# Homework 2 - Version 1.1

**Deadline:** Monday, Feb.10, at 11:59pm.

**Submission:** You must submit your solutions as a PDF file through MarkUs[1]. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

See the syllabus on the course website[2] for detailed policies. You may ask questions about the assignment on Piazza[3]. *Note that 10% of the homework mark (worth 1 pt) may be removed for a lack of neatness.*

The teaching assistants for this assignment are Ian Shi and Andrew Jung.
mailto:csc413-2020-01-tas@cs.toronto.edu

## 1 Optimization

This week, we will continue investigating the properties of optimization algorithms, focusing on stochastic gradient descent and adaptive gradient descent methods. For a refresher on optimization, please refer to: https://csc413-2020.github.io/assets/readings/L04.pdf.

We will continue using the linear regression model established in Homework 1. Given $n$ pairs of input data with $d$ features and scalar labels $(\mathbf{x_i}, t_i) \in \mathbb{R}^d \times \mathbb{R}$, we wish to find a linear model $f((x)) = \hat{\mathbf{w}}^T\mathbf{x}$ with $\hat{\mathbf{w}} \in \mathbb{R}^d$ such that the squared error on training data is minimized. Given a data matrix $X \in \mathbb{R}^{n\times d}$ and corresponding labels $\mathbf{t} \in \mathbb{R}^n$, the objective function is defined as:

$$\mathcal{L} = \frac{1}{n}\|X\hat{\mathbf{w}} - \mathbf{t}\|_2^2 \tag{1}$$

### 1.1 Stochastic Gradient Descent (SGD)

SGD performs optimization by taking a stochastic estimate of the gradient from a single training example. This process is iterated until convergence is reached. Let $\mathbf{x}_i \in \mathbb{R}^d$, $1 \le i \le n$ be a single training datum taken from the data matrix $X$. Assume that $X$ is full rank. Where $\mathcal{L}_i$ denotes the loss with respect to $\mathbf{x}_i$, the update for a single step of SGD at time $t$ with scalar learning rate $\eta$ is:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta\nabla_{w_t}\mathcal{L}_i(\mathbf{x}_i, \mathbf{w}_t) \tag{2}$$

SGD iterates by randomly drawing training samples and updating model weights using the above equation until convergence is reached.

#### 1.1.1 Minimum Norm Solution [2pt]

Recall Question 3.4 from Homework 1. For an overparameterized linear model, gradient descent starting from zero initialization finds the unique minimum norm solution $\mathbf{w}^*$ such that $X\mathbf{w}^* = \mathbf{t}$. Let $\mathbf{w}_0 = \mathbf{0}$, $d > n$. Assume SGD also converges to a solution $\hat{\mathbf{w}}$ such that $X\hat{\mathbf{w}} = \mathbf{t}$. Show that SGD solution is identical to the minimum norm solution $\mathbf{w}^*$ obtained by gradient descent, i.e., $\hat{\mathbf{w}} = \mathbf{w}^*$.

[1] https://markus.teach.cs.toronto.edu/csc413-2020-01
[2] https://csc413-2020.github.io/assets/misc/syllabus.pdf
[3] https://piazza.com/class/k58ktbdnt0h1wx?cid=1

*Hint*: Reuse the properties shown in Homework 1 Q3.4. Is $\mathbf{x}_i$ contained in span of $X$? Do the update steps of SGD ever leave the span of $X$?

**Answer**

Consider that $\mathbf{x}_i$ is sampled as a row of $X$. By definition, $\mathbf{x}_i$ is within the row space of $X$. The gradient calculation for squared loss is:

$$\nabla \mathcal{L}_i = \mathbf{x}_i^T \mathbf{x}_i \mathbf{w} - \mathbf{x_i}^T \mathbf{t}$$

Starting from zero weight initialization (contained in span of $X$), the gradient always lies within the span of $X$. Therefore, the update steps of SGD are linear combinations of vectors within $X$, and will never leave span of $X$. By the question premise, we can assume a solution $\hat{\mathbf{w}}$ is obtained. Since SGD only find linear combinations of $X$, we can represent $\hat{\mathbf{w}} = X^T a$ for some $a \in \mathbb{R}^n$ such that $\hat{\mathbf{w}} \in \text{span}\{X\}$. Let $\mathbf{w}_1$ be any other solution to the linear regression model. We have:

$$\begin{aligned}(\hat{\mathbf{w}} - \mathbf{w}_1)^T \hat{\mathbf{w}} &= (\hat{\mathbf{w}} - \mathbf{w}_1)^T X^T a \\ &= (X(\hat{\mathbf{w}} - \mathbf{w}_1))^T a \\ &= (\mathbf{t} - \mathbf{t})^T a = 0\end{aligned}$$

Using this orthogonality condition, the proof in HW1 Q3.4.2 via the generalized Pythagorean theorem can be used to show that $\hat{\mathbf{w}}$ has the smallest norm of all solutions, also implying $\hat{\mathbf{w}} = \mathbf{w}^*$.

