# Home Assignment 1

Christophe Saad
April 22, 2020

1. The inputs are grayscale (28 x 28) images, each pixel is represented by an 8-bit integer giving 256 possible levels of gray. We flatten the image to get an input vector of dimension 784.
The input space is thus $\mathcal{X} = \{0, \ldots, 255\}^{784}$ and is of dimension $d = 784$.
The program outputs 1 if the image is recognized as an A and -1 otherwise. The output space is $\mathcal{Y} = \{-1, 1\}$ and is of dimension 1.
The training data are represented as $D_{train} = \{(X_i, Y_i) \in \mathcal{X} \times \mathcal{Y} \,|\, 1 \leq i \leq n\}$, the set of tuples of the images pixels along with their labels, where $(X_i, Y_i)$ are i.i.d. random variables following distribution $\mathcal{P}$ in $\mathcal{X} \times \mathcal{Y}$.

2. (a) The risk associated with the 0-1 loss is

$$R(f) = \mathbb{E}\left[\ell_1(f(X), Y)\right] = \mathbb{E}\left[\mathbb{1}_{f(X) \neq Y}\right] = \mathbb{P}(f(X) \neq Y)$$

where $(X, Y) \sim \mathcal{P}$.
The empirical risk is

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell_1(f(X_i), Y_i) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{f(X_i) \neq Y_i}$$

The 0-1 loss function is not convex and not differentiable so we can't apply convex optimization method which make the problem $\arg\min_f \hat{R}(f)$ hard to solve.

(b) Since we work on the minimization of empirical risk, we compute the predictor according to the samples $D_{train}$ which are not 100% representative of the distribution of $(X, Y)$. A predictor may predict the training set very well and fails on predicting new data (for example when the training set is not representative enough).
We use the test data $D_{test} = \{(X_i, Y_i) \in \mathcal{X} \times \mathcal{Y} \,|\, 1 \leq i \leq n_1\}$ where $(X_i, Y_i)$ follow the same distribution $\mathcal{P}$ in order to confirm the performance of the predictor and to simulate an arrival of new samples.
We measure the test error in order to get an idea of the expected error and be able to avoid overfitting and underfitting.

(c) We are considering linear models so we look for predictors of the form

$$f(x) = \begin{cases} 1 & \theta^\top x \geq 0 \\ -1 & \theta^\top x < 0 \end{cases}$$

We let $X = (X_1, ..., X_n)^\top \in \mathbb{R}^{n \times d}$ and $Y = (Y_1, ..., Y_n)^\top \in R^n$ for $(X_i, Y_i) \in D_{train}$

- Linear Least Square Regression

$$\arg\min_f \hat{R}(f) = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{n} ||Y - X\theta||_2^2$$

- Linear Logistic Regression

$$\arg\min_f \hat{R}(f) = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell_3(\theta^\top X_i, Y_i) = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-(\theta^\top X_i) Y_i})$$

3. (a) Performances of gradient descent and stochastic gradient descend with squared and logistic losses.

