Intro to ML - HW3

Daniel Volkov, I.D: 330667494

July 27, 2024

Question 1

1. Let $\lambda \in [0,1]$, and let $x_1, x_2 \in \mathbb{R}^n$. From definition of g:

$$\begin{aligned} g(\lambda x_1 + (1-\lambda)x_2) &\triangleq f(A(\lambda x_1 + (1-\lambda)x_2) + b) = f(\lambda Ax_1 + \lambda b + (1-\lambda)Ax_2 + (1-\lambda)b) \\ &= f(\lambda(Ax_1+b) + (1-\lambda)(Ax_2+b)) \underset{\text{(From convexity of } f)}{\leq} \lambda f(Ax_1+b) + (1-\lambda)f(Ax_2+b) \\ &\triangleq \lambda g(x_1) + (1-\lambda)g(x_2) \end{aligned}$$

And so we got that g is convex by definition. ■

2. Let $\lambda \in [0,1]$, and let $x_1, x_2 \in \mathbb{R}^n$. Denote $j := \arg\max_i\{f_i(\lambda x_1 + (1-\lambda)x_2)\}$.
From definition of g:

$$\begin{aligned} & g(\lambda x_1 + (1-\lambda)x_2) \triangleq \max_i\{f_i(\lambda x_1 + (1-\lambda)x_2)\} = f_j(\lambda x_1 + (1-\lambda)x_2)) \\ & \underset{\text{(From convexity of } f_j)}{\leq} \lambda f_j(x_1) + (1-\lambda)f_j(x_2) \leq \lambda \max_i\{f_i(x_1)\} + (1-\lambda)\max_i\{f_i(x_2)\} \triangleq \lambda g(x_1) + (1-\lambda)g(x_2) \end{aligned}$$

And so we got that g is convex by definition. ■

3. Let $\lambda \in [0,1]$, and let $x_1, x_2 \in \mathbb{R}^n$. As we know (Part of the definition of a norm), $\|\cdot\|_2$ satisfies the triangle inequality. Thus:

$$\|\lambda x_1 + (1-\lambda)x_2\|_2 \leq \lambda\|x_1\|_2 + (1-\lambda)\|x_2\|_2$$

Meaning that $x \mapsto \|x\|_2$ is a convex function.
Notice that $\frac{d}{dx}(\frac{d}{dx}(x^2)) \equiv 2 \geq 0$, and thus $x \mapsto x^2$ is a convex function over $\mathbb{R}$ (From hint in (e)).
Also notice that $x \mapsto x^2$ is an increasing function over $[0,\infty)$, and thus we get:

$$\begin{aligned} f(\lambda x_1 + (1-\lambda)x_2) &\triangleq \|\lambda x_1 + (1-\lambda)x_2\|_2^2 \\ &\underset{\text{(From convexity of } \|\cdot\|_2)}{\leq} (\lambda\|x_1\|_2 + (1-\lambda)\|x_2\|_2)^2 \\ &\underset{\text{(From convexity of } x^2)}{\leq} \lambda\|x_1\|_2^2 + (1-\lambda)\|x_2\|_2^2 \triangleq \lambda f(x_1) + (1-\lambda)f(x_2) \end{aligned}$$

And so we got that f is convex by definition.

4. Remember that A is PSD $\iff \exists B \in \mathbb{R}^{n\times n}: \ A = B^tB$ (From ex.1 in this course).
Let $B \in \mathbb{R}^{n\times n}$ such that $A = B^tB$. Notice the following:

$$f(x) = x^tAx = \langle Ax, x\rangle = \langle B^tBx, x\rangle = \langle Bx, Bx\rangle = \|Bx\|_2^2$$

Meaning we got that $f(x) = \|h(x)\|_2^2$, when h is an affine (actually linear) transformation. From questions (a)+(c), we conclude that f is convex.

