**Introduction to Machine Learning (67577)**

# Problem Set 3 – Convexity and Neural Networks

*Due: 18.5.2017*

## Submission guidelines

- The assignment is to be submitted via Moodle only.
- Files should be zipped to ex_N_First_Last.zip (In this case N stands for 3, First is your first name, Last is your last name. In English, of course).
- The .zip file size should not exceed 10Mb.
- All answers should be submitted as a single pdf file. We encourage you to typeset your answers (either LaTeX, Lyx or Word), but it is not mandatory. Note that if you choose to scan a handwritten solution, then answers written in incomprehensible handwriting or scanned in poor quality will be counted as wrong answers. Moreover, the scans should be integrated with the plots to a single coherent pdf file.
- All of your Python code files used in the practical question should be attached in the zip file. There should be one .py file, other then that there are no special requirements or guidelines for your code. Note that if your code doesn't run on the CS environment you will not get points for this question.

## 1 Convexity (24 points)

Prove or disprove the following claims:

1. If $C_1, C_2 \subseteq \mathbb{R}^n$ are convex sets and $\alpha, \beta \in \mathbb{R}$ then the set $\{\alpha c_1 + \beta c_2 : c_1 \in C_1, c_2 \in C_2\}$ is convex.
2. If $f, g : \mathbb{R}_+ \to \mathbb{R}_+$ are convex functions then their composition $f \circ g$ is also convex.
3. If $f : \mathbb{R} \to \mathbb{R}$ is a convex function and $c \in \mathbb{R}$ then $c \cdot f$ is convex.

4. If $f, g : \mathbb{R} \to \mathbb{R}$ are convex functions then $\min\{f, g\}$ is a convex function.

5. If $f, g : \mathbb{R} \to \mathbb{R}$ are convex functions then $f - g$ is a convex function.

6. The function $f(x) = x^2$ is not $\rho$-Lipschitz over $\mathbb{R}$ for any $\rho$.

## 2 Convexity is not Sufficient for PAC Learnability (15 points)

In this question you will prove that not all convex problems are learnable. Specifically we will consider the linear regression problem with $d = 1$, i.e., $l(w, (x, y)) = (wx - y)^2$, where $w, x, y \in \mathbb{R}$. Show that for any $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and any deterministic learning algorithm there are $\epsilon, \delta \in (0, 1)$ and a distribution $\mathcal{D}$ over $\mathbb{R} \times \mathbb{R}$ such that when running the algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns a hypothesis $h$ (that is defined by $\hat{w} \in \mathbb{R}$) such that with probability of at least $1 - \delta$ (over the choice of the $m$ training examples),

$$L_{\mathcal{D}}(h) > \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon.$$

Hint: a possible solution will consider two distributions $\mathcal{D}_1, \mathcal{D}_2$ such that the learning algorithm cannot be correct on both of them. One of the distributions may include an example with low probability but with large loss.

## 3 Neural Networks are not Convex Learning Problems (8 points)

Show that even the class of all neural networks with only one neuron, one hidden layer, ReLU activation function and the squared loss do not define a convex learning problem. First formally write the ingredients of the learning problem: $(\mathcal{H}, \mathcal{Z}, \ell)$.

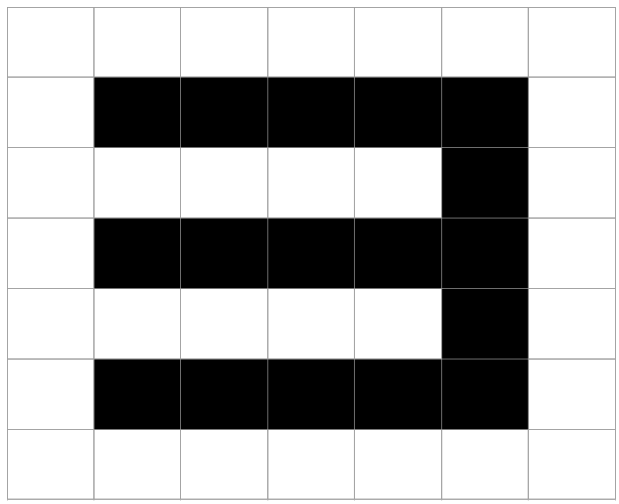
## 4 Convolution Layer (5 points)

