CS 189: Introduction to Machine Learning - Discussion 13

1. Kernel k-means

   Suppose we have a dataset $\{x_i\}_{i=1}^N, x_i \in \mathbb{R}^n$ that we want to split into K clusters. Furthermore, suppose we know a priori that this data is best clustered in a large feature space $\mathbb{R}^m$, and that we have a feature map $\phi : \mathbb{R}^n \to \mathbb{R}^m$. How should we perform clustering in this space?

   (a) Write the objective for K-means clustering in the feature space (using the squared L_2 norm in the feature space). Do so by explicitly constructing cluster centers $\{\mu_k\}_{k=1}^K$ with all $\mu_k \in \mathbb{R}^m$.

   (b) Write an algorithm that minimizes the objective in (a).

   (c) Write an algorithm that minimizes the objective in (a) without explicitly constructing the cluster centers $\{\mu_k\}$. Assume you are given a kernel function $\kappa(x, y) = \phi(x) \cdot \phi(y)$.

2. Parameters in Fully Connected Neural Networks

   (a) The neural network you built for Homework 6 had 784, 200, and 10 units in each layer. Determine the total number of parameters in this neural network. Be sure to include bias terms.

   (b) You would like to classify larger grayscale images (size 224×224). You have 1000 output classes and 3 hidden layers with 6000 units each. Approximately how many parameters are in this network?

3. Convolutional Neural Networks

A typical ConvNet has a series of convolutional and pooling layers, followed by some fully connected layers and an output layer. You already know how to implement fully connected layers. This question will give you some intuition about convolutional and pooling layers.

(a) A typical **convolutional layer** might consist of 128 3×3 filters. During the forward pass, we slide each filter across the width and height of the input layer and compute the dot product between the filter and the part of the image "underneath" the filter. So if our input image is 200×200 grayscale, we would end up with a 200×200 activation map for each of our 128 filters. Let's try it on the following example. The image is zero-padded for convenience; your output should be 3×3.

Input image: $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

Filter: $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

(b) **Pooling layers** are usually inserted between convolutional layers to reduce the dimensions of the image. This helps speed up computation of the network and increases the receptive field. A commonly used pooling layer is max-pooling with kernel size 2 and stride 2, which means that the max is taken over every block of 4 pixels. Try performing max-pooling on the following input:

Input image: $\begin{bmatrix} 2 & 9 & 3 & 8 \\ 1 & 2 & 4 & 8 \\ 2 & 4 & 5 & 1 \\ 0 & 3 & 2 & 6 \end{bmatrix}$

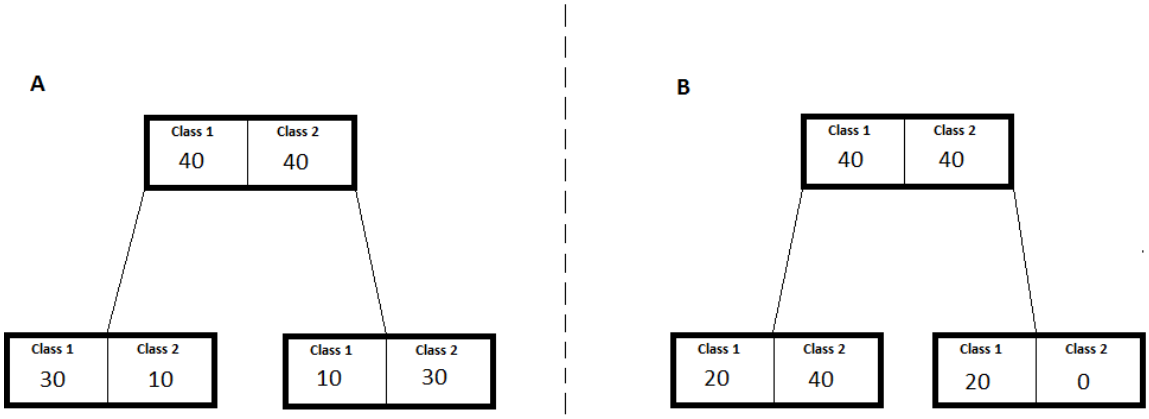
(c) **Receptive fields** The receptive field refers to how much of the original image a layer "sees." The very first conv layer with kernel size 3×3 has a receptive field of 3×3. What is the receptive field of a second 3×3 conv layer that immediately follows the first conv layer? What is the receptive field of a second 3×3 conv layer if a 2×2, stride 2 max-pooling layer is inserted in between?

(d) **Layer design** Would you prefer to stack 3 convolutional layers with kernel size 3×3, or use 1 convolutional layer with kernel size 7×7? We use ReLUs after each convolutional layer.

(e) **Calculating Parameters** Your ConvNet architecture is INPUT-(CONV-POOL)*5-FC-FC-OUTPUT. Input images are grayscale, 224×224. All CONV layers use 128 3×3 filters, with 1 pixel per side padding. All POOL layers are 2×2 stride 2 max-pooling. Both FC layers have 4096 units. Approximately how many parameters are in this network?

4. Decision Tree Review

(a) Let's say we are trying to build a 2-class decision tree and are currently at a node with 40 samples from each class. Our two possible splits are illustrated below. If we are trying to minimize misclassification error, which split should we select? What if we are trying to minimize entropy?



(b) An optimal decision tree is one that gives the most accurate classification with the least depth. Does using entropy as a heuristic always guarantee you will have an optimal decision tree?