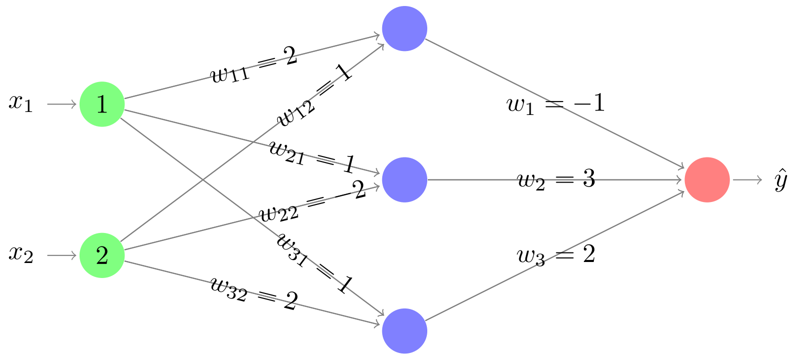
Tutorial Questions | Week 4

COSC2779 - Deep Learning

This tutorial is aimed at reviewing optimization techniques and regularization in deep learning. Please try the questions before you join the session.

1. Calculate the gradient update for the neural network below with given initialized weights, if the input data point is $\mathbf{x} = [1, 2]$ and the expected output is 3. The loss is mean squared error, hidden units have ReLU activation, linier activation in output unit and the learning rate is 0.1. Initial biases are all zero.



> **Solution:**
>
> Can compute the derivatives using the back prop rule.
>
> Output neuron: $\frac{\partial \mathcal{L}}{\partial w_1} = 24$; $\frac{\partial \mathcal{L}}{\partial w_2} = 0$; $\frac{\partial \mathcal{L}}{\partial w_3} = 30$; $\frac{\partial \mathcal{L}}{\partial b_o} = 6$;
>
> Hidden neuron 1: $\frac{\partial \mathcal{L}}{\partial w_{11}} = -6$; $\frac{\partial \mathcal{L}}{\partial w_{12}} = -12$; $\frac{\partial \mathcal{L}}{\partial b_1} = -6$;
>
> Hidden neuron 2: $\frac{\partial \mathcal{L}}{\partial w_{21}} = 0$; $\frac{\partial \mathcal{L}}{\partial w_{22}} = 0$; $\frac{\partial \mathcal{L}}{\partial b_2} = 0$;
>
> Hidden neuron 3: $\frac{\partial \mathcal{L}}{\partial w_{31}} = 12$; $\frac{\partial \mathcal{L}}{\partial w_{32}} = 24$; $\frac{\partial \mathcal{L}}{\partial b_3} = 12$;
>
> Update with: $w_j \leftarrow w_j + 0.1 \frac{\partial \mathcal{L}}{\partial w_j}$

2. What problem(s) will result from using a learning rate that's too

   (a) high?

   > **Solution:** Cost function does not converge to an optimal solution and can even diverge. To detect, look at the costs after each iteration (plot the cost function v.s. the number of iterations). If the cost oscillates wildly, the learning rate is too high. For batch gradient descent, if the cost increases, the learning rate is too high.

   (b) low?

   > **Solution:** Cost function may not converge to an optimal solution, or will converge after a very long time. To detect, look at the costs after each iteration (plot the cost function v.s. the number of iterations). The cost function decreases very slowly (almost linearly). You could also try higher learning rates to see if the performance improves.

3. What is a saddle point? What is the advantage/disadvantage of Stochastic Gradient Descent in dealing with saddle points?

Solution: Saddle point - The gradient is zero, but it is neither a local minima nor a local maxima. Also accepted - the gradient is zero and the function has a local maximum in one direction, but a local minimum in another direction. SGD has noisier updates and can help escape from a saddle point

4. One of the difficulties with the sigmoid activation function is that of saturated units. Briefly explain the problem, and whether switching to tanh fixes the problem

Solution: The derivative of $\sigma(z)$ is small for large negative or positive z.

No, switching to tanh does not fix the problem. The same problem persists in $\tanh(z)$. Both function has a sigmoidal shape. We can see tanh is effectively a scaled and translated sigmoid function: $\tanh(z) = 2\sigma(2z) - 1$

tanh activations are centered around zero, whereas sigmoid are centered around 0.5. Centering the data/hidden activation can help optimization due to similar effect to the batch normalization without the variance division.

5. What happens if we use batch-norm after an affine layer with bias? what effect does increasing the bias has?

Solution:

$$\mathbf{z}^{(i)} = \mathbf{W}^{(l)}\mathbf{x}^{(i)} + \mathbf{b}^{(l)} \quad \triangleright \text{Affine transform}$$

$$\mu_j = \frac{1}{m}\sum_i z_J^{(i)}; \quad \sigma_j^2 = \frac{1}{m}\sum_i \left(z_j^{(i)} - \mu_j\right)^2$$

$$\hat{z}_j^{(i)} = \frac{z_j^{(i)} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} \quad \triangleright \text{mean zero, variance 1}$$

$$\tilde{z}_j^{(i)} = \gamma \hat{z}_j^{(i)} + \beta \quad \triangleright \gamma, \beta \text{ learnable parameters}$$

$\text{mean}(\mathbf{z}) = \text{mean}(\mathbf{W}^{(l)}\mathbf{x}) + \mathbf{b}^{(l)}$

$$\hat{z}^{(i)} = \frac{\mathbf{W}^{(l)}\mathbf{x}^{(i)} + \mathbf{b}^{(l)} - \text{mean}(\mathbf{W}^{(l)}\mathbf{x}) - \mathbf{b}^{(l)}}{\sqrt{\sigma_j^2 + \epsilon}} \quad \triangleright \text{mean zero, variance 1}$$

Bias will be removed by the normalization step of batch norm. Therefore it has no affect.

6. Explain why dropout in a neural network acts as a regularizer.

Solution: There are several explanations:

- Dropout is a form of model averaging. In particular, for a layer of H nodes, we sampling from 2H architectures, where we choose an arbitrary subset of the nodes to remain active. The weights learned are shared across all these models means that the various models are regularizing the other models.
- Dropout helps prevent feature co-adaptation, which has a regularizing effect.
- Dropout adds noise to the learning process, and training with noise in general has a regularizing effect.
- Dropout leads to more sparsity in the hidden units, which has a regularizing effect ("shrinking the weights" or "spreading out the weights").

7. Why do we often refer to L2-regularization as "weight decay"? Derive a mathematical expression to explain your point.

Solution: In the case of L2 regularization, we can derive the following update rule for the weights:

$$w \leftarrow (1 - \alpha\lambda)w - \frac{\partial \mathcal{L}}{\partial w}$$

where α is the learning rate and λ is the regularization hyperparameter ($\alpha\lambda << 1$). This shows that at every iteration W's value is pushed closer to zero.

8. Explain what effect will the following operations generally have on the bias and variance of your model.

   (a) Regularizing the weights

   Solution: bias: Increases, variance: Decreases

   (b) Increasing the width of the layers

   Solution: bias: Decreases, variance: Increases

   (c) Using dropout to train a deep neural network

   Solution: bias: Increases, variance: Decreases

   (d) Getting more training data

   Solution: bias: No change, variance: Decreases