5. Let's look at the second derivative of $\ell_{\log}$:

$$\frac{d}{dx}\left(\frac{d}{dx}\ell_{\log}(x)\right) = \frac{d}{dx}\left(\frac{d}{dx}\log_2(1+e^{-x})\right) = \frac{d}{dx}\left(-\frac{\mathrm{e}^{-x}}{\ln(2)\left(\mathrm{e}^{-x}+1\right)}\right)$$
$$= \frac{\mathrm{e}^{x}}{\ln(2)\left(\mathrm{e}^{x}+1\right)^2}$$

As we know $\forall x \in \mathbb{R}: \; e^x > 0 \wedge (e^x+1)^2 > 1 \wedge \ln(2) > 0$. This of course implies that the second derivative of $\ell_{\log}$ is non-negative for all $x \in \mathbb{R}$. From the hint, we conclude that $\ell_{\log}$ is a convex function.

Now let's look at f:
Let $\lambda \in [0,1]$, and let $w_1, w_2 \in \mathbb{R}^d$. We get:

$$\begin{aligned} f(\lambda w_1 + (1-\lambda)w_2) &\triangleq \ell_{\log}(y\langle \lambda w_1 + (1-\lambda)w_2, x\rangle) = \ell_{\log}(y\cdot(\lambda\langle w_1, x\rangle + (1-\lambda)\langle w_2, x\rangle)) \\ &= \ell_{\log}(\lambda y\langle w_1, x\rangle + (1-\lambda)y\langle w_2, x\rangle) \\ \underset{\text{(From convexity of } \ell_{\log})}{\leq} & \lambda\ell_{\log}(y\langle w_1, x\rangle) + (1-\lambda)\ell_{\log}(y\langle w_2, x\rangle) \triangleq \lambda f(w_1) + (1-\lambda)f(w_2) \end{aligned}$$

Meaning that f is a convex function by definition. ■

Question 2

1. Let $w^* \in \mathbb{R}^d$ such that $\forall 1 \le i \le n :\ y_i w^* \cdot x_i > 0$.
   Denote $m_0 := \min_i\{y_i w^* \cdot x_i\}$. We get that $\forall 1 \le i \le n :\ y_i w^* \cdot x_i \ge m_0 > 0$.

   Let $\varepsilon > 0$. Choose $w_\varepsilon = w^* \cdot -\frac{\ln(2^\varepsilon - 1)}{m_0}$. We get: (For small enough ε)

$$y_i w_\varepsilon \cdot x_i = -\frac{\ln(2^\varepsilon - 1)}{m_0} \cdot y_i w^* \cdot x_i \ge -\ln(2^\varepsilon - 1)$$
$$\implies -y_i w_\varepsilon \cdot x_i \le \ln(2^\varepsilon - 1) \implies \log_2(1 + e^{-y_i w_\varepsilon \cdot x_i}) \le \varepsilon$$

   And so we conclude:

$$L(w_\varepsilon) = \frac{1}{n}\sum_{i=1}^{n} \log_2(1 + e^{-y_i w_\varepsilon \cdot x_i}) \le \frac{1}{n}\sum_{i=1}^{n} \varepsilon = \varepsilon$$

   Notice that the weak inequality I proved ($L(w_\varepsilon) \le \varepsilon$) also proves the version with the strong inequality; since ε was arbitrary, we could choose a bit smaller ε, and get the version with the strong inequality. ■
   (Just like the equivalence of strong/weak inequality in the limit's definition)

2. Notice that:

$$\ell_{\log}(y_i w \cdot x_i) < 1 \iff \log_2(1 + e^{-y_i w \cdot x_i}) < 1 \iff e^{-y_i w \cdot x_i} < 1 \iff y_i w \cdot x_i > 0$$

   Meaning that the classifier is *correct* on a sample iff the log-loss on the sample is less than 1.

