MATH 311 - Linear Algebra Applications and Proof Assignment

Evan Barker

Application Topic: **Deep Learning Networks**

Part I: Linear Algebra Applications Essay

- Introduction to the application

My chosen topic is deep learning, which is classified as a sub-branch of machine learning (ML). Its focus is to utilize multi-layered neural networks[1] (multiple, hence the 'deep' in 'deep learning') to perform tasks like classification, regression and representation learning [1]. Deep learning can be traced all the way back to the 1940s, where Walter Pitts and Warren McCulloch created a computer model based on the neural networks of the human brain [3]. Today, deep learning has taken off with regards to the transformer architecture being embraced in GPTs (generative pre-trained transformer) [4]. From a higher level standpoint, the modern day deep learning architectures (such as GPTs) have enabled unprecedented potential due to their parallel processing capabilities and increased scalability [5],[6]. This application is important and

[1] According to AWS, a neural network is a "method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning (ML) process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain" [2].

interesting to me because it demonstrates so vividly how much the mathematics behind everything, when combined with computing power and capabilities, is able to transform so many people's lives; teachers, doctors, lawyers, business people, students, and so on. In particular, I find it fascinating how this application is able to play a role in elevating everyone's baseline level of productivity in society. Overall, deep learning affects many fields/focuses including GPTs (like Chat GPT), computer vision systems, healthcare and bioinformatics, finance and economics, and autonomous robots and vehicles, just to name a few. These contributions matter deeply not only because they further advance technologies and innovations but because they can *directly* be used to help save peoples lives (see paper by Mathi et. al. in the references section detailing how CNNs, convolution neural networks, help to quickly diagnose cancers) [7].

- Role of Linear Algebra

Linear algebra is used <u>a lot</u> in deep learning. For starters, the input data such as text files, images, motion data, etc. are all represented numerically using vectors, matrices and higher dimensional tensors [8]. Now, there are also many <u>transformations</u> (both linear and nonlinear) that take place in these deep learning networks. Focusing specifically on the simplest building block 'atom' of deep learning networks, we arrive at the perceptron. Invented by Frank Rosenblatt in 1957, it was the first artificial neural network. Below depicts a visual representation for a single-layer perceptron,

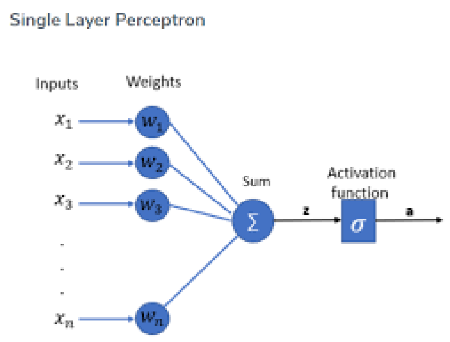

Image source: https://antoblog.medium.com/what-is-perceptron-a-beginnerstutorial-for-perceptron-632539884146

What can be realized here is that the input to your activation function, let's call it Z, is just a linear transformation of your inputs and weights (there's often also a bias term used) [9]. To be more concrete and mathematical, we can posit some scenario whereby we have some input vector $X \epsilon R^{4}$, some weight transformation matrix given as $W \epsilon R^{3x4}$ and then some bias $B \epsilon R^{3}$.Now, given this, our output can be represented as,

Z = WX + B, where $Z \epsilon R^{3}$

Clearly Z is a linear transformation! And, this linear transformation is hugely important for compressing/expanding information; that is, in the case above, we have compressed information because we reduced the dimensionality of the output layer by 1 (we went from 4 dimensions to 3). Also, by incorporating weight matrices, we are essentially dictating which inputs are more important compared to others; i.e., sometimes there's noise/unwanted data we want to eliminate or reduce, so we can set the weighting corresponding to that input noise/data to be very low. Below is another diagram, this time showing a multi-layered perceptron,

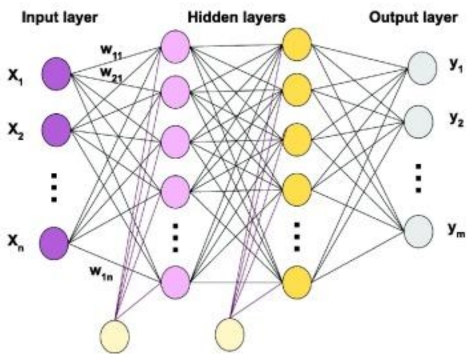

Image source: https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron

Again, we observe the same trend as compared with the single-layer perceptron above. Overall, we can view the information stored in each node as the scalar value under some field, the layers as vectors, and the links between layers as linear transformations dependent on the assigned weight (transformation) matrices. Furthermore, the linear transformations can induce a dimensionality reduction or expansion between layers to compress or expand the information.

