OPT2020: 12th Annual Workshop on Optimization for Machine Learning

Measuring optimization performance of stochastic gradient descent via neural networks with threshold activation

Junyoung Kim XHXHZLD@SNU.AC.KR
Kyungsik Lee OPTIMA@SNU.AC.KR
Department of Industrial Engineering, Seoul National University, Republic of Korea

Abstract

In this paper, we consider a single hidden layer neural network with threshold activations where the in-degree of each hidden node is restricted to some positive integer k less than the dimension of samples, which we call k-SNN. We first analyze the expressivity of k-SNN on finite samples and show that it can express any labeling of training samples under a mild assumption. Then, we show that the training problem for k-SNN can be solved in polynomial time for fixed k. We also show that we can construct a single hidden layer neural network with ReLU activations (R-SNN) with the same loss as that of the optimized k-SNN, which make it possible to measure the optimization performance of a widely used stochastic gradient descent (SGD) algorithm by training the constructed R-SNN using the algorithm. Preliminary computational results show that SGD cannot converge to a global optimum on small-sized network even though it is sufficiently large to fit training samples precisely.

1. Introduction

Neural networks and deep learning have shown outstanding empirical performance in many applications such as image recognition [9, 15, 23], natural language processing [6], speech recognition [11, 17, 21]. To explain their successes in many practical applications, complexity measures of hypothesis classes defined by various network architectures and regularization methods have been studied [12]. However, for an over-parameterized neural networks (OPNN) that have a large number of hidden nodes compared to the number of training samples, some phenomena unexplained by such complexity measures have been observed [18, 24]. Specifically, those OPNNs trained with widely used optimization method, stochastic gradient descent (SGD), show good generalization ability even without an explicit regularization such as weight decay or drop out. This observation has motivated studies on the generalization ability associated with SGD [4, 16, 20, 22]. However, analyzing the optimization performance of SGD is needed first to understand its generalization ability. Some studies show the global convergence of SGD for large-sized OPNNs with a huge number of hidden nodes [8, 25]. However, its optimization performance, for smaller networks, is still unknown as far as we know.

For a fully connected single hidden layer neural network with threshold activations, Bengio et al. [3] proposed a training algorithm that can find a global optimum. This algorithm iteratively solves sub-problems that generate hidden nodes improving the objective value based on the column generation method [7]. They also showed that the trained network has far less hidden nodes than the number of samples for some toy examples. However, this framework is not scalable since the sub-problem is strongly NP-hard [2].

We also consider a single hidden layer neural network with threshold activations. However, in this paper, we limit the in-degree of each hidden node to some fixed positive integer $k \leq d$, where d is the dimension of training samples, which we call k-SNN. The contents and contributions of our paper are summarized as follows :

- In section 2, we give the definition of k-SNN and analyze the expressivity of k-SNN on finite samples. We show that k-SNN can express any labeling of training samples for given $k \leq d$ under a mild assumption.
- Then, we present a polynomial time algorithm to optimize k-SNN specifically for classification problems in section 3. We also show that the optimized k-SNN can be transformed to a single hidden layer neural network with ReLU activations (R-SNN) with the same loss. This transformation gives an upper bound on the optimal loss that can be achieved by R-SNN.
- Based on the results of section 3, we measure the optimization performance of SGD on relatively small-sized network for the first time as far as we know. The computational tests show quite different results on the optimization performance of SGD from those with large-sized OPNNs.

2. Finite sample expressivity of k-SNN

In this section, we first define k-SNN which consists of a single hidden layer with τ hidden nodes with threshold activations and one output node. Formally, a k-SNN is a function $f : \mathbb{R}^d \to \mathbb{R}$ defined as $f(x) = \sum_{h=1}^{\tau} w_h \psi(\alpha_h^T x + \beta_h) + w_0$ such that $1 \leq \|\alpha\|_0 \leq k$ for given positive integer $k \leq d$, where $\alpha_h \in \mathbb{R}^d$, $\beta_h \in \mathbb{R}$, $w_h \in \mathbb{R}$ for all $h \in [\tau] := \{1, \ldots, \tau\}$, $w_0 \in \mathbb{R}$, and the threshold activation function $\psi(\cdot)$ is defined as

$$\psi(s) = \begin{cases} 1, & s > 0 \\ -1, & s \leq 0 \end{cases}$$

Let $S = \{(x^i, y^i)\}_{i=1}^n$ be a set of finite training samples, where $x^i \in \mathbb{R}^d$ and $y^i \in \mathbb{R}$ for all $i \in [n] := \{1, \ldots, n\}$. The first question is whether or not there exists a k-SNN such that $f(x^i) = y^i$ for all $i \in [n]$ for given positive integer k. Unfortunately, it does not hold in general. However, the following theorem shows that, under a mild condition, k-SNN can fit the given set of samples even for $k < d$, whose proof is given in Appendix A. Let $x_K \in \mathbb{R}^{|K|}$ be the sub-vector obtained by deleting x_j's for all $j \notin K$ from a given $x \in \mathbb{R}^d$, where $K \subseteq [d] := \{1, \ldots, d\}$.

