CS 189 Introduction to Machine Learning
Fall 2018 HW6

This homework is due **Friday, November 2nd, 2018 at 10pm**.

2 Stability and the Regularized Hinge Loss

In this problem, we are going to prove that minimizing the regularized hinge loss leads to a classifier with lower average prediction error. Throughout, let $\mathcal{D}$ denote a distribution over data-labels pairs $(\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, 1\}$, where, with probability 1, $\|\mathbf{x}\|_2 \leq 1$.

Ideally, we would want a predictor which minimizes the population misclassification error,

$$\mathcal{R}_{0,1}[\mathbf{w}] := \mathbf{P}_{(\mathbf{x},y)\sim\mathcal{D}}[\mathrm{sign}(\mathbf{w}^\top \mathbf{x}) \neq y].$$

If our data are not linearly speparable, it turns out that directly optimizing the $0-1$ loss $\mathbf{I}(\mathrm{sign}(\mathbf{w}^\top \mathbf{x}) \neq y)$ is computationally intractable (recall that $\mathbf{I}(a \neq b)$ is the function which is 1 if $a \neq b$ and 0 otherwise). Instead, we will optimize a regularized hinge loss,

$$\mathrm{hinge}_\lambda(\mathbf{w}, \mathbf{x}, y) := \max\{1 - y \cdot \mathbf{w}^\top \mathbf{x}, 0\} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2,$$

with $\|w\|_2 \leq R$. This is called an *surrogate loss* because it replaces the $0 - 1$ loss. Though surrogate losses help retain efficiency when the data are not separable, we will give an analysis of using the surrogate loss when the data are separable with margin.

Specifically, let $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ denote a dataset of n points drawn $i.i.d.$ from $\mathcal{D}$. We define the empirical and population risk

$$\mathcal{R}(\mathbf{w}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathrm{hinge}_\lambda(\mathbf{w}, \mathbf{x}, y)\right] ,$$
$$\mathcal{R}_{\mathcal{S}}(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} \mathrm{hinge}_\lambda(\mathbf{w}, \mathbf{x}_i, y_i) ,$$

and consider the *constrained ERM* estimator $\widehat{\mathbf{w}}$, for a fixed $\lambda > 0, R > 0$,

$$\widehat{\mathbf{w}}_{\mathcal{S}} = \arg \min_{\mathbf{w}:\|\mathbf{w}\|_2 \leq R} \mathcal{R}_{\mathcal{S}}(\mathbf{w}).$$

(a) **Show that** $\mathrm{hinge}_\lambda(\mathbf{w}, \mathbf{x}, y) \geq \mathrm{hinge}_0(\mathbf{w}, \mathbf{x}, y) \geq \mathbf{I}(\mathrm{sign}(\mathbf{w}^\top \mathbf{x}) \neq y)$**. Conclude that**

$$\mathbb{E}_{\mathcal{S}}[\mathcal{R}_{0,1}[\widehat{\mathbf{w}}_{\mathcal{S}}]] \leq \mathbb{E}_{\mathcal{S}}[\mathcal{R}(\widehat{\mathbf{w}}_{\mathcal{S}})] .$$

Solution: For the first inequality,

$$\mathrm{hinge}_\lambda(\mathbf{w}, \mathbf{x}, y) = \mathrm{hinge}_0(\mathbf{w}, \mathbf{x}, y) + \frac{\lambda}{2}\|w\|_2^2 \geq \mathrm{hinge}_0(\mathbf{w}, \mathbf{x}, y) .$$

For the second, note that if $\text{sign}(\mathbf{w}^\top \mathbf{x}) = y$, then

$$\max\{1 - y \cdot \mathbf{w}^\top \mathbf{x}, 0\} \geq 0 = \mathbf{I}(\text{sign}(\mathbf{w}^\top \mathbf{x}) \neq y),$$

whereas if $\text{sign}(\mathbf{w}^\top \mathbf{x}) \neq y$, then $y \cdot \mathbf{w}^\top \mathbf{x} \leq 0$, so

$$\max\{1 - y \cdot \mathbf{w}^\top \mathbf{x}, 0\} \geq 1 - y \cdot \mathbf{w}^\top \mathbf{x} \geq 1 = \mathbf{I}(\text{sign}(\mathbf{w}^\top \mathbf{x}) \neq y).$$

This proves $\text{hinge}_0(\mathbf{w}, \mathbf{x}, y) \geq \mathbf{I}(\text{sign}(\mathbf{w}^\top \mathbf{x}) \neq y)$. Therefore,

$$\begin{aligned} \mathbb{E}_{\mathcal{S}}[\mathcal{R}_{0,1}[\widehat{\mathbf{w}}_{\mathcal{S}}]] &= \mathbb{E}_{\mathcal{S}}[\mathbb{E}_{(\mathbf{x}',y')\sim\mathcal{D}}[\mathbf{I}(\text{sign}(\widehat{\mathbf{w}}_{\mathcal{S}}^\top \mathbf{x}') \neq y')]] \\ &\leq \mathbb{E}_{\mathcal{S}}[\mathbb{E}_{(\mathbf{x}',y')\sim\mathcal{D}}[\text{hinge}_\lambda(\widehat{\mathbf{w}}_{\mathcal{S}}, \mathbf{x}', y')]] = \mathbb{E}_{\mathcal{S}}[\mathcal{R}(\widehat{\mathbf{w}}_{\mathcal{S}})]. \end{aligned}$$

(b) Recall that a function $g(\mathbf{w})$ is L-Lipschitz on a ball of radius R, $\mathcal{B}_R := \{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\|_2 \leq R\}$ if

$$|g(\mathbf{w}_1) - g(\mathbf{w}_2)| \leq L\|\mathbf{w}_1 - \mathbf{w}_2\|_2, \qquad \text{for all } \mathbf{w}_1, \mathbf{w}_2 \in \mathcal{B}_R.$$

Furthermore, it is convex on $\mathcal{B}_R$ if

$$g(\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2) \leq \alpha g(\mathbf{w}_1) + (1-\alpha)g(\mathbf{w}_2), \qquad \text{for all } \mathbf{w}_1, \mathbf{w}_2 \in \mathcal{B}_R, \alpha \in [0,1].$$

Note that $\mathcal{B}_R$ is a convex set (by the triangle inequality), so $\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2 \in \mathcal{B}_R$ as long as $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{B}_R$. **Show that**

(a) if $f(\mathbf{w})$ is L_f-Lipschitz and $g(\mathbf{w})$ is L_g-Lipschitz on $\mathcal{B}_R$, then $h(\mathbf{w}) := \max\{f(\mathbf{w}), g(\mathbf{w})\}$ is $\max\{L_f, L_g\}$-Lipschitz on $\mathcal{B}_R$.

(b) If f and g are convex, then $h(\mathbf{w}) := \max\{f(\mathbf{w}), g(\mathbf{w})\}$ is convex.

