Anthony Poerio (adp59@pitt.edu)

CS1674: Homework 10 - Written

Due: 11/28/2016, 11:59pm

QUESTIONS

1) **Say a vision system predicts the following scores for a cat image: 10 for the category "cat," 5 for the category "dog," and 3 for the category "cow." Another system predicts scores 8, 6 and 1 for the same categories. Is the SVM loss the same or different for these two systems? What about the softmax loss? Why?**

   SVM Loss (assuming we use the loss function on slides 41-45 in the neural net slide set):

   - System 1 calculation
     - Max(0, 5-10 +1) + Max(0, 3-10 +1) = 0
   - System 2 Calculation
     - Max(0, 6-8 +1) + Max(0, 1-8 +1) = 0
       - Therefore, SVM Loss is *same* for both systems, **because we are taking a max with 0,** and therefore the lowest possible value is 0.
       - If we allowed for a negative value, however → then the SVM Losses would be different.

   Softmax Loss:

   - Softmax Loss would **very likely** be *different* for both systems, because its output is **an exact probability,** as a Real Number in the range [0,1). Moreover, we aren't using a max, in this function. So, there's no operation by which we would have an 'artificial cutoff' of sorts, like we do when taking the max calculations of the SVM Loss function.

2) **Briefly, how does gradient descent work?**

   Essentially, gradient descent works by iteratively looking for the vector with the steepest downward slope from any given point in some arbitrary high-dimensional space.

Then, we take a 'step' of some specified distance in the direction of that steepest downward slope, and repeat.

We repeat this until we find what is likely the 'bottom' or minimum point in that space. Often it is useful to find minimizing functions/values/vectors/distances (i.e. – for matching), and this is an efficient way to do so.

3) **What is mini-batch?**

Mini batch is a gradient descent algorithm that works ***by only taking a subset of the entire data-set into account at each step.***

Practically, this often yields similar results to a full gradient descent algorithm where ***all*** values are exhaustively checked, and it can be executed much more quickly.

4) **How can we prevent overfitting in a neural network?**

Speaking generally, overfitting in a neural network is the **result of two things**:

1. **Running too many epochs** on the training data
   - When this happens we fit *too tightly* against the training data itself, and can't generalize effectively
2. **Having too many hidden units** in our model
   - In this case, we are essentially creating a very high-degree polynomial function as our model, which—again—can't generalize well, because it is fit too tightly against our training data. And hence it will match those training data points alone, at the expense of all others.

5) **What do higher vs lower convolutional layers in a convolutional neural network capture?**

Higher convolutional layers capture more abstract features → (i.e. – whether the pixels we are examining show an image of a cat, or a dog, or fish, or a cow.)

Lower convolutional layers capture less abstract features → (i.e. – are we looking at an edge, or a corner?)

6) **Why is not necessary to have a massive amount of data to use convolutional neural networks?**

Abstractly – the reason is **transfer learning.**

When we have **very similar data** as a previous, trained CNN—even if we have ***very little*** of it—it's possible to use a linear classifier on the top (or "highest") layer of the neural network, and still achieve reasonable performance.