Theorem 1 *For given $k \leq d$, there exists a k-SNN such that $f(x^i) = y^i$ for all $i \in [n]$ if there exists $K \subseteq [d]$ such that $|K| = k$ and $x_K^i \neq x_K^j$ whenever $i \neq j$ for all $i, j \in [n]$.*

3. Training k-SNN for classification problems

Since $\psi(\cdot)$ is not continous, optimization algorithms based on the gradient descent cannot be applied to the training problem for k-SNN. However, the training problem can be convexified [3], i.e., it can be reformulated as a convex optimization problem.

Recall that a hidden node of k-SNN is specified by some $(\alpha, \beta) \in \mathbb{R}^{d+1}$ such that $1 \leq \|\alpha\|_0 \leq k$ which defines a hyperplane $\{x \in \mathbb{R}^d : \alpha^T x + \beta = 0\}$. For given (α, β), let $z(\alpha, \beta) \in \{-1, 1\}^n$

be the output vector for given sample $S = \{(x^i, y^i)\}_{i=1}^n$ defined as $z(\alpha, \beta)_i = \psi(\alpha^T x^i + \beta)$ for each $i \in [n]$. If $z(\alpha, \beta) \neq (1, \ldots, 1)$ and $z(\alpha, \beta) \neq (-1, \ldots, -1)$, then (α, β) is *nontrivial*. Note that if (α, β) is nontrivial, then the corresponding hyperplane partitions S into two non-empty subsets. For given $K \subseteq [d]$ and $z \in \{-1, 1\}^n$, z is *K-achievable* if there exists a nontrivial (α, β) such that $\alpha_j = 0$ for all $j \notin K$ and $z = z(\alpha, \beta)$. For given K-achievable $z \in \{-1, 1\}^n$, there may be infinitely many (α, β)'s corresponding to z. However, there are only finitely many K-achievable output vectors. The main idea of the convexification is first enumerating K-achievable output vectors for each $K \subseteq [d]$, finding (α, β) for each enumerated output vector such that the corresponding separating hyperplane has a maximum margin, and then finding w_0 and w_h for all $h \in [\tau]$ that minimize a convex objective function.

For given sample $S = \{(x^i, y^i)\}_{i=1}^n$ and $k \leq d$, let $\mathcal{Z}_K$ be the set of K-achievable output vectors for $K \subseteq [d]$ with $|K| = k$, and let $\mathcal{Z} := \cup_{\{K \subseteq [d]: |K| = k\}} \mathcal{Z}_K$. For each $z \in \mathcal{Z}$, the corresponding hyperplane with a maximum margin can be obtained by solving $\max\{\|\alpha\|_2^2 : z_i(\alpha^T x^i + \beta) \geq 1, \ \forall i \in [n], (\alpha, \beta) \in \mathbb{R}^{d+1}\}$, whose solution guarantees that $|\alpha^T x^i + \beta| > 0$ for all $i \in [n]$ by the definition of $\mathcal{Z}$. Now, we present a convex optimization problem, specifically a linear program, to train k-SNN for (binary) classification problems as follows. Here, we assume that $y^i \in \{-1, 1\}$ for all $i \in [n]$ for given sample $S = \{(x^i, y^i)\}_{i=1}^n$.

$$\text{(TP)} \quad \text{minimize} \quad \sum_{i \in [n]} \xi_i + \lambda \sum_{z \in \mathcal{Z}} w_z \tag{1}$$

$$\text{subject to} \quad 1 - y^i (w_0 + \sum_{z \in \mathcal{Z}} w_z z_i) \leq \xi_i, \ \forall \, i \in [n] \tag{2}$$

$$\xi_i \geq 0, \ \forall \, i \in [n] \tag{3}$$

$$w_0 \in \mathbb{R}, \ w_z \geq 0, \ \forall \, z \in \mathcal{Z} \tag{4}$$

The objective function (1) of TP consists of the hinge loss function and the ℓ_1-regularization function. w_z is the parameter between the output node and the hidden node corresponding to $z \in \mathcal{Z}$, ξ_i is the hinge loss for each sample, and $\lambda \in \mathbb{R}_+$ is a given regularization factor. Note that we have only to consider $w_z \geq 0$ since $z \in \{-1, 1\}^n$ by the definition of $\psi(\cdot)$.