   Choose $\varepsilon = \frac{1}{n}$. Let w_ε such that $L(w_\varepsilon) < \varepsilon = \frac{1}{n}$. We conclude:

$$L(w_\varepsilon) = \frac{1}{n}\sum_{i=1}^{n} \ell_{\log}(y_i w_\varepsilon \cdot x_i) < \frac{1}{n}$$
$$\implies \sum_{i=1}^{n} \ell_{\log}(y_i w_\varepsilon \cdot x_i) \le 1$$

   ($\ell_{\log}$ is positive $\forall x \in \mathbb{R}$) $\implies \forall 1 \le i \le n :\ \ell_{\log}(y_i w_\varepsilon \cdot x_i) < 1 \implies \forall 1 \le i \le n :\ y_i w_\varepsilon \cdot x_i > 0$

   And the last conclusion of course implies that $\forall 1 \le i \le n :\ sign(w \cdot x_i) = y_i$. ■

Question 3

Let $t \in \mathbb{N}$. From the recursive formula of GD, we get:

$$x_{t+1} - x_t = -\eta \nabla f(x_t)$$

f is β-smooth, and thus we get: (Definition for x_{t+1}, x_t)

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) + \nabla f(x_t)^t \cdot (x_{t+1} - x_t) + \frac{\beta}{2} \cdot \|x_t - x_{t+1}\|^2 \iff \\ f(x_{t+1}) - f(x_t) &\leq \nabla f(x_t)^t \cdot (-\eta \nabla f(x_t)) + \frac{\beta}{2} \|\eta \nabla f(x_t)\|^2 \iff \\ f(x_{t+1}) - f(x_t) &\leq (\frac{\beta}{2}\eta^2 - \eta) \cdot \|\nabla f(x_t)\|^2 = \eta \cdot (\frac{\beta}{2}\eta - 1) \cdot \|\nabla f(x_t)\|^2 \end{aligned}$$

Notice that $\eta < \frac{2}{\beta}$, and thus $\eta(\frac{\beta}{2}\eta - 1) < 0$. This implies:

$$\frac{f(x_{t+1}) - f(x_t)}{\eta(\frac{\beta}{2}\eta - 1)} \geq \|\nabla f(x_t)\|^2 \geq 0$$

Notice that $\|\nabla f(x_t)\|^2 \geq 0$ since the norm is non-negative. Since the LHS denominator is negative, this implies that also the numerator is non-positive, Meaning that $f(x_{t+1}) \leq f(x_t)$. Since this is true for all t (t was arbitrary), we conclude that $\{f(x_n)\}_{n=0}^{\infty}$ is a decreasing sequence.
Also notice that $\forall x \in \mathbb{R}^n : \ f(x) \geq 0$, meaning that $\{f(x_n)\}_{n=0}^{\infty}$ is a bounded sequence, which implies (together with monotony) that $\{f(x_n)\}_{n=0}^{\infty}$ is convergent to a limit I'll denote with c.
Now notice the following (telescoping) sum:

$$\sum_{t=0}^{n} \frac{f(x_{t+1}) - f(x_t)}{\eta(\frac{\beta}{2}\eta - 1)} = \frac{f(x_{n+1}) - f(x_0)}{\eta(\frac{\beta}{2}\eta - 1)} \xrightarrow{n\to\infty} \frac{c - f(x_0)}{\eta(\frac{\beta}{2}\eta - 1)} \in [0, \infty)$$

And the sum is convergent to a non-negative real number from the reasons discussed above.
From the direct comparison test, we conclude that $\sum_{t=0}^{\infty} \|\nabla f(x_t)\|^2$ is convergent.
Thus, we get that the sequence $\{\|\nabla f(x_t)\|^2\}_{t=0}^{\infty}$ upholds:

$$\lim_{t\to\infty} \|\nabla f(x_t)\|^2 = 0 \implies \lim_{t\to\infty} \|\nabla f(x_t)\| = 0$$

Question 4

In tis question, ReLU is denoted by σ.
Firstly, a few key observations:

- $\forall x \in \mathbb{R}: \ \sigma(x) + \sigma(-x) = \max\{0, x\} + \max\{0, -x\} = |x|$
- $\forall x \in \mathbb{R}: \ \sigma(x) - \sigma(-x) = \max\{0, x\} - \max\{0, -x\} = x$

As we were told in the hint: $\max\{x_1, x_2\} = \frac{1}{2} \cdot (x_1 + x_2 + |x_1 - x_2|)$.

