ML1 MNIST Challenge Report

Yelysei Bondarenko – yell.bondarenko@gmail.com

Results

Algorithm	Details	Test Error, %
k-NN	3-NN, Euclidean distance, uniform weights. ***Preprocessing***: Feature vectors extracted from **NN**.	**1.13**
k-NN$_2$	3-NN, Euclidean distance, uniform weights. ***Preprocessing***: Augment (training) data ($\times 9$) by using random rotations, shifts, Gaussian blur and dropout pixels; PCA-35 whitening and multiplying each feature vector by $e^{11.6s}$, where s – normalized explained variance by the respective principal axis. (equivalent to applying PCA whitening but with accordingly weighted Euclidean distance.)	**2.06**
NN	MLP 784–1337–D(0.05)–911–D(0.1)–666–333–128–10 (D-dropout); hidden activations – leaky ReLU $\max\{0.01x, x\}$, output – softmax; loss – categorical crossentropy; 1024 batches; 42 epochs; optimizer – Adam (learning rate $5 \cdot 10^{-5}$, rest – defaults from paper). ***Preprocessing***: Augment (training) data ($\times 5$) by using random rotations, shifts, Gaussian blur.	**1.04**
LogReg	32 batches; 91 epoch; L2-penalty, $\lambda = 3.16 \cdot 10^{-4}$; optimizer – Adam (learning rate 10^{-3}, rest – defaults from paper). ***Preprocessing***: Feature vectors extracted from **NN**.	**1.01**
GP	794 random data points were used for training; $\sigma_n = 0$; RBF kernel ($\sigma_f = 0.4217$, $\frac{1}{2l^2} = 0.0008511$); Newton iterations for Laplace approximation till ΔLog-Marginal-Likelihood $\leq 10^{-7}$; Solve linear systems iteratively using CG with 10^{-7} tolerance; for prediction generate 2000 samples for each test point. ***Preprocessing***: Feature vectors extracted from **NN**.	**1.61**

Remarks

The average error therefore is $(1.13 + 1.04 + 1.01 + 1.61)/4 = \mathbf{1.1975}\%$. I also included **k-NN$_2$** that does only basic preprocessing and still shows quite competitive results. BTW, after submission I realized that I forgot to subtract mean for final GP model, so its actually **1.59**%.

Further Ideas

Here the list of what also can be tried regarding these particular 4 ML algorithms (didn't have time to check it, or it was forbidden by the rules, e.g. ensemble learning):

- Bagging for k-NN: train a group of k-NNs with different parameter k (say, 2, 4, ..., 128) and average;
- More sophisticated metrics (say, from `scipy.spatial.distance`) for k-NN;
- Weighting metrics according to some other functions of explained variance from PCA;
- NCA;
- Different kernels or compound kernels for k-NN;

- Commitee of MLPs, CNN, commitee of CNNs or more advanced NNs (these should literally kill MNIST xD);
- Unsupervised pretraining for MLP/CNN ...;
- Different kernels or compound kernels for GPs;
- Derive equations for derivatives of Log-Marginal-Likelihood for multiclass Laplace approximation w.r.t kernel parameters in order to optimize them by using gradient-based methods;
- Bagging for GPs: train a collection of GPs on different parts of the data and then average their predictions;
- IVM.