UNIVERSITY OF OXFORD

**Machine Learning**
*Hilary Term 2016*
*Week 6*

# Classifying digits using SVM

In this practical, we address the problem of classification using support vector machines.

The MNIST dataset consists of 1024-dimensional inputs corresponding to pixels of a $32 \times 32$ image showing a handwritten digit from 0 to 9, and corresponding labels stating what digit the image shows, the goal is to learn a classifier that predicts the class (label) of several test-set inputs. For support vector machines, it is a lot more efficient to use a sparse representation of the input data—many pixels are 0 across all the dataset, and every individual image has a large number of 0 pixels. We will use a torch wrapper for the popular libsvm package.

The files for this practical can be found here: https://github.com/oxford-cs-ml-2016/practical4

Read `README.md` for setup instructions for the lab machine. Clone the entire repository or download all the `.sh` and `.lua` files.

After you run `practical4.sh` you should have a directory named `practical4-data` which contains the MNIST data that is tailored for the svm package. The file practical4.lua already has a function defined to load this data and divide this into training, validation and test sets of different sizes.

## Task 1: Grid Search

The aim is to find suitable parameters for SVM. In the first part of the assignment you will be using SVM with RBF or Gaussian Kernels. The parameter $\gamma = \frac{1}{2\sigma^2}$ and $C$ can be passed to the training procedure for libsvm. To understand the full range of parameters that can be passed, do the following:

```
require 'svm';
libsvm.train()
```

If you call libsvm.train() without any parameters, it will show you the options that you can pass to the classifier. To train and test do the following:

```
model = libsvm.train(train_data, '-q -g 0.01 -c 2')
_,accuracy,_ = libsvm.predict(test_or_validation_data, model)
```

In the first part you are training with $\gamma = 0.01$ and $C = 2$. The `-q` flag ensures that no libsvm output is shown. The prediction function returns several parameters, you are mainly concerned with the accuracy. Note that `accuracy[1]` contains the actual prediction accuracy that you will need.

Since the running time to train SVM can be substantial, you can start with a small subset of the data. The starter lua code we have provided makes a small dataset of 2000 training examples

and 2000 validation examples. The medium sized dataset contains 10,000 training examples and 2000 validation examples and the large dataset contains 48,000 training examples and 12,000 validation examples. All of these are taken from the MNIST training dataset; we have not touched the test dataset, which we will only use for the last part.

We suggest starting with $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{13}\}$ and $\gamma \in \{2^{-13}, 2^{-11}, \ldots, 2^{5}\}$. First run the grid search on the small dataset.

Then choose a smaller and finer grid, $C \in \{c^*/4, c^*/2, c^*, 2c^*, 4c^*\}$ and $\gamma \in \{\gamma^*/4, \gamma^*/2, \gamma^*, 2\gamma^*, 4\gamma^*\}$, where $c^*$ and $\gamma^*$ are the best values obtained from the grid search with the small dataset.

Finally, you might want to take an even smaller grid in the last step, say three values of $C$ and $\gamma$ each which differ multiplicatively by 1.5 on the large validation set. If you find that your program is taking too long to run, try reducing the sizes of the datasets, or skip the large set completely and choose the parameters based on the medium sized dataset.

When you have found the parameters that give good *validation* accuracy, train on the full training set and report the accuracy on the *test set*.

- What values of $C$ and $\gamma$ did you get from the first grid search?
- What part of the grid do you think is underfitting? What part is overfitting?
- What were the final values of $C$ and $\gamma$ that you used for training on the full dataset?
- What was the accuracy you obtained on the test set?

**Hand in answers to the above questions**

## Task 2: Polynomial Kernels

In this second part you are asked to train svm using a polynomial kernel. Since, the training procedures take a while, we'll restrict our attention to using degree 2 polynomial kernels.

Use a similar method to find the best value of $C$. Start with $C \in \{8, 32, ..., 8192\}$, and refine the grid as you feel necessary. `libsvm` will allow you to set a value of gamma, but in the first instance, we recommend letting it be the default value. Note that overfitting is reduced when you use a larger dataset, so it's a good idea to search around the values you found in the grid search performed using a small dataset.

- What value of C did you choose after the gridsearch?
- What test accuracy did you get on the full test set?
- How does this compare to what you got using RBF kernels?

**Hand in answers to the above questions**

## Handin

See directions above. Hand in answers to 1 and 2.

## Advanced: For enthusiastic students

You will notice that it takes a while to train SVM classifiers. As you wait, try to figure out how to implement SVMs directly in torch using the `nn` package and train using stochastic gradient descent. In particular, look at `nn.MarginCriterion`. See how you can use this criterion together with passing weight decay parameters to optimization algorithms, or implementing the regularization directly in the closure, to implement SVMs directly in torch.