Geometric interpretation: the minimum norm solution lies in the span of $X$, and is orthogonal to all other solutions as shown in Q3.4.2. This implies all other solutions involve an addition of a vector in the null space of $X$ (due to the fundamental theorem of linear algebra). The updates in SGD do not involve any vectors in the null space of $X$. Assuming a solution is found, then it must be orthogonal to the null space of $X$. By Q3.4.2, this solution indeed has minimum norm.

### 1.1.2 SGD with Momentum [0pt]

As a corollary to Question 1.1.1, consider SGD with momentum. The update step at time $t$ with scalars $\alpha$ and $\eta$ is described as:

$$\delta_{t+1} = -\eta \nabla \mathcal{L}_i(\mathbf{x}_i) + \alpha \delta_t \tag{3}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \delta_{t+1} \tag{4}$$

Under the assumptions in Question 1.1.1, provide an intuitive argument regarding whether stochastic gradient descent with momentum obtains the minimum norm solution on convergence.

*Hint:* Consider the effects of the momentum term on the linearity of gradient update.

**Answer**

Yes. Solution essentially follows from solution in 1.1.1. Since momentum terms do not affect the linearity of the SGD update steps, the same arguments can be made to show that SGD recovers the minimum norm solution.

### 1.1.3 Mini-batch SGD [0pt]

Recall that mini-batch SGD performs stochastic gradient descent by considering the gradient of mini-batches of data $B \in \mathbb{R}^{b \times d}$ where $1 < b << n$ and $B$ is taken from the rows of $X$. Under the assumptions in Question 1.1.1, does mini-batch stochastic gradient descent obtain the minimum norm solution on convergence?

**Answer**

Yes. Solution again follows from solution in 1.1.1. Since each batch $B$ is sampled from the rows of $X$, it is in the row space of $X$, and thus gradients will be within the span of $X$, as is the solution. The prior proofs can then be used to show minimum norm.

## 1.2 Adaptive Methods

We will next consider the behavior of adaptive gradient descent methods. In particular, we will investigate the Adagrad method. Let $w_i$ denote the $i$-th parameter. A scalar learning rate $\eta$ is used. At time $t$ for parameter $i$, the update step for Adagrad is shown by:

$$w_{i,t+1} = w_{i,t} - \frac{\eta}{\sqrt{G_i, t} + \epsilon} \nabla_{w_{i,t}} \mathcal{L}(w_{i,t}) \tag{5}$$

$$G_{i,t} = G_{i,t-1} + (\nabla_{w_{i,t}} \mathcal{L}(w_{i,t}))^2 \tag{6}$$

The term $\epsilon$ is a fixed small scalar used for numerical stability. Intuitively, Adagrad can be thought of as adapting the learning rate in each dimension to efficiently move through badly formed curvatures (see lecture slides/notes).

### 1.2.1 Minimum Norm Solution [0pt]

Consider the overparameterized linear model ($d > n$) for the loss function defined in Section 1. Assume the Adagrad optimizer converges to a solution. Provide a proof or counterexample for whether Adagrad always obtains the minimum norm solution.

*Hint:* Compute the 2D case from HW1. Let $\mathbf{x}_1 = [2, 1]$, $w_0 = [0, 0]$, $t = [2]$.

**Answer**

It should be clear after a few updates that the updates take the weights outside of the span of $X$. The crux of the issue is that the adaptive gradient "corrections" break the linearity of the data, and it can be extrapolated that the updates will in most cases reach a solution which is not minimum norm. In a particularly degenerate case, there exists some learning rate which perfectly takes Adagrad to a non-minimum norm solution after just a 2 steps (that students can potentially find). Sufficient also to show that the update after a few steps does not lie within the span of $X$, meaning previous proof is violated.

### 1.2.2 [0pt]

Consider the result from the previous section. Does this result hold true for other adaptive methods (RMSprop, Adam) in general? Why might making learning rates independent per dimension be desireable?

# 2 Gradient-based Hyper-parameter Optimization

In this problem, we will implement a simple toy example of *gradient-based hyper-parameter optimization*, introduced in Lecture 3 (slides 21).