Based on the results of [19], for given $k \leq d$ and $K \subseteq [d]$ such that $|K| = k$, we can prove that $|\mathcal{Z}_K|$ is at most n^k by showing that for each $z \in \mathcal{Z}_K$, there exist a nontrivial (α, β) with a set of k affinely independent x^i's, $\mathcal{X}(z)$, such that $\alpha_j = 0$ for all $j \notin K$, $\alpha^T x^i + \beta = 0$ for all $x^i \in \mathcal{X}(z)$ and $z = z(\alpha, \beta)$. Since there are at most d^k possibilities to choose K, $|\mathcal{Z}|$ is $O(n^k d^k)$, which means the number of variables of TP is bounded by a polynomial function in n and d for fixed k. Therefore, we have the following theorem since a linear program with polynomial number of variables and constraints can be solved in polynomial time [13].

Theorem 2 *TP can be solved in polynomial time for fixed $k \leq d$.*

By the fundamental theorem of linear programming [5], we can find an optimal extreme point solution (ξ^*, w^*) which satisfies $\|(\xi^*, w^*)\|_0 \leq n$, which means at most n variables of TP can be non-zero. Therefore, the corresponding k-SNN to (ξ^*, w^*) has at most n hidden nodes.

Now, we show that we can construct a single hidden layer neural network with ReLU activations (R-SNN) with the same loss as that of k-SNN corresponding to the obtained optimal solution

(ξ^*, w^*) to TP. Let $m = |\{w_z^* > 0 : z \in \mathcal{Z}\}|$. For the sake of simplicity, let us denote the trained k-SNN as $f^*(x) = \sum_{h=1}^{m} w_h^* \psi(\alpha_h^{*T} x + \beta_h^*) + w_0^*$ where $w_h^* > 0$ and $|\alpha_h^*| \leq k$ for all $h \in \{1, ..., m\}$. Note that there exists ϵ_h such that $0 < \epsilon_h < \min_{i \in [n]}\{|\alpha_h^{*T} x^i + \beta_h^*|\}$ since we chose (α_h^*, β_h^*) such that $|\alpha_h^{*T} x^i + \beta_h^*| > 0$ for all $i \in [n]$.

Formally, an R-SNN is a function defined as $g(x) = \sum_{h=1}^{\tau} w_h R(\alpha_h^T x + \beta_h) + w_0$, where $\alpha_h \in \mathbb{R}^d$, $\beta_h \in \mathbb{R}$, $w_h \in \mathbb{R}$ for all $h \in [\tau]$, $w_0 \in \mathbb{R}$, and the ReLU activation function $R(s) = \max\{s, 0\}$. Then, for given $f^*(x)$, the following equalities hold :

$$\psi(\alpha_h^{*T} x^i + \beta_h^*) = \frac{1}{\epsilon_h}[R(\alpha_h^{*T} x^i + \beta_h^* + \epsilon_h) - R(\alpha_h^{*T} x^i + \beta_h^* - \epsilon_h)] - 1, \ \forall i \in [n].$$

If we define $g^*(x)$ as

$$g^*(x) = \sum_{h=1}^{m} \frac{w_h^*}{\epsilon_h}[R(\alpha_h^{*T} x + \beta_h^* + \epsilon_h) - R(\alpha_h^{*T} x + \beta_h^* - \epsilon_h)] - \sum_{h=1}^{m} w_h^* + w_0^*.$$

Then, $g^*(x)$ is a sparsely connected R-SNN with $2m$ hidden nodes. This network has maximum $2m(k+1) + 2m + 1$ parameters (the number of variables of the associated optimization problem). Moreover, it is clear that $g^*(x)$ has the same loss as that of $f^*(x)$ since $f^*(x^i) = g^*(x^i)$ for all $i \in [n]$. Therefore, the optimal objective value of TP obtained with $\lambda = 0$ gives an upper bound on the optimal hinge loss that can be achieved by the R-SNN with $2m$ hidden nodes. That is, if SGD can find a global optimal solution to the associated training problem with the hinge loss and no regularization for the R-SNN, the corresponding optimal loss should not exceed the optimal objective value of TP with no regularization. Based on this result, we can measure the optimization performance of SGD.