And so, if we choose $W_1 = \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix}$, $W_2 = \begin{pmatrix} 0.5 & -0.5 & 0.5 & 0.5 \end{pmatrix}$, $b_1 = b_2 = \bar{0}$, We get that our network will be: (For a given input $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$)

$$\begin{aligned} W_2(\sigma(W_1 x + b_1)) + b_2 &= \begin{pmatrix} 0.5 & -0.5 & 0.5 & 0.5 \end{pmatrix} (\sigma(\begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix})) \\ &= \frac{1}{2} \cdot (\sigma(x_1 + x_2) - \sigma(-(x_1 + x_2)) + \sigma(x_1 - x_2) + \sigma(-(x_1 - x_2))) \\ &= \frac{1}{2} \cdot (x_1 + x_2 + |x_1 - x_2|) = \max\{x_1, x_2\} \end{aligned}$$

Meaning that we expressed $f(x_1, x_2) = \max\{x_1, x_2\}$ using a neural network with one hidden layer and ReLU activation.

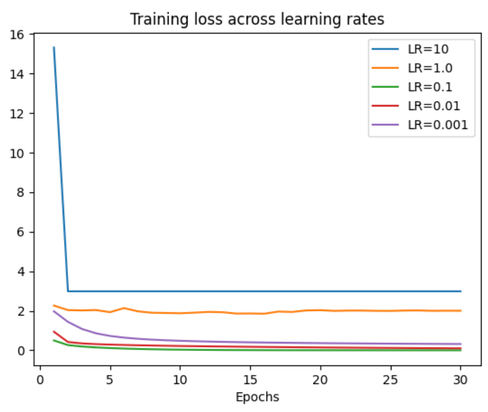
Neural networks

(a)

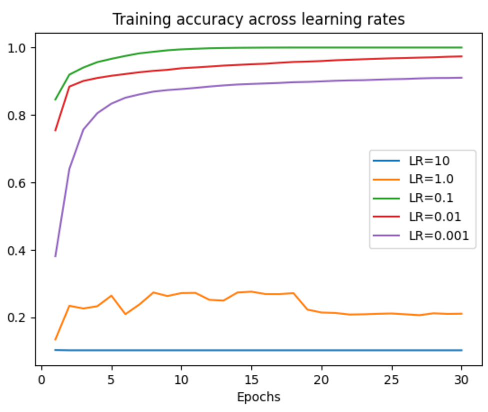
Code submitted.
I implemented mini-batching using matrix multiplication as instructed in recitation 6.
Notice that using this method, the partial derivatives in respect to W_ℓ should only be divided by the batch size, since the matrix multiplication already does the summation as part of the process. The partial derivatives in respect to b_ℓ result in a matrix using this method, and we should take the average of the columns to get the desired result.

Other than the normalization according to the batch size, the algorithm is exactly what we got in recitation 6.

(b)



(a) Training loss



(b) Training accuracy

Figure 1: Training loss and accuracy across epochs, over a range of different learning rates

As we can observe, the models with learning rate $\eta \leq 0.1$ seem to converge to a "good" model; low loss, high accuracy.
We can identify two phenomena:

- When the learning rate is *too big*:
  Our SGD optimizer does not yield good results: when η is *too big*, we get that the optimizer gets *stuck* at a high loss low accuracy scenario, which is caused by the gradient descent algorithm "overshooting" the minimum of the loss function, and achieving something close to "guessing accuracy".