Often in practice, hyper-parameters are chosen by trial-and-error based on a model evaluation criterion. Instead, *gradient-based hyper-parameter optimization* computes gradient of the evaluation criterion w.r.t. the hyper-parameters and use this gradient to directly optimize for the best set of hyper-parameters. For this problem, we will optimize for the learning rate of gradient descent in a linear regression problem, like in homework 1.

Similar to homework 1, a linear model will be used for this problem. Specifically, given $n$ pairs of input data with $d$ features and scalar label $(\mathbf{x}_i, t_i) \in \mathbb{R}^d \times \mathbb{R}$, we wish to find a linear model $f(\mathbf{x}) = \hat{\mathbf{w}}^\top \mathbf{x}$ with $\hat{\mathbf{w}} \in \mathbb{R}^d$ that minimizes the squared error of prediction on the training samples. Using the concise notation for the data matrix $X \in \mathbb{R}^{n \times d}$ and the corresponding label vector $\mathbf{t} \in \mathbb{R}^n$, the squared error loss can be written as:

$$\mathcal{L} = \frac{1}{n} \|X\hat{\mathbf{w}} - \mathbf{t}\|_2^2.$$

Starting with an initial weight parameters $\mathbf{w}_0$, gradient descent updates $\mathbf{w}_0$ with learning rate $\eta$ for $t$ number of iterations. Let's denote the model parameters after $t$ gradient descent iterations as $\mathbf{w}_t$. After gradient descent, the loss is calculated with $\mathbf{w}_t$ and gradient of the loss w.r.t. the learning rate $\eta$ can be computed. Using this gradient, the best $\eta$ can be obtained.

## 2.1 Computation Graph [2pt]

### 2.1.1

Consider a case of 2 gradient descent iterations. Let's denote the loss and gradient of the loss computed with $\mathbf{w}_t$ as $\mathcal{L}_t$ and $\nabla_{\mathbf{w}_t}\mathcal{L}_t$. Draw the computation graph for calculating final loss (i.e. $\mathcal{L}_2$ calculated with $\mathbf{w}_2$) using variables $\mathbf{w}_0, \mathcal{L}_0, \nabla_{\mathbf{w}_0}\mathcal{L}_0, \mathbf{w}_1, \mathcal{L}_1, \nabla_{\mathbf{w}_1}\mathcal{L}_1, \mathbf{w}_2$, and $\eta$.

### 2.1.2

Then, consider a case of $t$ gradient descent iterations. What would be the memory complexity for forward-propagation (i.e. just calculating $\mathcal{L}_t$) in terms of $t$? What would be the memory complexity for computing back-propagation (i.e. computing $\nabla_{\mathbf{w}_t}\mathcal{L}_t$) in terms of $t$?

### 2.1.3

Explain one potential problem for applying gradient-based hyper-parameter optimization in more realistic examples where models often take many iterations to converge.

## 2.2 Single gradient descent iteration [2pt]

Let's consider a case with only one gradient descent iteration.

Starting from $\mathbf{w}_0$, gradient descent will update the model weight to $\mathbf{w}_1$.

### 2.2.1

Calculate $\mathbf{w}_1$ and using this expression, write down the loss $\mathcal{L}_1$ after single gradient descent iteration. Determine if this $\mathcal{L}_1$ is convex w.r.t. the learning rate $\eta$.

*Hint*: if the expression gets too messy, introduce a constant vector $\mathbf{a} = X\mathbf{w}_0 - \mathbf{t}$
*Hint*: a function is convex if its second order derivative is positive

**Answer**

$\mathbf{w}_1 = \mathbf{w}_0 - \eta X^T\mathbf{a}$, $\mathcal{L}_1 = \frac{1}{2}\mathbf{a}^T\left(-\eta XX^T + I\right)^2\mathbf{a}$. $\mathcal{L}$ is convex since positive order 2 polynomial w.r.t. $\eta$.

#### 2.2.2

Calculate the derivative of $\mathcal{L}_1$ w.r.t. $\eta$ and find the best $\eta$.

**Answer**

$\frac{\partial \mathcal{L}_1}{\partial \eta} = \mathbf{a}^T\left(-\eta XX^T + I\right)(-XX^T)\mathbf{a} = 0$. Thus, $\eta = \frac{(X^T\mathbf{a})^2}{(XX^T\mathbf{a})^2}$

#### 2.2.3

For the 2D over-parameterized case from homework 1 (i.e. $X = [2; 1]$ and $\mathbf{t} = [2]$), what is the optimal $\eta$? Compare $\mathbf{w}_1$ with the optimal $\eta$ in the question with the result you obtained from homework 1. How does the optimal $\eta$ help reach this solution? (i.e. describe the trajectory)

**Answer**

$\eta = \frac{1}{5}$. Get the same $\mathbf{w}_1$ from homework 1. Learning rate such that in a single gradient descent, reach optimal $\mathbf{w}$.