Hint: Begin by showing that for any real numbers a, b, c, d,

(a) $|\max\{a,b\} - \max\{c,d\}| \leq \max\{|a-c|, |b-d|\}$

(b) $\max\{a+b, c+d\} \leq \max\{a,c\} + \max\{b,d\}$

Note that, by swapping out (a,b) for (c,d), you can assume without loss of generality that $\max\{a,b\} \geq \max\{c,d\}$ when trying to prove the first inequality. Similarly, you can assume without loss of generality that $\max\{a+b, c+d\} = a+b$ when trying to prove the second inequality.

Solution: Let's start with the hints. For the first hint, we may assume without loss of generality that $\max\{a,b\} \geq \max\{c,d\}$. Then,

$$\begin{aligned} |\max\{a,b\} - \max\{c,d\}| = |\max\{a,b\} - \max\{c,d\} &= \max\{a - \max\{c,d\}, b - \max\{c,d\}\} \\ &\leq \max\{a-c, b-d\} \leq \max\{|a-c|, |b-d|\}. \end{aligned}$$

For the next hint, again we may assume without loss of generality that $\max\{a+b, c+d\} = a+b$. Then,

$$\max\{a+b, c+d\} = a+b \leq \max\{a,c\} + b \leq \max\{a,c\} + \max\{b,d\}.$$

Having proved the hints, lets establish the first part of the problem: We have

$$\begin{aligned}|h(\mathbf{w}_1) - h(\mathbf{w}_2)| &= |\max\{f(\mathbf{w}_1), g(\mathbf{w}_1)\} - \max\{f(\mathbf{w}_2), g(w_2)\}| \\ &\overset{\text{first hint}}{\le} \max\{|f(\mathbf{w}_1) - f(\mathbf{w}_2)|, |g(\mathbf{w}_1) - g(\mathbf{w}_2)\} \\ &\overset{\text{Lipschitz def.}}{\le} \max\{L_f(\|\mathbf{w}_1 - \mathbf{w}_2\|_2, L_g\|\mathbf{w}_1 - \mathbf{w}_2\|_2\} \\ &= \max\{L_f, L_g\}\|\mathbf{w}_1 - \mathbf{w}_2\|_2.\end{aligned}$$

which proves h is L Lipschitz. To see that h is convex, we have

$$\begin{aligned}h(\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2) &= \max\{f(\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2), g(\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2)\} \\ &\overset{\text{convexity}}{\le} \max\{\alpha f(\mathbf{w}_1) + (1-\alpha)f(\mathbf{w}_2), \alpha g(\mathbf{w}_1) + (1-\alpha)g(\mathbf{w}_2))\} \\ &\overset{\text{second hint}}{\le} \max\{\alpha f(\mathbf{w}_1), \alpha g(\mathbf{w}_1)\} + \max\{(1-\alpha)f(\mathbf{w}_2), (1-\alpha)g(\mathbf{w}_2))\} \\ &= \alpha\max\{f(\mathbf{w}_1), g(\mathbf{w}_1)\} + (1-\alpha)\max\{f(\mathbf{w}_2), g(\mathbf{w}_2))\} = \alpha h(\mathbf{w}_1) + \alpha h(\mathbf{w}_2).\end{aligned}$$

(c) **Show that for any $\mathbf{x} \in \mathbb{R}^d$ and $y \in \{-1, 1\}$,**

  (a) $\text{hinge}_0(\mathbf{w}, \mathbf{x}, y)$ **is convex in $\mathbf{w}$. That is, the function $\mathbf{w} \mapsto \text{hinge}_0(\mathbf{w}, \mathbf{x}, y)$ is convex, i.e. for all $\alpha \in [0, 1]$,**

$$\text{hinge}_0(\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2, \mathbf{x}, y) \le \alpha\text{hinge}_0(\mathbf{w}_1, \mathbf{x}, y) + (1-\alpha)\text{hinge}_0(\mathbf{w}_2, \mathbf{x}, y)\ .$$

  (b) **The function $\text{hinge}_0(\mathbf{w}, \mathbf{x}, y)$ is $\|\mathbf{x}\|_2$-Lipschitz in $\mathbf{w}$ on $\mathbb{R}^d$. That is, the function $\mathbf{w} \mapsto \text{hinge}_0(\mathbf{w}, \mathbf{x}, y)$ satisfies**

$$|\text{hinge}_0(\mathbf{w}_1, \mathbf{x}, y) - \text{hinge}_0(\mathbf{w}_2, \mathbf{x}, y)| \le \|\mathbf{x}\|_2\|\mathbf{w}_1 - \mathbf{w}_2\|_2.$$

Hint: Use part (b).

Solution: Let $f(\mathbf{w}) = 0$ and let $g(\mathbf{w}) = 1 - y \cdot \mathbf{w}^\top\mathbf{x}$. f and g are affine functions, so they are convex. Moreover, f is 0-Lipschitz, and $|g(\mathbf{w}_1) - g(\mathbf{w}_2)| = |y(\mathbf{w}_1 - \mathbf{w}_2)^\top\mathbf{x}| \le \|\mathbf{x}\|_2\|\mathbf{w}_1 - \mathbf{w}_2\|_2$ by Cauchy Schwartz, so g is $\|\mathbf{x}\|_2$-Lipschitz. Since $\text{hinge}_0(\mathbf{w}, \mathbf{x}, y) = \max\{f(\mathbf{w}), g(\mathbf{w})\}$, the answer follows.

(d) **Show that for any $\mathbf{x}, y$ such that $y \in \{-1, 1\}$ and $\|\mathbf{x}\|_2 \le 1$, the function $\mathbf{w} \mapsto \text{hinge}_\lambda(\mathbf{w}, \mathbf{x}, y)$ is $L = (1 + 2\lambda R)$-Lipschitz on the ball of radius R, $\mathcal{B}_R$. That is, for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{B}_R$,**

$$|\text{hinge}_\lambda(\mathbf{w}_1, \mathbf{x}, y) - \text{hinge}_\lambda(\mathbf{w}_2, \mathbf{x}, y)| \le (1 + 2\lambda R)\|\mathbf{w}_1 - \mathbf{w}_2\|_2.$$

Hint: You may need to bound $|\|\mathbf{w}_1\|_2^2 - \|\mathbf{w}_2\|_2^2|$. Try writing $\|\mathbf{w}_1\|_2^2 = \|\mathbf{w}_2 + (\mathbf{w}_1 - \mathbf{w}_2)\|_2^2$, and then expanding out $\|\mathbf{w}_2 + (\mathbf{w}_1 - \mathbf{w}_2)\|_2^2$. Ultimately, you should try to establish the following inequality:

$$|\|\mathbf{w}_1\|_2^2 - \|\mathbf{w}_2\|_2^2| \le \|\mathbf{w}_1 - \mathbf{w}_2\|_2(2\|\mathbf{w}_2\|_2 + \|\mathbf{w}_1 - \mathbf{w}_2\|_2).$$

Now think about how to use the condition that $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{B}_R$.