  Also notice that for the biggest η value, we don't observe any change in loss/accuracy after the first epoch. This is a result of the learning rate being astronomically high, which amplifies this "overshooting" and yields something close to "guessing accuracy".

- When the learning rate is *too small*:
  Our SGD optimizer does converge to a good solution, but it does so very slowly, because each GD step only decreases the loss by a very small value.
  Also note that in the non-convex case, for smaller η values - the optimizer can get *stuck* in some sub-optimal local minima of the loss function.

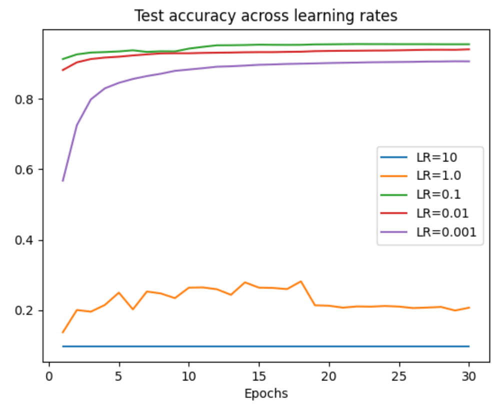


Figure 2: Test set accuracy of model across different learning rates

As we can observe from the third graph, the test set accuracy behaves very similarly to the training set accuracy. This stems from the traits of SGD we saw in class: SGD behaves as it was performing GD on the real loss, if the amount of epochs is small enough (not too much data repetition).

The final test set accuracy I got with $\eta = 0.1$ was 95.54%.

(c)

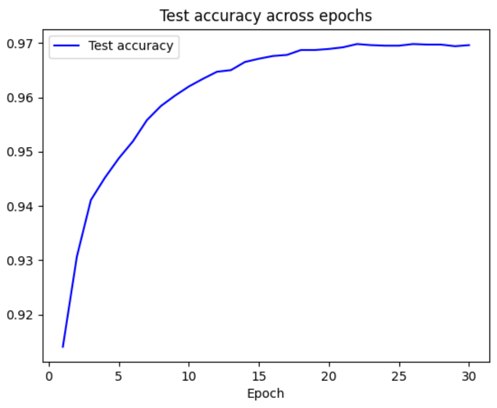
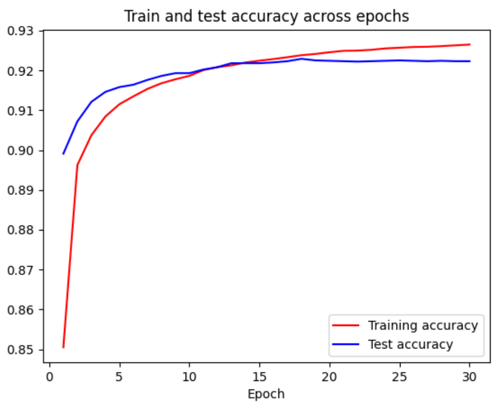


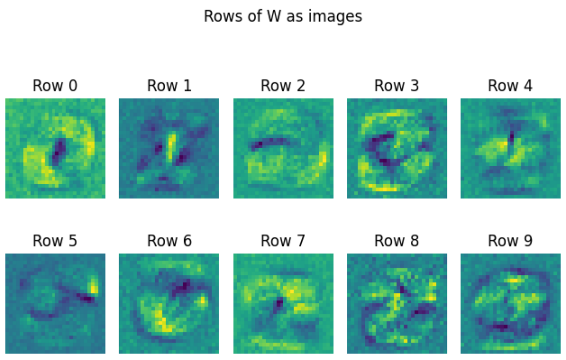
Figure 3: Test accuracy across epochs for the entire dataset

When training on the entire dataset I got a better result: 96.96% accuracy on the test set.

