# 9 Kernel Support Vector Machines/Artificial Neural networks

## Kernel Support Vector Machines

A support vector machine with Kernel function $K(x, \hat{x})$ is equivalent to a linear support vector machine on the dataset $\{V(x) : x \text{ in the dataset}\}$ where

$$K(x, \hat{x}) = V(x) \cdot V(\hat{x}).$$

Common Kernel functions are

- The linear kernel $K(x, \hat{x}) = x \cdot \hat{x}$
- The polnomial kernel $K(x, \hat{x}) = (1 + x \cdot \hat{x})^p$,$p = 2, 3, \ldots$
- The radial basis function kernel $K(x, \hat{x}) = exp(-\gamma|x - \hat{x}|)$

## Neural Networks

Given a nonlinear function such as

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

A Neural network for classifying points in $R^{n_1}$ into $n_5$ classes can take the form

$$x_1 = input \in \mathbb{R}^{1 \times n_1}$$

$$x_2 = \sigma(x_1 W_{12} + B_2), \qquad W \in \mathbb{R}^{n_1 \times n_2}, \qquad B \in \mathbb{R}^{1 \times n_2}.$$

$$x_3 = \sigma(x_2 W_{23} + B_3), \qquad W \in \mathbb{R}^{n_2 \times n_3}, \qquad B \in \mathbb{R}^{1 \times n_3}.$$

$$x_4 = \sigma(x_3 W_{34} + B_4), \qquad W \in \mathbb{R}^{n_3 \times n_4}, \qquad B \in \mathbb{R}^{1 \times n_4}.$$

$$x_5 = \text{softmax}(x_4 W_{45} + B_5), \qquad W \in \mathbb{R}^{n_4 \times n_5}, \qquad B \in \mathbb{R}^{1 \times n_5},$$

where the softmax function is defined

$$\text{softmax}(t_1, \ldots t_n) = \left( \frac{\exp(t_1)}{\exp(t_1) + \cdots + \exp(t_n)}, \frac{\exp(t_2)}{\exp(t_1) + \cdots + \exp(t_n)}, \ldots, \frac{\exp(t_n)}{\exp(t_1) + \cdots + \exp(t_n)} \right)$$

The network can be trained by gradient descent with respect to a cost function such as negative log likelihood. Letting $y \in \mathbb{R}^{1 \times n_5}$ denote the true label (one-hot encoded) corresponding to $x_1$,

$$NLL(x_1, y) = -y \log x_5.$$

Let $f$ denote the mean negative log-likelihood over the whole training set. Gradient descent in parameter space $p$ is

$$p_{n+1} = p_n - \alpha \text{grad } f(p_n).$$

There are various ways of improving the training, such as

- Mini-batch gradient descent: calculate an approximation to grad $f(p_n)$ using only a small subset of the data
- Momentum: instead of gradient descent, use the update rules

  $$\text{momentum}_{n+1} = 0.9\text{momentum}_n + 0.1\text{grad } f(p_n)$$

  $$p_{n+1} = p_n - \alpha \text{momentum}_{n+1}.$$

  This is a smoothed form of gradient descent, suitable for use with mini-batch gradient descent.
- Rectified-Linear Units: Replace the function $\sigma$ with the positive part function:

  $$\text{ReLu}(x) = \begin{cases} x & x > 0 \\ 0 & x < 0 \end{cases}$$
- Dropout: See `arxiv.org/pdf/1207.0580`

## Questions

Download dataset mnist.small.rdata and cifar10-small.rdata from the website. There are also files lab9svm.r and lab9ann.R.

1. Adapt the commands in lab9svm.r to get the best test accuracy for mnist.small.rdata and cifar10-small.rdata using support vector machines and appropriately chosen kernels (i.e. use trial and error and validation).
2. Adapt lab9ann.r to get the best test accuracy for mnist.small.rdata and cifar10-small.rdata.