Solution: Fix an $\mathbf{x}$ and y with $y \in \{-1, 1\}$ and $\|\mathbf{x}\|_2 \leq 1$. Then

$$\begin{aligned}|\text{hinge}_\lambda(\mathbf{w}_1, \mathbf{x}, y) - \text{hinge}_\lambda(\mathbf{w}_2, \mathbf{x}, y)| &\leq |\text{hinge}_0(\mathbf{w}_1, \mathbf{x}, y) - \text{hinge}_0(\mathbf{w}_2, \mathbf{x}, y)| + |\frac{\lambda}{2}\|\mathbf{w}_1\|_2^2 - \frac{\lambda}{2}\|\mathbf{w}_2\|_2^2| \\ &= |\text{hinge}_0(\mathbf{w}_1, \mathbf{x}, y) - \text{hinge}_0(\mathbf{w}_2, \mathbf{x}, y)| + \frac{\lambda}{2}|\|\mathbf{w}_1\|_2^2 - \|\mathbf{w}_2\|_2^2| \\ &= \|\mathbf{w}_1 - \mathbf{w}_2\|_2 + \frac{\lambda}{2}|\|\mathbf{w}_1\|_2^2 - \|\mathbf{w}_2\|_2^2|,\end{aligned}$$

where the last inequality is by part a. Next, using the hint, we write

$$\|\mathbf{w}_1\|_2^2 = \|\mathbf{w}_2 + (\mathbf{w}_1 - \mathbf{w}_2)\|_2^2 = \|\mathbf{w}_2\|_2^2 + 2\langle \mathbf{w}_1 - \mathbf{w}_2, \mathbf{w}_2\rangle + \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2$$

Hence,

$$\begin{aligned}|\|\mathbf{w}_1\|_2^2 - \|\mathbf{w}_2\|_2^2| &= |\|\mathbf{w}_1\|_2^2 - \left(\|\mathbf{w}_2\|_2^2 + 2\langle \mathbf{w}_1 - \mathbf{w}_2, \mathbf{w}_2\rangle + \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2\right)| \\ &= |-2\langle \mathbf{w}_1 - \mathbf{w}_2, \mathbf{w}_2\rangle - \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2| \\ &\leq 2\|\mathbf{w}_1 - \mathbf{w}_2\|_2\|\mathbf{w}_2\|_2 + \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2 \\ &= \|\mathbf{w}_1 - \mathbf{w}_2\|_2(2\|\mathbf{w}_2\|_2 + \|\mathbf{w}_1 - \mathbf{w}_2\|_2).\end{aligned}$$

On $\mathcal{B}_R$, we have $\|\mathbf{w}_2\|_2 \leq R$ and $\|\mathbf{w}_1 - \mathbf{w}_2\|_2 \leq 2R$. Hence, the above is at most $4\|\mathbf{w}_1 - \mathbf{w}_2\|_2$. Combining with the bound above, we have

$$\begin{aligned}|\text{hinge}_\lambda(\mathbf{w}_1, \mathbf{x}, y) - \text{hinge}_\lambda(\mathbf{w}_2, \mathbf{x}, y)| &\leq \|\mathbf{w}_1 - \mathbf{w}_2\|_2 + \frac{\lambda}{2}|\|\mathbf{w}_1\|_2^2 - \|\mathbf{w}_2\|_2^2| \\ &\leq \|\mathbf{w}_1 - \mathbf{w}_2\|_2 + 2\lambda R\|\mathbf{w}_1 - \mathbf{w}_2\|_2 = (1 + 2\lambda R)\|\mathbf{w}_1 - \mathbf{w}_2\|_2.\end{aligned}$$

(e) In the slides from EE227c, we linked to notes which prove a theorem which links convexity to uniform stability. You do not need to prove the theorem (if you are curious, read the proof at `https://ee227c.github.io/notes/ee227c-lecture11.pdf`). The theorem is as follows:

Theorem 1. *Let $\mathcal{D}$ be a distribution over data-label pairs $(\mathbf{x}, y)$. Moreover, suppose that if $(\mathbf{x}, y)$ is drawn from $\mathcal{D}$, the loss functions $\mathbf{w} \mapsto \ell(\mathbf{w}, \mathbf{x}, y)$ are always (i.e. with probability one) L-Lipschitz and α-strongly convex in w for all $\mathbf{w} \in \mathcal{B}_R$. Then, the constrained empirical risk minimizer, defined as*

$$\widehat{\mathbf{w}} := \arg\min_{\mathbf{w} \in \mathcal{B}_R} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}, \mathbf{x}_i, y_i)$$

always (i.e. with probability 1) has uniform stability bounded by $\Delta_{\text{sup}} \leq \frac{4L^2}{\alpha n}$, where Δ_{sup} is the uniform stability bound defined in the lecture slides.

Use this bound, the relationship between uniform stabililty, average stability and generalization error, and the properties of $\operatorname{hinge}_\lambda$ derived above, to **show that for the estimator $\widehat{\mathbf{w}}_{\mathcal{S}}$ defined above, we have**

$$\mathbb{E}_{\mathcal{S}}\left[\mathcal{R}(\widehat{\mathbf{w}}_{\mathcal{S}}) - \mathcal{R}_{\mathcal{S}}(\widehat{\mathbf{w}}_{\mathcal{S}})\right] \le \frac{1}{n}\left(\frac{8}{\lambda} + 32\lambda R^2\right)$$

Hint: Recall that $\mathbf{w} \mapsto \ell(\mathbf{w}, \mathbf{x}, y)$ is α strongly convex if $\mathbf{w} \mapsto \ell(\mathbf{w}, \mathbf{x}, y) - \frac{\alpha}{2}\|\mathbf{w}\|_2^2$ is convex. You may also want the inequality $(a+b)^2 \le 2(a^2+b^2)$.

Solution: From lecture, we saw that the expected generalization error is bounded by the uniform stability

$$\mathbb{E}_{\mathcal{S}}\left[\mathcal{R}(\widehat{\mathbf{w}}_{\mathcal{S}}) - \mathcal{R}_{\mathcal{S}}(\widehat{\mathbf{w}}_{\mathcal{S}})\right] \le \Delta_{\text{sup}}$$

Since $\operatorname{hinge}_0(\mathbf{w}, \mathbf{x}, y)$ is convex on $\mathbb{R}^d$, $\mathbf{w} \mapsto \operatorname{hinge}_\lambda(\mathbf{w}, \mathbf{x}, y)$ is λ strongly convex on $\mathbb{R}^d \subset \mathcal{B}_R$. Moreover, $\mathbf{w} \mapsto \operatorname{hinge}_\lambda(\mathbf{w}, \mathbf{x}, y)$ is $(1+2\lambda R)$-Lipschitz over $\mathcal{B}_R$. Therefore, by the given theorem, we have that

$$\Delta_{\text{sup}} \le \frac{4((1+2\lambda R)^2}{\lambda} \le \frac{8(1+4\lambda^2 R^2)}{\lambda} \le \frac{8}{\lambda n} + \frac{32\lambda R^2}{n}$$

(f) Suppose that there exists an exact classifier with margin b. That is, there exists a w_* such that

$$\mathbf{P}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\frac{y \cdot \mathbf{w}_*^\top \mathbf{x}}{\|\mathbf{w}_*\|_2} \ge b\right] = 1.$$

Show that by choosing R appropriately as a function of b, that

$$\mathbb{E}_{\mathcal{S}}[\mathcal{R}_{\mathcal{S}}(\widehat{\mathbf{w}}_{\mathcal{S}})] \le \frac{\lambda}{2b^2}.$$

Conclude that for this value of R, we have the following bound on the $0, 1$ loss

$$\mathbb{E}_{\mathcal{S}}[\mathcal{R}_{0,1}(\widehat{\mathbf{w}}_{\mathcal{S}})] \le C\left(\frac{1}{\lambda n} + \frac{\lambda}{b^2}\right),$$

where C is a universal constant that does not depend on b, n, λ. Finally, what λ would you choose to optimize the right handside of the above equation? What do you get?