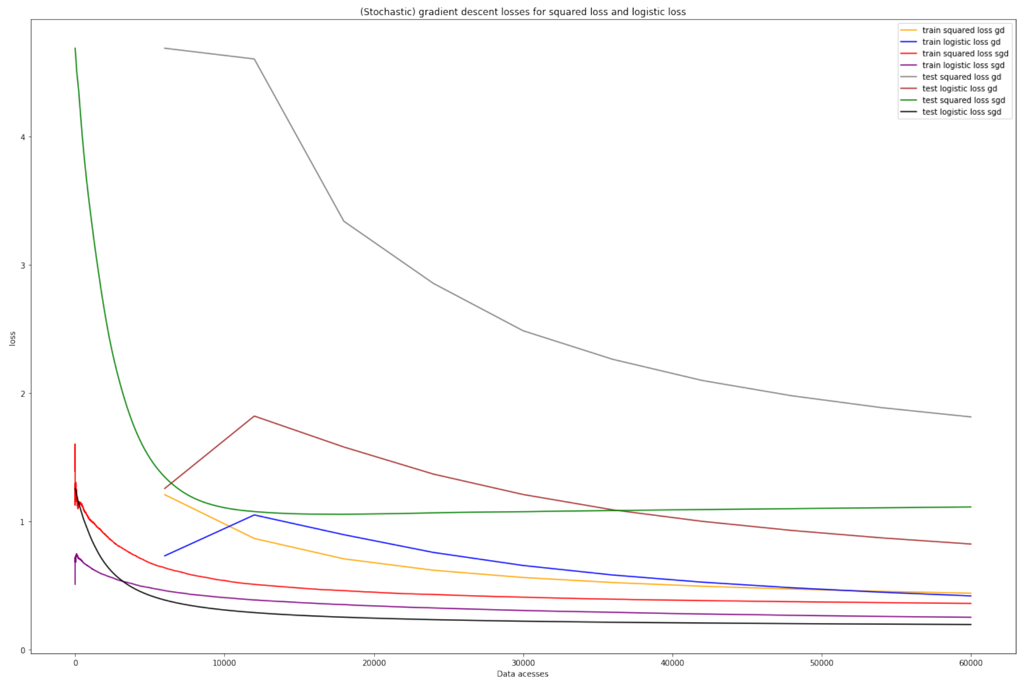

Figure 1: Cumulative mean squared loss and logistic loss

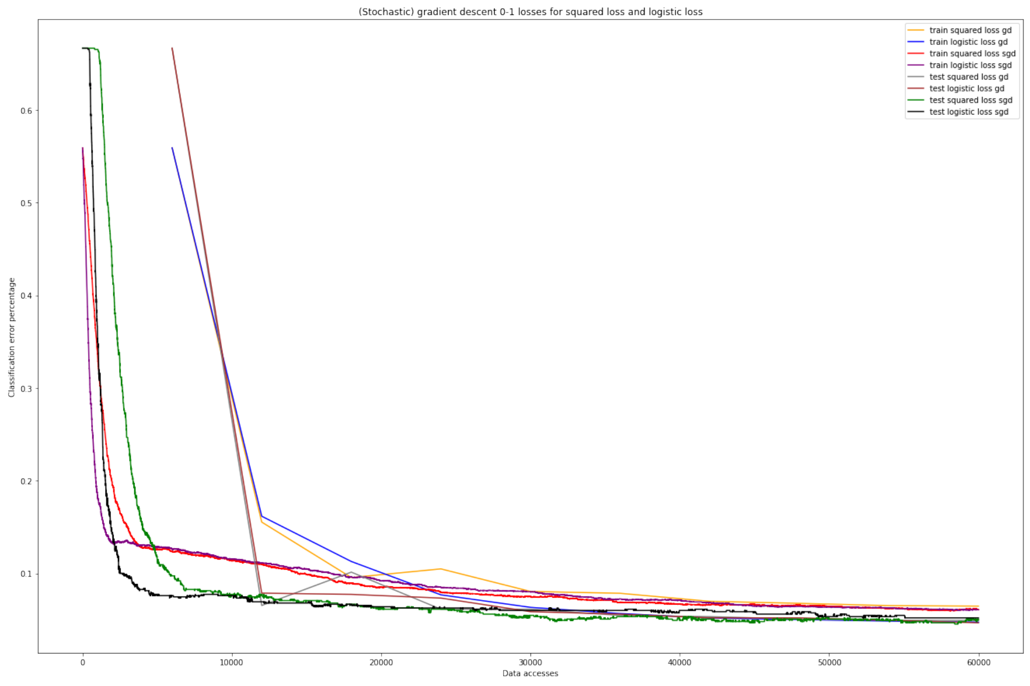

Figure 2: 0-1 loss of squared loss and logistic loss estimations

(b) Estimators of GD and SGD for squared and logistic losses.

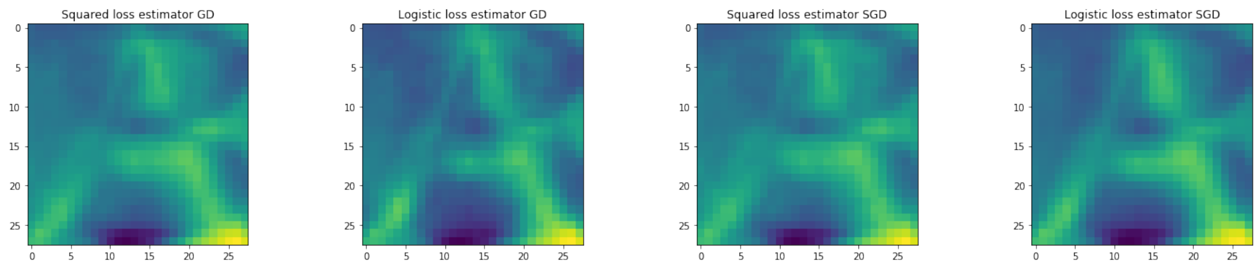

Figure 3: Estimators of squared and logitic losses for GD and SGD

(c) Averaged estimators.

