Machine Learning
Hilary Term 2016
Week 7

Problem Sheet 5

1 Support Vector Machines

1.1 SVM Formulation

Let us look at support vector machines without kernels and assume that the data is *linearly separable*. In order to maximize the margin, a more natural formulation would be as follows: Fix $\|\mathbf{w}\|_2 = 1$, so the distance of $\mathbf{x}$ from hyperplane defined by $(\mathbf{w}, w_0)$ is exactly $|\mathbf{x} \cdot \mathbf{w} + w_0|$. Then, we can define a mathematical program:

$$\begin{aligned} \text{maximize} \quad & \alpha \\ \text{subject to} \quad & y_i(\mathbf{x}_i \cdot \mathbf{w} + w_0) \geq \alpha \text{ for } i = 1, \dots, m \\ & \|\mathbf{w}\| = 1 \end{aligned}$$

Unfortunately, the condition $\|\mathbf{w}\| = 1$ does not result in a convex set. Argue that relaxing the constraint to be $\|\mathbf{w}\| \leq 1$ does not change the optimal solution of the above program. Then show that this formulation is equivalent to the one we considered in class, *i.e.,* show that an optimal solution for one can be used to obtain an optimal solution for the other.

1.2 Simple Observations

1. Suppose we use the SVM formulation for separable data, and that the data indeed is linearly separable. Recall that in this case, support vectors are those points $\mathbf{x}_i$ in the dataset those for which $y_i(\mathbf{w}^* \cdot \mathbf{x}_i + w_0^*) = 1$, where $\mathbf{w}^*, w_0^*$ is the max-margin hyperplane. If your dataset consists of m points in n dimensional space, what is the *maximum* number of suppor vectors possible? the minimum?

2. Suppose you use the formulation for the non-separable case, *i.e.,* with the slack variables ζ_i, but your data is actually linearly separable. Do you recover the "true" max-margin separating hyperplane?

2 Neural Networks and Back Propagation

2.1 LogSoftMax and ClassNLL

Let us derive the backpropagation equations for the last (output) layer. Let L be the last layer of the neural network. Let $\mathbf{z}^L$ be the input to the *non-linearity* at layer L. (Let's say $\mathbf{z}^L = \mathbf{W}^L \mathbf{a}^{L-1} + \mathbf{b}^L$, though in this question we are not concerned with how $\mathbf{z}^L$ was computed.)

Let $\mathbf{a}^L$ the output of the last layer, where $\mathbf{a}^L = \text{LogSoftMax}(\mathbf{z}^L)$. To be precise, if $\mathbf{z}^L \in \mathbb{R}^k$, then $\mathbf{a}^L \in \mathbb{R}^k$. We compute the intermediate probabiliies,

$$p_i^L = \frac{\exp(z_i^L)}{\sum_{j=1}^k \exp(z_j^L)}$$

Thus the vector $\mathbf{p}^L$ of probabilities is just $\text{SoftMax}(\mathbf{z}^L)$. Finally, $a_i^L = \log(p_i^L)$.

The loss function we use is the negative log likelihood. Let $y \in \{1, \ldots, k\}$ be the class label of the input vector $\mathbf{x}$ that was used to produce $\mathbf{a}^L$ as the network output. Then the loss:

$$\ell(\mathbf{a}^L, y) = -a_y^L$$

Observe that $\mathbf{a}^L$ already contains logarithms of the estimated probabilities. Compute $\boldsymbol{\delta}^L = \frac{\partial \ell}{\partial \mathbf{z}^L}$.

2.2 Weight Initialization

- One common way to initialize weight parameters of the neural network is by choosing small random values. If instead, you set every weight parameter to the same small value ϵ, what do you think will happen? Consider a simple network with two input units, a hidden layer with two units and one output unit and explain how the weights will change when you use stochastic gradient descent.

- It is also important to think about choosing the initialization for the biases. For example when using simoid neurons, you may want to initialize the bias also to small values around 0. If on the other hand you are using a ReLU unit (see part 3), then you may want to initialize with a somewhat positive value. Can you think of reasons why?

3 Reading and Research

In this part you are asked to answer a few question about neural network training and choosing units. Unlike other machine learning methods we have seen in the course, the optimization problems arising in training neural networks are highly non-convex. While convex programming has become a technology, with robust and efficient optimization packages, non-convex problems often require a good deal of insight into the problem and some art in designing the problem structure and optimization procedure.

You should attempt to answer the following questions. You may have to refer to the online textbooks, the research papers listed below, or the world wide web.

1. (Refer to Glorot and Bengio (2010)) What is saturation? Why might this be a problem? As an extreme case, imagine using a new nonlinearity:

$$\text{almost-sign}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \le x < 1 \\ 1 & \text{if } 1 \le x \end{cases}$$

   What happens if an entire middle layer is driven into saturation with this nonlinearity?

2. What is a rectified linear unit (RELU)? In what case are they preferred to sigmoid or tanh neurons and why?

3. What is a softplus unit? How does it compare to the ReLU unit?

4. Explain dropout, a technique invented by Geoff Hinton and collaborators, in the context of deep learning.

Here is a list of suggested articles for reading this week. You will learn a great deal about some of the difficulties and possible ways to overcome them while training deep neural networks. We don't expect that you will understand everything in these articles, however, try to understand the key difficulties and ideas.

1. **Understanding the difficulty of traning deep feedforward neural networks**. X. Glorot and Y. Bengio. *AISTATS 2010*.

2. **Deep Sparse Rectifier Neural Networks**. X. Glorot, A. Bordes and Y. Bengio. *AISTATS 2011*.

3. **Dropout: a simple way to prevent neurons from overfitting**. N. Srivastava, G. Hinton, A. Krizhevksy. *JMLR 2014*.