Hint: For the conclusion, will may want to remember what you proved in part (a).

Solution: If there exists a classifier $\mathbf{w}_*$ with margin b, then by rescaling, we may assume that $\|\mathbf{w}_*\|_2 = 1/b$ and $y \cdot \mathbf{w}_*^\top \mathbf{x} \ge 1$ with probability 1 over all $(\mathbf{x}, y) \sim \mathcal{D}$. Hence, for this $\mathbf{w}_*$, we have $\operatorname{hinge}_\lambda(\mathbf{w}_*, \mathbf{x}, y) = \max\{1 - y \cdot \mathbf{w}^\top \mathbf{x}, 0\} + \frac{\lambda}{2}\|\mathbf{w}_*\|_2^2 = \max\{1-1, 0\} = 0 + \frac{\lambda}{2b^2}$. Therefore, we always have

$$\mathcal{R}_{\mathcal{S}}(\mathbf{w}_*) = \frac{1}{n}\sum_{i=1}^{n} \operatorname{hinge}(\mathbf{w}_*, \mathbf{x}_i, y_i) \le \frac{1}{n}\sum_{i=1}^{n} \frac{\lambda}{2b^2} = \frac{\lambda}{2b^2}.$$

Now, set $R = 1/b$. Therefore, since $\|\mathbf{w}\|_* \leq \frac{1}{b}$

$$\begin{aligned}\mathbb{E}_{\mathcal{S}}[\mathcal{R}_{\mathcal{S}}(\widehat{\mathbf{w}}_{\mathcal{S}})] &= \mathbb{E}_{\mathcal{S}}\left[\min_{\mathbf{w}:\|\mathbf{w}\|\leq\frac{1}{b}} \mathcal{R}_{\mathcal{S}}(w)\right] \\ &\leq \mathbb{E}_{\mathcal{S}}[\mathcal{R}_{\mathcal{S}}(\mathbf{w}_*)] \leq \frac{\lambda}{2b^2}.\end{aligned}$$

Substituing in $R = 1/b$ to the previous subproblem, we have

$$\begin{aligned}\mathbb{E}_{\mathcal{S}}[\mathcal{R}(\widehat{\mathbf{w}}_{\mathcal{S}})] &= \mathbb{E}_{\mathcal{S}}\left[\mathcal{R}(\widehat{\mathbf{w}}_{\mathcal{S}}) - \mathcal{R}_{\mathcal{S}}(\widehat{\mathbf{w}}_{\mathcal{S}})\right] + \mathbb{E}_{\mathcal{S}}[\mathcal{R}_{\mathcal{S}}(\widehat{\mathbf{w}}_{\mathcal{S}})] \\ &\leq \frac{8}{\lambda n} + \frac{32\lambda}{nb^2} + \frac{\lambda}{2b^2} \leq C\left(\frac{1}{\lambda n} + \frac{\lambda}{b^2}\right),\end{aligned}$$

where we remember that $1/n \leq 1$. Finally, from part a,

$$\mathbb{E}_{\mathcal{S}}[\mathcal{R}_{0,1}(\widehat{\mathbf{w}}_{\mathcal{S}})] \leq \mathbb{E}_{\mathcal{S}}[\mathcal{R}(\widehat{\mathbf{w}}_{\mathcal{S}})] = C\left(\frac{1}{\lambda n} + \frac{\lambda}{b^2}\right)$$

To conclude, choose $\lambda = b/\sqrt{n}$. This gives an error of $C\frac{1}{b\sqrt{n}}$.

3 Non-Convex Optimization

In this question, we will look at the importance of random initialization for gradient descent on non-convex problems. We will look one of the simplest non-trivial non-convex problems we know, which is the familiar *principal component analysis* problem. Instead of using SVD, however, we will consider the use of gradient descent to solve PCA.

Specifically, let $\mathbf{M} \in \mathbb{R}^{d\times d}$ be a symmetric positive semi-definite matrix with rank $r \leq d$. Let the non-zero eigenvalues of $\mathbf{M}$ be distinct, i.e. $\lambda_1 > \lambda_2 > ... > \lambda_r > 0$. Define $f : \mathbb{R}^d \to \mathbb{R}$ as:

$$f(\mathbf{w}) = \frac{1}{4}\left\|\mathbf{M} - \mathbf{w}\mathbf{w}^\top\right\|_F^2 .$$

Let $\mathbf{M} = \sum_{i=1}^r \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$ denote the eigendecomposition of $\mathbf{M}$. Now, consider the optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) .$$

From our study of PCA, we know the optimal solution to this problem is to set $\mathbf{w} \in \{\pm\sqrt{\lambda_1}\mathbf{u}_1\}$. Let us consider what happens when we instead apply gradient descent to solve this optimization problem:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t \nabla f(\mathbf{w}_t) .$$

(a) Show that $f(\mathbf{w})$ is ***not*** convex.

Hint: Consider the line between $\sqrt{\lambda_1}\mathbf{u}_1$ and $-\sqrt{\lambda_1}\mathbf{u}_1$.

Solution:

Solution 1. Consider two points $\mathbf{w}_1 = \sqrt{\lambda_1}\mathbf{u}_1$ and $\mathbf{w}_2 = -\sqrt{\lambda_1}\mathbf{u}_1$. We have that $f(\mathbf{w}_1) = f(\mathbf{w}_2) = \frac{1}{4}\sum_{i=2}^{d}\lambda_i^2$. However, the midpoint between $\mathbf{w}_1$ and $\mathbf{w}_2$ is zero, and we have $f(\mathbf{0}) = \frac{1}{4}\sum_{i=1}^{d}\lambda_i^2 > f(\mathbf{w}_1)$, since $\lambda_1 > 0$. That is, we have that $f(\mathbf{0}) > f(\mathbf{w}_1) = \frac{1}{2}f(\mathbf{w}_1) + \frac{1}{2}f(\mathbf{w}_2)$ which violates convexity.

Solution 2. The second solution involves computing the Hessian and checking that it is not PSD. We compute the Hessian to be $\nabla^2 f(\mathbf{w}) = \|\mathbf{w}\|_2^2\mathbf{I}_d + \mathbf{w}\mathbf{w}^\top + (\mathbf{w}\mathbf{w}^\top - \mathbf{M})$. Then we see that $\nabla^2 f(\mathbf{0}) = -\mathbf{M}$ which is not PSD (since $\mathbf{M} \succeq 0$ and $\mathbf{M} \neq 0$), and hence f is not convex.