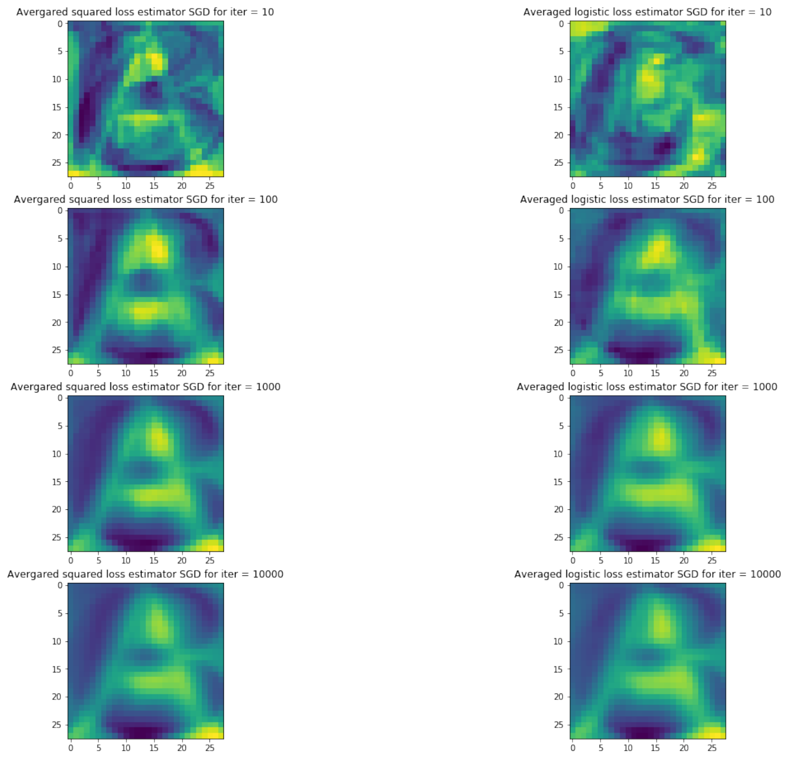

Figure 4: Estimators of squared and logitic losses for GD and SGD

4. (a) $k$-NN is a supervided learning classification method. It assigns a label to an input according to the most frequent label of the $k$ closest training data. $k$-NN requires a proximity measure in order to compare the distance between data points. The $\ell_2$ metric is the Euclidean norm $d(x_i, x_j) = ||x_i - x_j||_2$

(b) $k$-nn performances for some values of $k$

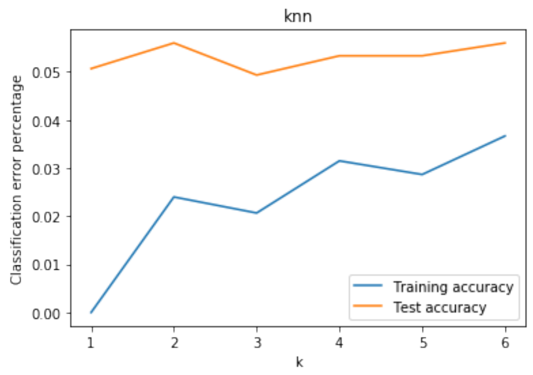

Figure 5: Cumulative mean squared loss and logistic loss

(c) $K$-fold cross-validation.

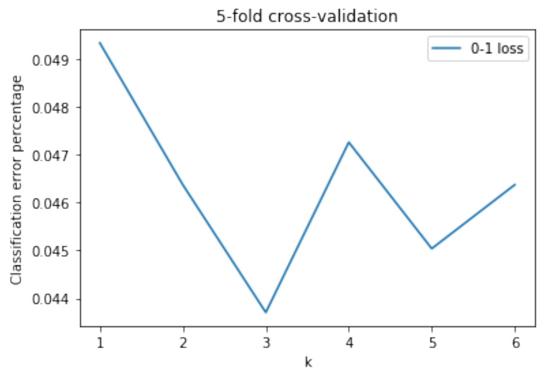

Figure 6: Cumulative mean squared loss and logistic loss

5. We take for $k$-NN $k = 3$ which has the best performances in both experiments.

| | **Logistic regression** | **Linear regression** | **$k$-NN ($k = 2$)** |
|---|---|---|---|
| Empirical error (0-1 loss) | 0.048 | 0.056 | 0.021 |
| Test error (0-1 loss) | 0.047 | 0.048 | 0.049 |

The error rate of the three method is approximatly similar for the test error.

Concerning the train error rate, $k$-NN performs better. Indeed, it classifies the training set by using the training labels as input.

## Implementation details

- I used the behaviors on Figure 1 to calibrate the learning rates. I checked for which value the losses converged the best.
- You can choose to reduce the dimension of the input space by changing the $size$ and $d$ variables in the Loading Data section. The image will be loaded with the given size (pixels are interpolated).
- You can choose to balance the dataset for $k$-NN by specifying the $train\_range$. Balancing the dataset does not significantly yield better results.
- $k$-NN is implemented using parallelism, I compute the loss of each $k$ on a different thread.