VANDERBILT UNIVERSITY

Homework 2

CS 8395 - Special Topics in Deep Learning

Due Tuesday, March 1st, Midnight

1 Error Decomposition [25]

Let $\mathcal{R}$ denote the population risk, $\hat{\mathcal{R}}$ denote the empirical risk, $\mathcal{F}$ denote our hypothesis class (i.e., our function space), and $\mathcal{F}_\delta$ denote a complexity constrained hypothesis class (e.g., norm regularized function space). Consider an arbitrary hypothesis $\hat{f} \in \mathcal{F}_\delta$ for some $\delta > 0$.

1. Define the population error for hypothesis $\hat{f}$
2. Prove that the population error can be decomposed into "optimization error," "approximation error," and "statistical error." Be sure to prove **EVERY** step of your derivations.
3. State the meaning of each of these errors? Explicitly, what causes an increase/decrease in these errors?
4. When we are using gradient descent to find the solution to an overparameterized neural network, which of these terms will be affected by GD?
5. We use data augmentation to reduce which of these terms?

2 Group Equivariance and Group Invariance [25]

In class we looked at the definition of functions that are invariant or equivariant with respect to a symmetry group. Let Ω define our domain, and $\mathcal{X}(\Omega)$ denote the set of signals that are defined on the domain, e.g., $x \in \mathcal{X} : \Omega \to \mathbb{R}$. Let $\mathcal{G}$ be a symmetry group, where $g \in \mathcal{G} : \Omega \to \Omega$, and let $\rho(g)$ represent the group presentation, where $\rho(g)x = x(g^{-1})$.

1. Define $\mathcal{G}$-Invariance
2. Define $\mathcal{G}$-Equivariance
3. Let $h : \mathcal{X} \to \mathcal{X}$ be $\mathcal{G}$-Invariant and $f : \mathcal{X} \to \mathcal{X}$ be $\mathcal{G}$-Equivariant then:
   - Is the composition $h \circ f$ invariant, equivariant, or neither? Prove it.
   - Is the composition $f \circ h$ invariant, equivariant, or neither? Prove it.

3 Deep Learning Blue Print [50]

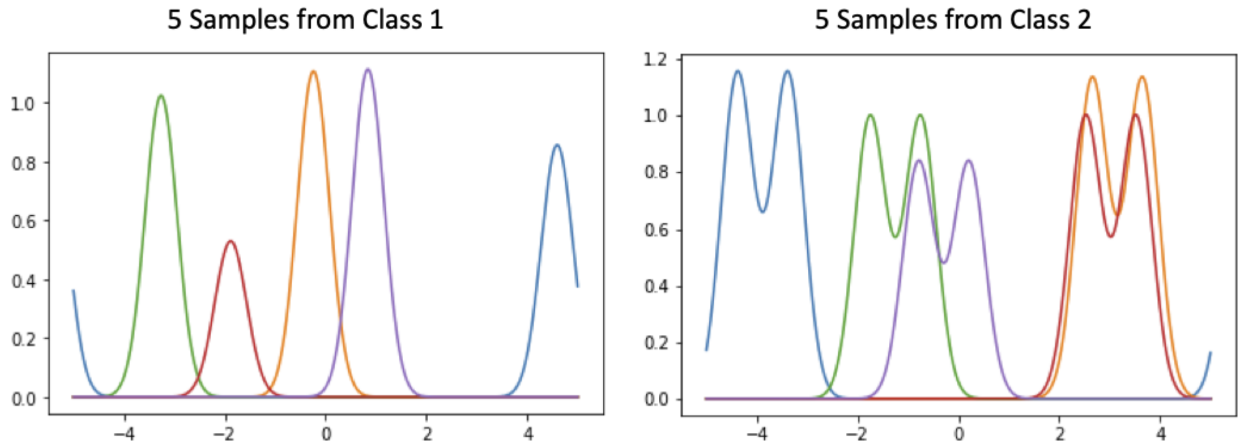

Figure 1: Classes 1 and 2 that are generated as circular shifts of one and two Gaussians.

In this problem, we consider a binary classification of two classes of signals defined on $\Omega = \mathbb{Z}^d$ where $d = 1000$. The two classes are generated via shifts of one (for class one) and two (for class two) bumps (i.e., Gaussians). Figure 1 shows samples from the two classes. We only have few training samples 25 per class. You can find the train and test samples in 'hw2_p3.pkl.'

1. Train a multi-layer perceptron to classify the two classes. Use the adam optimizer, and train the network for multiple epochs. Your architecture should lead to 100% training accuracy (i.e., interpolation regime!). Calculate training and test accuracy per epoch and plot them. Please ensure that you have x_label, y_label, and legends for your plots. How many parameters does your network have? Does it generalize to the test data?
2. Repeat the above experiment, using your model, but this time use data augmentation. In each iteration, and for each training sample, you will need to flip a coin, and with probability 0.5 you will need to augment your sample via a random circular shift. Use $torch.roll(x, shift)$ for this purpose. Plot your train and test accuracies for the data-augmented training. Does your model generalize to test data? What are we enforcing on our network when we perform this data augmentation?

3. Design a one-dimensional convolutional neural network, which has one layer of a translation equivariant function, followed by nonlinearity, and a translation invariant function. Then send the output of this network to a linear classifier (i.e., use *torch.nn.Linear*) to perform translation invariant classification. For your translation equivariant function use:

   $$torch.Conv1d(1, 16, kernel\_size = ksize, padding\_mode = `circular', padding = ksize//2)$$

   with 16 convolutional kernels, and kernel size of $ksize = 25$, followed by a nonlinearity (e.g., ReLU). For your translation invariant function use global average pooling on d. Use Adam optimizer and repeat the classification experiment for this convolutional network. How many parameters does your network have? Does it generalize to the test data?