Design a convolution layer that:

- gets a black-white image $M$ of size $m \times n$, where $M_{i,j} = 1$ if $(i, j)$ square is black and otherwise $M_{i,j} = -1$

- returns a $m-2 \times n-2$ array $M'$ such that $M'_{i,j} = 9$ if a horizontal black line of length 3 starts from the coordinate $(i, j-1)$ in $M$ and $M'_{i,j} < 9$ otherwise. That is if $M_{i,j-1} = M_{i,j} = M_{i,j+1} = 1$ and $M_{i-1,j-1} = M_{i-1,j} = M_{i-1,j+1} = M_{i+1,j-1} = M_{i+1,j} = M_{i+1,j+1} = -1$ then $M'_{i,j} = 9$. Otherwise $M'_{i,j} < 9$.

For example, the following image



is represented as

| | | | | | | |
|---|---|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 1 | 1 | 1 | 1 | 1 | -1 |
| -1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | 1 | 1 | 1 | 1 | 1 | -1 |
| -1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | 1 | 1 | 1 | 1 | 1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 |

and a possible answer is

| | | | | |
|---|---|---|---|---|
| 0 | 9 | 9 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 9 | 9 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 9 | 9 | 0 | 0 |

Bonus (2 points): return a $m-2 \times n-2$ array $M'$ such that $\mathbf{M'_{i,j}} = \mathbf{1}$ if a horizontal black line of length 3 starts from the coordinate $(i, j-1)$ in $M$ and $\mathbf{M'_{i,j}} = \mathbf{0}$ otherwise.

Hint: watch Shai's lecture on neural networks.

## 5 Subgradient of $L_{\mathcal{D}}$ (8 points)

Prove that if for all $z \in \mathcal{Z}$, $\mathbf{v}_z \in \partial l(\mathbf{w}, z)$ then $\mathbb{E}_{z \sim \mathcal{D}}[\mathbf{v}_z] \in \partial L_{\mathcal{D}}(\mathbf{w})$.

## 6 Practical Question: Learning with Neural Networks (40 points)

In this question we explore the meaning of the different parameters of neural networks. In Moodle you can find the code used in class to learn MNIST using the LeNet architecture.

For each of the following items (1–6):

- change the program accordingly.
- draw in one plot the training and test error at each time step before and after the change.
- explain the plot you got

1. GD versus SGD — We learned two algorithms: GD which uses all the training data at each step and SGD which uses one example at each step. Change the algorithm to run with different batch sizes (by changing the command `mnist.train.next_batch(32)`). In addition to the plot, state and explain how does the running time change as a function of the batch size (use `import time`).

2. Learning Rate — Change the algorithm to run with different learning rates (by changing the command tf.train.GradientDescentOptimizer(learning_rate=0.1)).

3. Learning algorithm — Change the optimization algorithm from GradientDescentOptimizer to other optimization algorithms (e.g., to AdamOptimizer or AdagradOptimizer).

4. Activation Function — Change the activation function (e.g., using activation_fn=tf.nn.relu).

5. Initialization — Change the weights at the initialization.

6. Architecture — Change the neural network architecture.

Please attach all the relevant code and plots.

Tip: to import the package cv2 you can download it using the command pip install opencv-python

### 6.1 CIFAR10 (15 bonus points)

CIFAR10 is another famous benchmark. It contains $32 \times 32$ color images in 10 classes. You can download it from `https://www.cs.toronto.edu/~kriz/cifar.html`. In this bonus question your goal is to learn CIFAR10 using a neural network. Submit your code, explain your steps in the solution, and print the test error after training. The code should be submitted in a single file called: bonus.py