To further glean what's going on with dimensionality, if we reuse our concrete example above ($Z = WX + B$) and assume that W is full rank, then by the Rank-Nullity theorem, it must be that,

$$\dim(\ker(W)) + \dim(\mathrm{im}(W)) = \dim(\mathrm{domain})$$

In this case, if W is full rank, then dim(imW)) = 3 and dim(ker(W)) = 1. And, this indeed lines up with the Rank-Nullity theorem. Translating this back to the application, this means that one degree of freedom in the input space is completely mapped to zero. In other words, the weights determine that one direction of the input data doesn't matter for prediction.

- Impact and Significance

This application solves numerous problems. In daily life, this application has solved the problem of lack of intelligent automation; in particular, with application of deep learning networks, you can train the computing system to become 'intelligent' and automated in the sense that it's 'independent' and 'self-aware' [10]. For instance, with the Tesla cars, the deep learning networks were pivotal to Tesla's autopilot and full self driving (FSD) systems [11]. One aspect of how linear algebra has advanced this field relates with principal component analysis (PCA). Although it gets very complicated very quickly, at a high level PCA works by choosing vectors where the first chosen vector is chosen based on it having the greatest amount of variance along its projection. The second vector is orthogonal to the first, and is chosen such that it contains the greatest variance left after the first principle component [12]. Overall, these principal components form an orthonormal basis that spans the subspace of the original input data space. The overarching idea here is that if your input data set is incredibly large, then you can apply PCA to simplify things; this will reduce the computational load, which in turn reduces the cost with training a deep learning network [13]. Without PCA, it would be much harder to train the deep learning networks because there would be so much information inputted, and given the *hard constraints* on computation, it would cause the system to be slower, less efficient, and less scalable. Moreover, the signal-to-noise (SNR, a common concept in communications in electrical engineering) would decrease because with the increased inputs that are irrelevant (possible outliers) would be factored into the training and thus the models may learn to overfit spurious patterns in the data. Besides PCA though, if other linear algebra concepts weren't applied (matrices, vectors, transformations), then practically all of deep learning would crumble; practically everything I've outlined above with deep learning rests upon linear algebra structures. One interesting development taking place which piques my interest, especially with all the ethical calamities that can potentially rise with deep learning in the future, relates with something called federated learning. According to IBM, this is when the AI models are trained with data that is not sent to any remote server; in other words, the data exchanged between you and the AI stays on your device. I find this particularly fascinating not only because it assuages my qualms around ethical mishaps of AI revolving around privacy, but because it means that AI is more likely to be further incorporated in information sensitive domains like business or healthcare [14].

Part II: Theorem Statement and Proof

Eckart-Young-Mirsky Theorem (EYM Theorem)

This theorem comes from Eckart & Young [17] and Mirsky [18], construction of it was aided by [19] and [20].

Let $W \epsilon R^{mxn}$ be a matrix of rank r, and suppose $1 \leq k < r$. Then the matrix,

$$W_k = U_k \Sigma_k V_k^\top$$

is the best rank-k approximation to W with respect to the Frobenius norm. Here,

- $U_k \epsilon R^{mxk}$ are the first k columns of U
- $\Sigma_k \epsilon R^{kxk}$ are the top-left k x k submatrix of Σ
- $V_k \epsilon R^{nxk}$ are the first k columns of V

This approximation minimizes the Frobenius norm distance over all rank-$\leq k$ matrices $A \epsilon R^{mxn}$,

$$\left\|W - W_k\right\|_F = min_{rank(A) \leq k} ||W - A||_F$$

Where W_k is the matrix of rank at most k that is closest to W in the Frobenius norm, and here $A \epsilon R^{mxn}$ where A has characteristic that rank(A) ≤ k. This theorem directly relates to part 1 where I explored how linear algebra enables information compression in deep learning networks. In practice, when the neural networks become too large or computationally expensive, it is common to apply low-rank matrix approximations to weight matrices between layers to reduce model size while retaining the most of the representational power. In this case, EYM theorem gives us a 'solution' to find the best low-rank matrix approximation of the original weighted matrix W. Overall, the importance of this theorem stems from the fact that it enables for lighter, faster, more deployable deep learning neural networks, while at the same time minimizing information loss, thereby preserving the networks learning capacity [21]-[24].