(b) Compute the gradient $\nabla f(\mathbf{w})$.

Hint: There are two ways to approach this, which are both valid. The first way is to write $f(\mathbf{w})$ in terms of the sum of quantities for which it is either easy to compute the gradients or we already know the answer. The second way is from first principles, by finding the vector $\nabla f(\mathbf{w}) \in \mathbb{R}^d$ such that $f(\mathbf{w} + \mathbf{\Delta}) = f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{\Delta}\rangle + o(\|\mathbf{\Delta}\|_2)$ holds.

Solution: The gradient is $\nabla f(\mathbf{w}) = (\mathbf{w}\mathbf{w}^\top - \mathbf{M})\mathbf{w}$.

We show a solution based on the first approach. We define $g(\mathbf{w}) = \left\|\mathbf{M} - \mathbf{w}\mathbf{w}^\top\right\|_F^2$ (without the constant $1/4$). We can expand the norm as follows:

$$\begin{aligned} g(\mathbf{w}) &= \|\mathbf{M}\|_F^2 + \left\|\mathbf{w}\mathbf{w}^\top\right\|_F^2 - 2\mathbf{w}^\top\mathbf{M}\mathbf{w} \\ &= \|\mathbf{M}\|_F^2 + \|\mathbf{w}\|_2^4 - 2\mathbf{w}^\top\mathbf{M}\mathbf{w}\,. \end{aligned}$$

We now observe that:

$$\begin{aligned} \nabla_{\mathbf{w}}\|\mathbf{w}\|_2^4 &= 4\|\mathbf{w}\|_2^2\mathbf{w} = 4\mathbf{w}\mathbf{w}^\top\mathbf{w}\,, \\ \nabla_{\mathbf{w}}\mathbf{w}^\top\mathbf{M}\mathbf{w} &= 2\mathbf{M}\mathbf{w}\,. \end{aligned}$$

Therefore,

$$\nabla g(\mathbf{w}) = 4\mathbf{w}\mathbf{w}^\top\mathbf{w} - 4\mathbf{M}\mathbf{w} = 4(\mathbf{w}\mathbf{w}^\top - \mathbf{M})\mathbf{w}\,.$$

Hence $\nabla f(\mathbf{w}) = (\mathbf{w}\mathbf{w}^\top - \mathbf{M})\mathbf{w}$ as claimed.

(c) Recall that a stationary point of f is a point $\mathbf{w}$ such that $\nabla f(\mathbf{w}) = \mathbf{0}$. In this question we will show that the set of stationary points of f is given by the zero vector and scaled eigenvectors of $\mathbf{M}$. Specifically, show that:

$$\{\mathbf{w} \in \mathbb{R}^d : \nabla f(\mathbf{w}) = \mathbf{0}\} = \{\mathbf{0}\} \cup \{\pm\sqrt{\lambda_i}\mathbf{u}_i \;:\; i = 1, ..., r\}\,.$$

Solution: A stationary point is any $\mathbf{w}$ that solves the non-linear equation $(\mathbf{w}\mathbf{w}^\top - \mathbf{M})\mathbf{w} = \mathbf{0}$. From this, it is immediate that $\mathbf{w} = \mathbf{0}$ is a stationary point.

For the remaining stationary points, writing $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$ where $\mathbf{U} \in \mathbb{R}^{d\times d}$, we have

$$(\mathbf{w}\mathbf{w}^\top - \mathbf{M})\mathbf{w} = (\mathbf{U}\mathbf{U}^\top\mathbf{w}\mathbf{w}^\top\mathbf{U}\mathbf{U}^\top - \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top)\mathbf{w}$$

$$= \mathbf{U}(\mathbf{U}^\top \mathbf{w}\mathbf{w}^\top \mathbf{U} - \mathbf{\Sigma})\mathbf{U}^\top \mathbf{w} \,.$$

Define the invertible variable transformation $\mathbf{z} := \mathbf{U}^\top \mathbf{w}$. Setting the RHS equal to zero and left multiplying by $\mathbf{U}^\top$, we conclude that:

$$(\mathbf{z}\mathbf{z}^\top - \mathbf{\Sigma})\mathbf{z} = \mathbf{0} \Longleftrightarrow \|\mathbf{z}\|_2^2 \mathbf{z} = \mathbf{\Sigma}\mathbf{z} \,. \tag{1}$$

We now study the non-zero solutions to RHS eigenvector equation (1). For every $i = 1, ..., r$, plugging in $\alpha_i \mathbf{e}_i$ into this equation, we have that it simplifies to:

$$\alpha_i^2 \mathbf{e}_i = \lambda_i \mathbf{e}_i \Longrightarrow \alpha_i = \pm\sqrt{\lambda_i} \,.$$

Hence the points $\{\pm\sqrt{\lambda_i}\mathbf{e}_i : i = 1, ..., r\}$ are non-zero solutions to equation (1). Furthermore, it is clear that the only $\alpha_i \mathbf{e}_i$ that solves (1) for $i > r$ is when $\alpha_i = 0$. Hence, the points $\{\pm\sqrt{\lambda_i}\mathbf{e}_i : i = 1, ..., r\}$ are the **only** non-zero solutions to (1) that have exactly one non-zero coordinate.

To conclude the proof, we need to show that there are no non-zero solutions with two or greater non-zero coordinates. Suppose that $\mathbf{z} \neq 0$ solves (1), and there are two distinct indices i_1, i_2 satisfies $\mathbf{z}_{i_1} \neq 0$ and $\mathbf{z}_{i_2} \neq 0$. Then (1) implies that:

$$\|\mathbf{z}\|_2^2 \mathbf{z}_{i_1} = \lambda_{i_1} \mathbf{z}_{i_1} \,, \quad \|\mathbf{z}\|_2^2 \mathbf{z}_{i_2} = \lambda_{i_2} \mathbf{z}_{i_2} \,.$$

Since both $\mathbf{z}_{i_1}$ and $\mathbf{z}_{i_2}$ are non-zero, we can divide by $\mathbf{z}_{i_1}$ (resp. $\mathbf{z}_{i_2}$) and obtain:

$$\|\mathbf{z}\|_2^2 = \lambda_{i_1} \,, \quad \|\mathbf{z}\|_2^2 = \lambda_{i_2} \,.$$

If either $i_1 > r$ or $i_2 > r$ then we have a contradiction that $\mathbf{z} = 0$, so we can assume that $i_1 \leq r$ and $i_2 \leq r$. However, since $i_1 \neq i_2$, this is also a contradiction since $\|\mathbf{z}\|_2^2 = \lambda_{i_1} \neq \lambda_{i_2} = \|\mathbf{z}\|_2^2$ (here we made crucial use of the distinctness assumption of the eigenvalues given in the problem setup). Hence all non-zero solutions to (1) have exactly one non-zero coordinate, and we enumerated this set previously. This concludes the proof.

(d) Show that for any $i = 2, ..., r$, we have that $\mathbf{w}_i = \pm\sqrt{\lambda_i}\mathbf{u}_i$ is a saddle point of f. Recall that a saddle point is a stationary point that is not a local extremum of the function.