(d)



(a) Training and test accuracy of linear classifier



(b) Linear classifier rows as images

Figure 4: Training and test accuracy of linear classifier, and it's rows as images

As we can see, the rows of W (roughly) represent blurred digits, with each row corresponding to the digit in it's index. This stems from the fact that with a linear classifier, each row of the weight matrix is the vector of weights for the corresponding digit - when multiplying, we get:

$$Wx = \begin{pmatrix} -\langle W_0, x\rangle- \\ \vdots \\ -\langle W_9, x\rangle- \end{pmatrix}$$

And so because of that, it make sense that every row of W should be something that resembles the corresponding digit. The noise is because a big dataset is used and our algorithm tries to fit the model to the whole dataset, rather than a specific sample.

(e)

I used a network with the following architecture:

- Layer dimensions of $[784, 1000, 500, 250, 100, 50, 10]$, when 784 is the input and 10 is the output.
- Learning rate of $\eta = 0.1$
- Batch size of 100

And I ran training for 20 epochs.
This method resulted in test set accuracy of 98.04%.

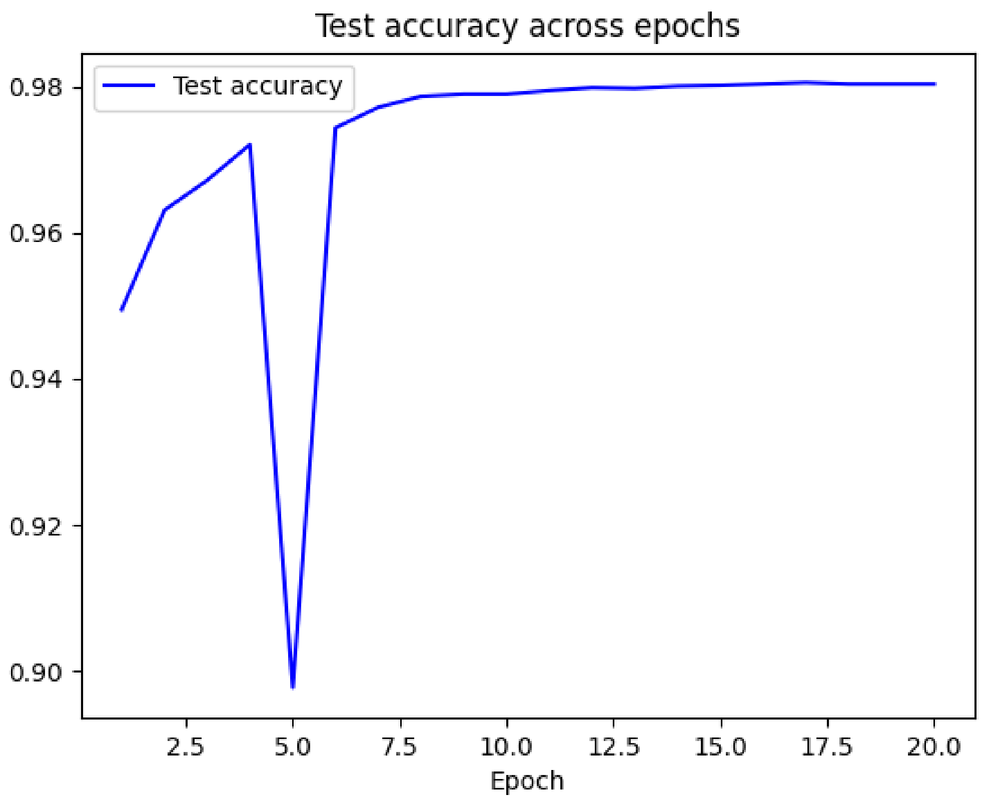


Figure 5: Test set accuracy of my best network

Notice the weird behaviour in epoch No.5. I don't have a concrete explanation for this behaviour, but I think that it stems from SGD overshooting the local minimum of the loss function by a little bit.