4. The optimization performance of SGD for small-sized networks

To measure the optimization performance of SGD based on the previous results, computational tests on randomly generated dataset are performed. For each $n \in \{250, 500, 1000, 2000\}$ and d (from 4 up to 20), we generate 30 instances with $x^i \in [0, 1]^d$ and $y^i \in \{-1, 1\}$ for all $i \in [n]$. The average results are reported for each n and d. We tested k-SNN for $k = 1$ and $\lambda = 0$, and we used a generic optimization solver Xpress [10] to solve TP. For every instance, the optimal objective value of TP was 0, that is, the loss is 0 and the training accuracy is 100%. Figure 1 shows the average number of used hidden nodes (m). For the tested instances, m is around 25% of the number of samples on average and it is more dependent on n than d. For each instance, we constructed a sparsely connected R-SNN with $2m$ hidden nodes. The average numbers of parameters of of R-SNNs are presented as multiples of n in Figure 2(a). Each of constructed R-SNNs has around $0.5n$ hidden nodes and $1.5n$ parameters. Then, R-SNNs were trained using SGD up to $50,000$ epochs with the hinge loss and no regularization. The batch size was set to 32. We used Tensorflow [1] and the Adam optimizer [14] with its default setting. The results in terms of training accuracy are given in Figure 2.

The results given in Figure 2 show that SGD cannot find a global optimum on relatively small network even though the sizes of R-SNNs are sufficiently large to fit training samples precisely. Also, regardless of number of parameters, the optimization performance of SGD gets better as the dimension increases in each number of samples, and it gets worse as the number of samples increases in each dimension. This implies that its performance depends on the density of samples.

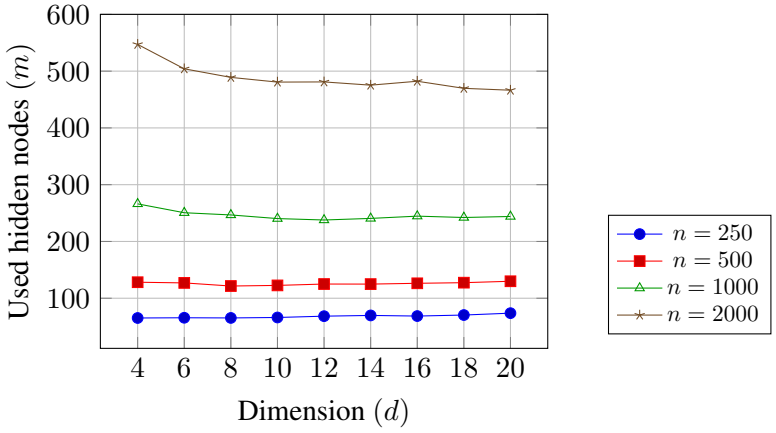
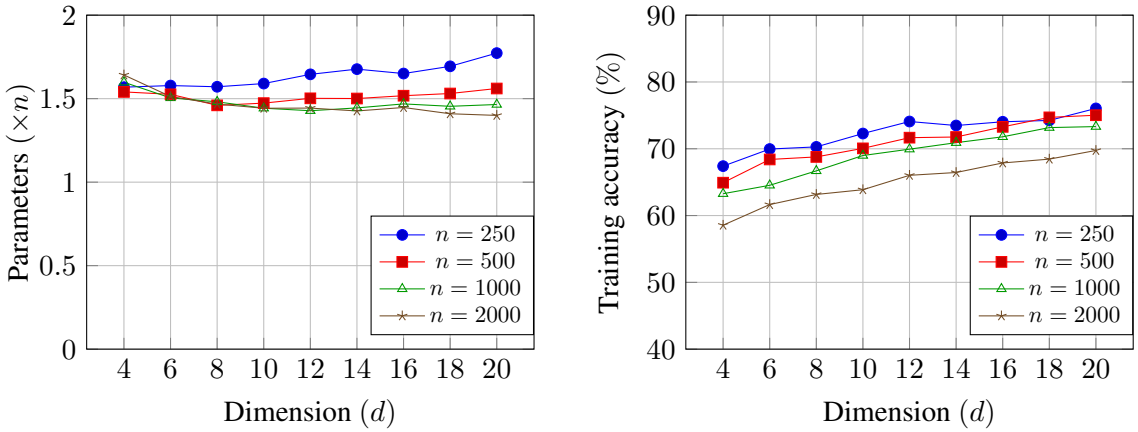