Hint: For a unit vector $\mathbf{v} \in \mathbb{R}^d$, define the function $g(t; \mathbf{v}) = f(\mathbf{w}_i + t\mathbf{v})$. Find two directions $\mathbf{v}_1, \mathbf{v}_2$ such that $\frac{d^2}{dt^2} g(t; \mathbf{v}_1)\Big|_{t=0} < 0$ and $\frac{d^2}{dt^2} g(t; \mathbf{v}_2)\Big|_{t=0} > 0$. That is, the point $\mathbf{w}_i$ is a local minimum along the direction $\mathbf{v}_2$, but there exists a direction $\mathbf{v}_1$ for which it the function at $\mathbf{w}_i$ can be decreased.

Another proof involves computing the Hessian $\nabla^2 f(\mathbf{w}_i)$ and showing that it has a positive and negative eigenvalue.

Solution:

We assume wlog that $\mathbf{w}_i = \sqrt{\lambda_i}\mathbf{u}_i$; the claim for $\mathbf{w}_i = -\sqrt{\lambda_i}\mathbf{u}_i$ holds by symmetry.

We first study the direction for which $\mathbf{w}_i$ is a local minimum. We will show that $\mathbf{u}_1$ is this direction, by studying $g(t; \mathbf{u}_i)$. We compute:

$$g(t; \mathbf{u}_i) = \frac{1}{4}\left\| \mathbf{M} - (\sqrt{\lambda_i} + t)^2 \mathbf{u}_i \mathbf{u}_i^\top \right\|_F^2$$

$$= \frac{1}{4}(\|\mathbf{M}\|_F^2 + (\sqrt{\lambda_i}+t)^4 - 2(\sqrt{\lambda_i}+t)^2\mathbf{u}_i^\top \mathbf{M}\mathbf{u}_i)$$
$$= \frac{1}{4}(\|\mathbf{M}\|_F^2 + (\sqrt{\lambda_i}+t)^4 - 2(\sqrt{\lambda_i}+t)^2\lambda_i)\,.$$

Differentiating $g(t;\mathbf{u}_i)$ twice yields:

$$\frac{d^2}{dt^2}g(t;\mathbf{u}_i) = 3(\sqrt{\lambda_i}+t)^2 - \lambda_i\,.$$

Evaluating this at $t=0$ yields:

$$\frac{d^2}{dt^2}g(t;\mathbf{u}_i)\bigg|_{t=0} = 2\lambda_i > 0\,.$$

This shows the direction of local minimum.

Now we show the descent direction. We will show that the direction $\mathbf{u}_1$ is a descent direction. First, we have the intermediate calculations:

$$\left\|(t\mathbf{u}_1 + \sqrt{\lambda_i}\mathbf{u}_i)(t\mathbf{u}_1 + \sqrt{\lambda_i}\mathbf{u}_i)^\top\right\|_2^4 = (t^2+\lambda_i)^2\,,$$
$$(t\mathbf{u}_1 + \sqrt{\lambda_i}\mathbf{u}_i)^\top \mathbf{M}(t\mathbf{u}_1 + \sqrt{\lambda_i}\mathbf{u}_i) = t^2\lambda_1 + \lambda_i^2\,.$$

We have:

$$g(t;\mathbf{u}_1) = \frac{1}{4}\left\|\mathbf{M} - (t\mathbf{u}_1 + \sqrt{\lambda_i}\mathbf{u}_i)(t\mathbf{u}_1 + \sqrt{\lambda_i}\mathbf{u}_i)^\top\right\|_F^2$$
$$= \frac{1}{4}(\|\mathbf{M}\|_F^2 + (t^2+\lambda_i)^2 - 2(t^2\lambda_1 + \lambda_i^2))\,.$$

Differentiating $g(t;\mathbf{u}_1)$ twice yields:

$$\frac{d^2}{dt^2}g(t;\mathbf{u}_1) = 3t^2 + \lambda_i - \lambda_1\,.$$

Evaluating this at $t=0$ yields:

$$\frac{d^2}{dt^2}g(t;\mathbf{u}_1)\bigg|_{t=0} = \lambda_i - \lambda_1 < 0\,.$$

(e) Show that for any step sizes α_t and for any index set $\mathcal{I} \subset \{1,...,d\}$, that if $\mathbf{w}_0 \in \text{span}\{\mathbf{u}_i\}_{i\in\mathcal{I}}$, then $\mathbf{w}_t \in \text{span}\{\mathbf{u}_i\}_{i\in\mathcal{I}}$ for all $t \geq 0$.

Solution: The gradient descent update is:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t(\mathbf{w}_t\mathbf{w}_t^\top - \mathbf{M})\mathbf{w}_t$$
$$= (1-\alpha_t\|\mathbf{w}_t\|_2^2)\mathbf{w}_t + \alpha_t\mathbf{M}\mathbf{w}_t\,.$$

Since $\text{span}\{\mathbf{u}_i\}_{i\in\mathcal{I}}$ is an invariant subspace of $\mathbf{M}$, we have that $\mathbf{M}\mathbf{w}_t \in \text{span}\{\mathbf{u}_i\}_{i\in\mathcal{I}}$. The claim now follows by induction.

(f) In light of the previous part, how would you initialize gradient descent (i.e. how would you set $\mathbf{w}_0$) so that gradient descent does not converge to the incorrect answer? You do not need to prove your initialization works, just provide an explanation based on your intuition.

Hint: consider a random initialization.

Solution: Uniform on the sphere or isotropic Gaussian.

4 Backpropagation Algorithm for Neural Networks

In this problem, we will be implementing the backpropagation algorithm to train a neural network to classify the difference between two handwritten digits (specifically the digits 3 and 9).

Before we start we will install the library mnist so we can load our data properly.
Run `pip install mnist` in your terminal so the library is properly installed.
Please attach screenshots of all your code in your code and terminal outputs in your submission!

To establish notation for this problem, we define:

$$\mathbf{a}_{i+1} = \sigma(\mathbf{z}_i) = \sigma(\mathbf{W}_i\mathbf{a}_i + \mathbf{b}_i).$$