- Proof (Eckart-Young-Mirsky)

Definitions:

1. If $A \epsilon R^{mxn}$ then $A = U\Sigma V^{\top} = \sum_{i=1}^{r} \sigma_i u_i v_i^{\top}$ is the SVD of A, where

- $\sigma_i \epsilon R$, $u_i \epsilon R^m$ and $v_i \epsilon R^n$
- r = rank(A), and $\sigma_1 > \sigma_2 > \sigma_3 > \ldots > \sigma_r > 0$ are the singular values of A
- $U \epsilon R^{mxm}$ *and* $V \epsilon R^{nxn}$ are orthogonal matrices

2. The Frobenius norm is defined, for some matrix $A \epsilon R^{mxn}$, as

- $||A||_F = (\sum_{i=1}^{n}\sum_{j=1}^{m} \left|a_{ij}\right|^2)^{1/2} = tr(AA^\top) = tr(A^\top A)$

3. Weyl's Inequality

- Let A, B be Hermitian on inner product space V with dimension n, with spectrum ordered in descending order $\lambda_1 \geq \lambda_2 \geq \ \dots \ \geq \ \lambda_n$. Then, the following inequality holds,

$$\lambda_{i+j-1}(A + B) \leq \lambda_i(A) + \lambda_j(B) \leq \lambda_{i+j-n}(A + B)$$

Let us begin the proof. We shall prove this for the <u>Frobenius norm</u>.

Let $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^\top$ be the truncated SVD matrix of A with rank(A) = n. Let H be any *arbitrary* rank-k matrix, our goal is to show that,

$$\left|\left|A - A_k\right|\right|_F \leq ||A - H||_F$$

Hence, we need to show that A_k is the closest to A in the Frobenius norm out of all possible rank k matrices H. If we apply definition 3 (Weyl's inequality), and letting A = A-H and letting B = H from above, then we can say,

$$\lambda_{i+k}(A) \ \leq \ \lambda_i(A - H) + \lambda_{k+1}(H)$$

Since we defined H to be of rank k then it must hold that $\lambda_{k+1}(H) \ = \ 0$. Hence,

$$\lambda_{i+k}(A) \ \leq \ \lambda_i(A - H) \text{ holds.} \quad *$$

Now, based on this inequality, let us state,

$$\left\|A - A_k\right\|_F = \left\|\sum_{i=1}^{n} \sigma_i u_i v_i^\top - \sum_{i=1}^{k} \sigma_i u_i v_i^\top\right\|_F \text{, by definition one (SVD)}$$

$$= \left\|\sum_{i=k+1}^{n} \sigma_i u_i v_i^\top\right\|_F \text{, by rearranging things}$$

$$= \left\|U\Sigma_k V^\top\right\|_F \text{, by definition one (SVD)}$$

$$= \left\|\Sigma_k\right\|_F \text{, since U and } V^\top \text{ are orthogonal}$$

$$= \sum_{i=k+1}^{n} \sigma_i(A)^2 \text{, by definition two (Frobenius norm)}$$

$$= \sum_{i=1}^{n-k} \sigma_{i+k}(A)^2 \text{, by rearranging things}$$

$$\leq \sum_{i=1}^{n-k} \sigma_i(A - H)^2 \text{, by our construction of * above}$$

$$\leq \sum_{i=1}^{min(m,n)} \sigma_i(A - H)^2 \text{, (follows trivially)}$$

$$\leq ||A - H||_F$$

Hence, we have just shown that $\left\|A - A_k\right\|_F \leq ||A - H||_F$.

This implies that A_k is the best rank-k approximation to W with respect to the Frobenius norm out of all possible rank-k matrices. ▨

- Reflection

The biggest challenge I faced wasn't proving this theorem per se because I understood how to piece components/definitions together, the hard part was gaining a deep understanding for it by first deeply understanding SVD. In other words, the hard part was conceptually digesting a bunch of information in a shorter period of time. Moreover, understanding how it connected with my practical application was a bit tricky but as I thought about it for a bit it started to make more sense. This theorem connected to concepts we went over such as orthogonality with respect to when I claimed we can go from $\left\|U\Sigma_k V^\top\right\|_F$ to $\left\|\Sigma_k\right\|_F$. Also, this theorem relies on a firm and deep understanding of all the underlying structures of linear algebra such as matrix multiplication, diagonal matrices, symmetric matrices, and so on which all are a part of SVD. One insight I gained from linear algebra with this proof is how elegant it is for supporting practical applications/results; in this case, the theoretical aspects of linear algebra supported something very powerful practically! (compression in deep learning networks) Proving this theorem helped me to understand more deeply/abstractly what the *process* is like for finding the best lower ranked weight matrix in a neural network. Overall, I really loved this assignment and loved MATH 311; taking this course and doing this assignment made me realize how much more there is to learn and explore, which is exhilarating.