### 2.3 Multiple inner-loop iteration [0pt]

#### 2.3.1

Calculate the loss $\mathcal{L}_t$ after $t$ gradient descent updates.
*Hint*: proof by induction and binomial coefficients can be useful

**Answer**

$\mathcal{L}_t = \frac{1}{2}\mathbf{a}^T\left(-\eta XX^T + I\right)^{2t}\mathbf{a}$.

#### 2.3.2

Determine if this $\mathcal{L}_t$ is in general convex w.r.t. the learning rate $\eta$?

**Answer**

After eigenvalue decomposition, $XX^T = Q\Lambda Q^T$.
Then, $\mathcal{L}_t = \frac{1}{2}\mathbf{a}^T Q\left(-\eta\Lambda + I\right)^{2t} Q^T\mathbf{a} = \frac{1}{2}\sum_{i=1}^n b_i^2(-\eta\lambda_i + 1)^{2t}$ where $b_i, \lambda_i$ are i'th element of $Q^T\mathbf{a}, \Lambda$.

$\frac{\partial^2 \mathcal{L}_t}{\partial \eta^2} = t(2t-1)\sum_{i=1}^n b_i^2\lambda_i^2(-\eta\lambda_i + 1)^{2t-2} > 0$

Therefore, convex.

#### 2.3.3

For the previous 2D over-parameterized case from homework 1, describe the convexity of $\mathcal{L}_t$ w.r.t. $\eta$. What is the optimal $\eta$?

**Answer**

In this case, the loss is convex and get the same $\eta = \frac{1}{5}$.

# 3 Convolutional Neural Networks

The last set of questions aims to build basic familiarity with Convolutional Neural Networks. To refresh your knowledge, please refer to the reading on CNNs: `https://csc413-2020.github.io/assets/readings/L05.pdf`

## 3.1 Convolutional Filters [1pt]

Given the input matrix $\mathbf{I}$ and filter $\mathbf{J}$ shown below, 1) Write down the values of the resulting matrix $(\mathbf{I} * \mathbf{J})$ (the convolution operation as defined in the Lec 5 slides). Assume we have use zero padding around the input. 2) What feature does this convolutional filter detect?

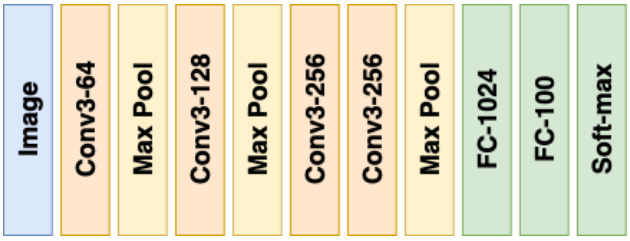
$$\mathbf{I} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \mathbf{J} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \mathbf{I} * \mathbf{J} = \begin{bmatrix} ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix}$$

**Answer**

$$\mathbf{I} * \mathbf{J} = \begin{bmatrix} -1 & 2 & 2 & -2 & 0 \\ -2 & 1 & 0 & 2 & -1 \\ 3 & 0 & 0 & 1 & -1 \\ -2 & 2 & 0 & 2 & -1 \\ 0 & -2 & 3 & -2 & 0 \end{bmatrix}$$

## 3.2 Size of a conv net [1pt]

Consider a conv net with 4 conv layers like in the diagram below. All 4 conv layers have kernel size of $3 \times 3$. The number after the hyphen specifies the number of output channels or units of a layer (e.g. *Conv3-64* layer has 64 output channels and *FC-1024* has 1024 output units). All the *Max Pool* in the diagram has size of $2 \times 2$. Assume zero padding for conv layers and stride 2 for *Max Pool*.



Size of the RGB input image is $112 \times 112$ (3 channels).
Calculate the number of parameters for this conv net including the bias units.

**Answer**

Conv layers: $(9{\cdot}3{\cdot}64{+}64){+}(9{\cdot}64{\cdot}128{+}128){+}(9{\cdot}128{\cdot}256{+}256){+}(9{\cdot}256{\cdot}256{+}256){+}(9{\cdot}128{\cdot}256{+}256) = 1792 + 73856 + 295168 + 590080 = 960896$

Dense layers: $(14 \cdot 14 \cdot 256 \cdot 1024 + 1024) + (1024 \cdot 100 + 100) = 51381248 + 102500 = 51483748$

Total: 52444644