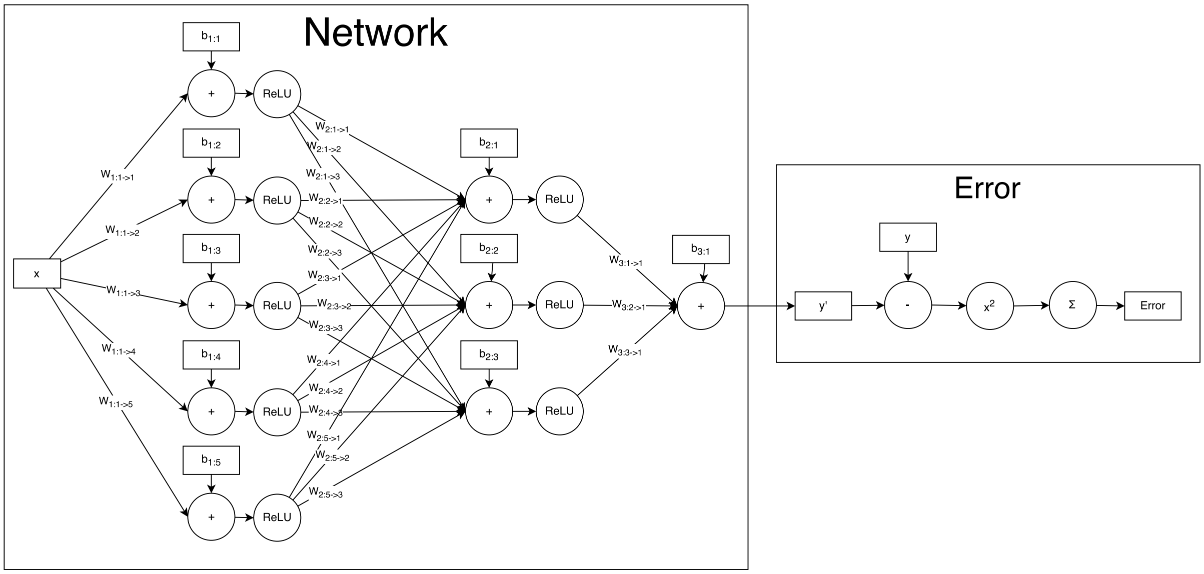
In this equation, $\mathbf{W}_i$ is a $n_{i+1} \times m_i$ matrix that maps the input $\mathbf{a}_i$ of dimension m_i to a vector of dimension n_{i+1}, where n_{i+1} is the size of layer $i+1$ and we have that $m_i = n_i$. The vector $\mathbf{b}_i$ is the bias vector added after the matrix multiplication, and σ is the nonlinear function applied element-wise to the result of the matrix multiplication and addition. $\mathbf{z}_i = \mathbf{W}_i\mathbf{a}_i + \mathbf{b}_i$ is a shorthand for the intermediate result within layer i before applying the nonlinear activation function σ. Each layer is computed sequentially where the output of one layer is used as the input to the next. To compute the derivatives with respect to the weights $\mathbf{W}_i$ and the biases $\mathbf{b}_i$ of each layer, we use the chain rule starting with the output of the network and work our way backwards through the layers, which is where the backprop algorithm gets its name.

You are given starter code with incomplete function implementations. For this problem, you will fill in the missing code so that we can train a neural network to learn your nonlinear classifier. The code currently trains a network with one hidden layer with 4 nodes.

(a) **Start by drawing a small example network with one hidden layer, where the last layer has a single scalar output. The first layer should have 784 inputs corresponding to the input image x which consists of a 28×28 pixels. The computational layers should have widths of 16, and 1 respectively. The final "output" layer's "nonlinearity" should be a linear unit that just returns its input. The earlier "hidden" layers should have ReLU units. Label all the n_i and m_i as well as all the a_i and $\mathbf{W}_i$ and b_i weights. You can consider the bias terms to be weights connected to a dummy unit whose output is always 1 for the purpose of labeling. You can also draw and label the loss function that will be important during training — use a squared-error loss.**

Here, the important thing is for you to understand your own clear ways to illustrate neural nets. You can follow conventions seen online or in lecture or discussion, or you can modify those conventions to pick something that makes the most sense to you. The important thing is to have your illustration be unambiguous so you can use it to help understand the forward flow of information during evaluation and the backward flow during gradient computations. Since you're going to be implementing all this during this problem, it is good to be clear.

Solution: The following diagram is a sample of what you could have drawn for your answer. Here, the weights a marked as $W_{i:j->k}$ which indicates the weight in layer i going from the jth input to the kth output. The weights are drawn along the lines connecting graph nodes to indicate multiplication along the connection. Operations are indicated with circles, and model parameters and inputs are indicated with rectangles.



(b) Let's start by implementing the cost function of the network. We will be using a simple least squares classifier. We will regress all the positive classes to 1, and the negative classes to -1. The sign of the predicted value will be the predicted class label. This function is used to assign an error for each prediction made by the network during training.

The error we actually care about is the misclassification error (MCE) which will be:

$$\text{MCE}(\hat{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{I}\{\text{sign}(y_i) \neq \text{sign}(\hat{y}_i)\}\ .$$

However this function is hard to to optimize so the implementation will be optimizing the mean squared error cost (MSE) function, which is given by

$$\text{MSE}(\hat{\mathbf{y}}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the observation that we want the neural network to output and $\hat{y}_i$ is the prediction from the network.

Write the derivative of the mean squared error cost function with respect to the predicted outputs $\hat{\mathbf{y}}$. In `backprop.py` implement the functions `QuadraticCost.fx` and `QuadraticCost.dx`

Solution: The derivative is given by

$$\begin{aligned}\frac{\partial \text{MSE}}{\partial \hat{\mathbf{y}}} &= 2\frac{1}{2}(-1)(\mathbf{y} - \hat{\mathbf{y}}) \\ &= \hat{\mathbf{y}} - \mathbf{y}\end{aligned}$$

Please see the solution code for the implementation.

(c) Now, let's take the derivatives of the nonlinear activation functions used in the network. **Implement the following nonlinear functions in the code and their derivatives:**

$$\sigma_{\text{linear}}(z) = z$$

$$\sigma_{\text{ReLU}}(z) = \begin{cases} 0 & z < 0 \\ z & \text{otherwise} \end{cases}$$

$$\sigma_{\tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

For the `tanh` function, feel free to use the `tanh` function in `numpy`. We have provided the sigmoid function as an example activation function.

Solution:

$$\frac{\partial \sigma_{\text{linear}}}{\partial z} = 1$$

$$\frac{\partial \sigma_{\text{ReLU}}}{\partial z} = \begin{cases} 0 & z < 0 \\ 1 & \text{otherwise} \end{cases}$$

$$\frac{\partial \sigma_{\tanh}}{\partial z} = 1 - \sigma_{\tanh}(z)^2$$

Note in particular for $\sigma_{\tanh}$ there are multiple valid ways to express its derivative.

Please see the code for the implementation.