Works Cited

[1] “Deep learning,” Wikipedia, https://en.wikipedia.org/wiki/Deep_learning (accessed Jun. 8, 2025).

[2] Amazon Web Services, “What is a neural network?” https://aws.amazon.com/what-is/neural-network/ (accessed Jun. 8, 2025).

[3] “A brief history of deep learning,” Dataversity, https://www.dataversity.net/brief-history-deep-learning/ (accessed Jun. 8, 2025).

[4] “Transformer (deep learning architecture),” Wikipedia, https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture) (accessed Jun. 8, 2025).

[5] G. Klockwood, “Scaling laws for deep learning,” Glenn Klockwood's Blog, https://glennklockwood.com/garden/scaling-laws (accessed Jun. 8, 2025).

[6] A. Vaswani et al., “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[7] S. M. Zaman, “Use of artificial intelligence in neonatal healthcare,” *J. Neonatal Surg.*, vol. 11, no. 1, 2022. [Online]. Available: https://jneonatalsurg.com/index.php/jns/article/view/5117/4274

[8] “ML – Linear algebra operations,” GeeksforGeeks, https://www.geeksforgeeks.org/ml-linear-algebra-operations/ (accessed Jun. 8, 2025).

[9] T. Raiko, H. Valpola, M. Harva, and J. Karhunen, “Deep learning made easier by linear transformations in perceptrons,” in *Proc. 15th Int. Conf. Artif. Intell. Stat. (AISTATS)*, vol. 22, 2012, pp. 924–932. [Online]. Available: https://proceedings.mlr.press/v22/raiko12/raiko12.pdf

[10] Nivida Software, “What are the real-life examples of deep learning?” https://www.nividasoftware.com/blog/detail/what-are-the-real-life-examples-of-deep-learning (accessed Jun. 8, 2025).

[11] Tesla, “Autopilot,” https://www.tesla.com/en_ca/support/autopilot (accessed Jun. 8, 2025).

[12] Keboola, “PCA in machine learning: Principal component analysis explained,” https://www.keboola.com/blog/pca-machine-learning (accessed Jun. 8, 2025).

[13] Y. Liu, R. Yu, and Y. Wang, “A comparative study on transformer model variants,” *arXiv preprint*, arXiv:2501.19114, 2025. [Online]. Available: https://arxiv.org/html/2501.19114v1

[14] IBM Research, “What is federated learning?” https://research.ibm.com/blog/what-is-federated-learning (accessed Jun. 8, 2025).

[15] S. Axler, *Linear Algebra Done Right*, 3rd ed. New York, NY, USA: Springer, 2015.

[16] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Are all layers created equal?" in *Advances in Neural Information Processing Systems*, 2019. [Online]. Available: https://arxiv.org/abs/1902.01996

[17] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep. 1936. doi: 10.1007/BF02288367.

[18] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *Q. J. Math.*, vol. 11, no. 1, pp. 50–59, 1960. doi: 10.1093/qmath/11.1.50.

[19] G. Rabusseau, "Low-rank matrix approximation," Lecture Notes, Université de Montréal, 2020. [Online]. Available: https://www-labs.iro.umontreal.ca/~grabus/courses/ift6760_W20_files/lecture-5.pdf (accessed Jun. 8, 2025).

[20] "Weyl's inequality," Wikipedia, https://en.wikipedia.org/wiki/Weyl%27s_inequality (accessed Jun. 8, 2025).

[21] "Eigenvector computation and low-rank approximations," GeeksforGeeks, https://www.geeksforgeeks.org/eigenvector-computation-and-low-rank-approximations/ (accessed Jun. 8, 2025).

[22] Y. Idelbayev and M. Á. Carreira-Perpiñán, "Low-Rank Compression of Neural Nets: Learning the Rank of Each Layer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 18048–18056.

[23] S. Dittmer, E. J. King, and P. Maass, "Learning Low‑Rank Deep Neural Networks via Singular Vector Orthogonality Regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 552–565, Mar. 2020.

[24] A. Yaguchi et al., "Decomposable‑Net: Scalable Low‑Rank Compression for Neural Networks," in *Proc. IEEE/IJCAI Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 3249–3255.