Figure 1: The number of hidden nodes of optimized 1-SNN



(a) The size of R-SNN

(b) Training accuracy

Figure 2: Training results for R-SNN with SGD

5. Conclusion

We consider a single hidden layer neural network with threshold activations, k-SNN. Even though the in-degree of each hidden node is restricted, k-SNN can fit any given finite samples under a mild assumption. A linear program with a polynomial number of variables and constraints to optimize k-SNN for binary classification problems were presented. We also showed that the optimized k-SNN can be transformed to a single hidden layer neural network with ReLU activations with the same loss. Based on the results, the optimization performance of SGD was computationally tested on randomly generated instances. The results show that SGD hardly converge to global optimum on small-sized network even though it is sufficiently large to fit training samples precisely. Also it depends on the density of training samples. We conjecture that these properties of SGD may affect its generalization performance. That is, limited optimization performance under certain condition may lead to good generalization by preventing overfitting. To prove this conjecture, more elaborate experiments or theoretical anaysis should be conducted further.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[3] Yoshua Bengio, Nicolas L. Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte. Convex neural networks. In *Advances in Neural Information Processing Systems 18*, pages 123–130. 2006.

[4] Yuan Cao and Quanquan Gu. Generalization error bounds of gradient descent for learning over-parameterized deep relu networks. *arXiv preprint arXiv:1902.01384*, 2019.

[5] Vašek Chvátal. *Linear Programming*. W.H. Freeman, 1983.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[7] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[8] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685, 2019.

[9] Clement Farabet, Camille Couprie, Laurent Najman, and Yann Lecun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013.

[10] FICO® Xpress Optimization, 2019. https://www.fico.com/en.

[11] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[12] Daniel Jakubovitz, Raja Giryes, and Miguel R. D. Rodrigues. Generalization error in deep learning. *arXiv preprint arXiv:1808.01174*, 2018.

[13] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk USSR*, 244(5):1093–1096, 1979.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[16] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems 31*, pages 8157–8166. 2018.

[17] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531, 2011.

[18] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[19] Tan Nguyen and Scott Sanner. Algorithms for direct 0–1 loss optimization in binary classification. volume 28 of *Proceedings of Machine Learning Research*, pages 1085–1093, 2013.

[20] Daniel S. Park, Jascha Sohl-Dickstein, Quoc V. Le, and Samuel L. Smith. The effect of network width on stochastic gradient descent and generalization: an empirical study. *arXiv preprint arXiv:1905.03776*, 2019.

[21] Tara N. Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618, 2013.

[22] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

[23] Jonathan J. Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems 27*, pages 1799–1807. 2014.

[24] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[25] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, 2020.

Appendix A. Proof for Theorem 1

Note that by the assumption on $\{x^i\}_{i=1}^n$, there exists $\hat{\alpha} \in \mathbb{R}^d$ such that $\hat{\alpha}_j = 0$ for all $j \notin K$ and $s_i \neq s_j$ for all $i, j \in [n]$ where $s_i = \hat{\alpha}^T x^i$. For the sake of simplicity, let $s_1 < s_2 < \cdots < s_n$. Then there exists $-\beta_i \in \mathbb{R},\ \forall i \in [n]$ such that $-\beta_1 < s_1 < -\beta_2 < s_2 < \cdots < -\beta_n < s_n$. Consider k-SNN with $\tau = n$ hidden nodes and suppose that each hidden node h represents a function $\psi\left(\hat{\alpha}^T x + \beta_h\right)$, i.e., $f(x) = \sum_{h=1}^n w_h \psi(\hat{\alpha}^T x + \beta_h)$. Let A be a $n \times n$ matrix where $A_{ij} = \psi(\hat{\alpha}^T x^i + \beta_j)$ for all $i, j \in [n]$. From the relationship between s_i's,

$$A = \begin{bmatrix} 1 & -1 & -1 & -1 & \cdots & -1 \\ 1 & 1 & -1 & -1 & \cdots & -1 \\ 1 & 1 & 1 & -1 & \cdots & -1 \\ & & \vdots & & & \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}.$$

Then the equality system, $y^i = f(x^i)$ for all $i \in [n]$, can be represented as $\mathbf{y} = A\mathbf{w}$ where $\mathbf{y} = [y^1, ..., y^n]$ and $\mathbf{w} = [w_1, ..., w_n]$. Since A has full rank, there always exists $\mathbf{w}$ such that $\mathbf{y} = A\mathbf{w}$ for arbitrary $\mathbf{y} \in \mathbb{R}^n$.