(d) We have implemented the forward propagation part of the network for you (see `Model.evaluate` in the code). We now need to compute the derivative of the cost function with respect to the weights $\mathbf{W}$ and the biases $\mathbf{b}$ of each layer in the network. We will be using all of the code we previously implemented to help us compute these gradients. **Assume that $\frac{\partial \text{MSE}}{\partial \mathbf{a}_{i+1}}$ is given, where $\mathbf{a}_{i+1}$ is the input to layer $i+1$. Write the expression for $\frac{\partial \text{MSE}}{\partial \mathbf{a}_i}$ in terms of $\frac{\partial \text{MSE}}{\partial \mathbf{a}_{i+1}}$.**

Then implement these derivative calcualtions in the function `Model.compute_grad`. Recall, $\mathbf{a}_{i+1}$ is given by

$$\mathbf{a}_{i+1} = \sigma(\mathbf{z}_i) = \sigma(\mathbf{W}_i \mathbf{a}_i + \mathbf{b}_i) \ .$$

Solution: We start with looking at the formula that takes the activations from layer i and computes the activations for layer $i+1$:

$$\mathbf{a}_{i+1} = \sigma(W_i \mathbf{a}_i + \mathbf{b}_i).$$

We know that the error is computed as a composition of the layers in the network, so we can use the chain rule to help us compute $\frac{\partial \text{MSE}}{\partial \mathbf{a}_i}$ by using

$$\begin{aligned}
\frac{\partial \text{MSE}}{\partial \mathbf{a}_i} &= \frac{\partial \text{MSE}}{\partial \mathbf{a}_{i+1}} \frac{\partial}{\partial \mathbf{a}_i} \big(\sigma(W_i \mathbf{a}_i + \mathbf{b}_i)\big) \\
&= \frac{\partial \text{MSE}}{\partial \mathbf{a}_{i+1}} \frac{\partial \sigma(z_i)}{\partial z_i} \frac{\partial z_i}{\partial a_i} \\
&= \frac{\partial \text{MSE}}{\partial \mathbf{a}_{i+1}} \text{diag}(\sigma'(W_i \mathbf{a}_i + \mathbf{b}_i)) W_i
\end{aligned}$$

Please see the code for the implementation.

(e) One method to shortcut training is to randomly initialize the weights of the neural network and only train the weights of the last layer. This effectively treats the hidden layers of the neural network as random feature mappings, similar in spirit to the random feature mappings we implemented in Homework 5. **Use regularized least-squares classification to optimize the weights of the final layer. Compare the resulting approximation mean-squared-error values and misclassification error values for the 8 cases above (2 nonlinearities times 4 widths). Comment on what you see.**

Solution:

(f) We use gradients to update the model parameters using batched stochastic gradient descent. **Implement the function `GDOptimizer.update` to update the parameters in each layer of the network.** You will need to use the derivatives $\frac{\partial \text{MSE}}{\partial \mathbf{z}_i}$ and the outputs of each layer $\mathbf{a}_i$ to compute the derivates $\frac{\partial \text{MSE}}{\partial \mathbf{W}_i}$ and $\frac{\partial \text{MSE}}{\partial \mathbf{b}_i}$. Use the learning rate η, given by `self.eta` in the function, to scale the gradients when using them to update the model parameters. **Choose several different batch sizes and number of training epochs. Report the final error on the training set given the various batch sizes and training epochs**

Solution: Please see the code for the implementation.

(g) Let's now explore how the number of hidden nodes per layer affects the approximation. **Train a models using the tanh and the ReLU activation functions with** $2, 4, 8, 16,$ **and** 32 **hidden nodes per layer (width)**. Use the same training iterations and learning rate from the starter code. **Report the resulting error on the training set after training for each combination of parameters.**

(h) **Bonus:** Currently the code classifies between the digits 3 and 9, modify the variables `N0` and `N1` in the code to classify between other pairs of digits. Do your results change if you pick different digits? Do you need more or less hidden nodes to reach the same accuracy? Do you need more or less training epochs?

(i) **Bonus:** Modify the code to implement **multi-class** classification. That is regress to a *vector* of a one hot encoding of the label. (Class 0 will be mapped to e_1 (the first standard basis vector), class 1 will be mapped to e_2 and so on). You will need to change the code that loads the data to load all the classes. Report the accuracy on the test set.

Solution:

5 Dataset Creation: Accessibility

In March of 2018, Google announced a "wheelchair accessible" routes option in transit navigation on Google Maps.[1] The tool is based largely on crowd-sourced information,[2] and it is currently only available in six cities. Can we scale up this process using other information?

Motivated by this question, we will consider the problem of automatically labeling images of building entrances as either accessible or inaccessible to a person in a wheelchair from the sidewalk. The first step is to collect labeled examples, which is your task for this homework problem.

Take twenty pictures each of specific building entrances that are accessible or inaccessible, for a total of 40 images. The images should be taken from the sidewalk or the street, with the entrance centered. A human should be able to determine from the image whether or not the entrance is accessible or inaccessible. See examples below.

Once you have collected the images, **crop and/or resize each one to be** 256×256 **pixels**, keeping the entrance centered. You are provided with a script which demonstrates how to use functions from `skimage` to automate this process in `crop_resize.py`. **Save the resized images as .png files**. Do not save the pictures as .jpg files, since these are lower quality.

Next, place all images of accessible entrances in a folder named *acc* and all images of inaccessible entrances in a folder named *inacc*. In each of these folders, **include a file named *rich_labels.txt* which contains information about the geographic location of each image**. The file should contain entries for each image in the folder. Entries should be on new lines prefixed by the file name in the following form:

$$\underbrace{\text{sodahall.png}}_{\text{filename}}\underbrace{\quad}_{\text{space}}\underbrace{37.875315}_{\text{latitude}}\underbrace{\quad}_{\text{space}}\underbrace{-122.258645}_{\text{longitude}}$$

The GPS coordinate need not be exact, but please report the latitude and longitude to at least 4 points after the decimal.[3] Place the *acc* and *inacc* folders into a folder titled *data*. To turn in

[1] https://www.blog.google/products/maps/introducing-wheelchair-accessible-routes-transit-navigation/

[2] https://www.blog.google/products/maps/building-map-everyone/

[3] You can use Google Maps to find the GPS coordinate of a location on the map after the fact. Additionally, many cell phones will automatically tag images with the location. You can view this under 'Get Info'→'More Info' or 'Properties'→'Details'.

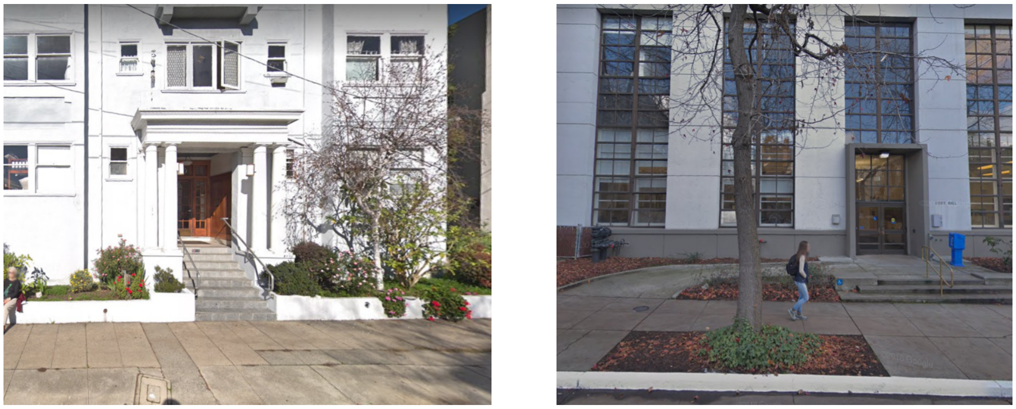


Figure 1: Example of images of entrances: inaccessible (left) and accessible (right).

the folder, **compress it to a .zip and upload it to Gradescope**. We have provided a file called `view_data.py` that will be used to examine and grade your submission. You are encourage to validate the form of your submissions by ensuring that this code runs without warnings.