25), **soldier** (0.25), **citizen** (0.25), **witch** (0.24), **watch** (0.22), **singly** (0.21), **gentleman** (0.21), **exploits** (0.21)
- **dishonest** (1.0), **wiser** (0.29), **wight** (0.29), **wellfavoured** (0.29), **recount** (0.28), **drench** (0.27), **respite** (0.27), **ides** (0.26), **sentry** (0.25), **cutpurse** (0.25)
- **man** (1.0), **man"** (0.29), **squire** (0.26), **carcass** (0.26), **snail** (0.25), **pig** (0.24), **sincerity** (0.24), **breeder** (0.24), **ducat** (0.24), **woman** (0.23)
- **horse** (1.0), **mare** (0.27), **crutch** (0.25), **club** (0.23), **sword** (0.23), **beaver** (0.22), **bounties** (0.22), **tribe** (0.21), **reference** (0.21), **team** (0.21)
- **prevail** (1.0), **attain** (0.29), **detain** (0.29), **peruse** (0.29), **meddle** (0.28), **mix'd** (0.27), **oaks** (0.26), **o'ertake** (0.26), **befall** (0.26), **surrender** (0.25)