SGD: Un régularisateur implicite?

Simon Dufort-Labbé [1]

Abstract

Les réseaux de neurones (NN) entraînés avec la descente de gradient stochastique (SGD), ou une de ses variantes, possèdent une capacité de généralisation qui contraste fortement avec le comportement "typique" d'un algorithme d'apprentissage automatique (ML). Pour mieux cerner cette propriété, qui semble implicite, la présente revue présentera d'abord les résultats traditionnels sur la généralisation afin mieux les contraster avec les résultats obtenus en apprentissage profond. S'ensuivra ensuite une discussion de diverses hypothèses pouvant expliquer cet écart observé entre la capacité de généralisation théorique et pratique pour les réseaux de neurones entraînés avec la SGD.

1. La généralisation traditionnelle en ML

La généralisation, soit la capacité d'un algorithme ML à faire des prédictions sur un jeu de données (issu de la même distribution) qui n'a pas été utilisé pendant l'entraînement, est difficile à quantifier de manière exacte. Quelques outils nous permettent toutefois de mieux définir ce concept de "généralisation".

1.1. VC-Dimension

La VC-Dimension d'une classe d'hypothèse $\mathcal{H}$, dénotée $d_{\mathcal{H}}$ est définit comme étant la taille maximale d'un ensemble $C \subseteq \mathcal{X}$ pouvant être "shattered" par $\mathcal{H}$.
Cette définition est plus aisée à saisir dans le cas où $\mathcal{H}$ est un modèle de classification, f, paramétré par θ. On dit alors que f "shatter" un ensemble de données S si pour toute attribution de label possible, il existe θ tel que f ne fait aucune erreurs lorsque évalué sur S. La VC-dimension de f est le cardinal $\mathcal{D}$ maximal tel qu'il existe un ensemble de données de cardinalité $\mathcal{D}$ pouvant être "shattered" par f.

L'utilité de la VC-Dimension peut être mieux cerné en rappelant d'abord quelques concepts utiles en ML(Shalev-Shwartz & Ben-David, 2014):

1.1.1. Risque et risque empirique

En posant:

- $\ell : \mathcal{Y}^2 \mapsto \mathbb{R}$ est une fonction de coût, qui est la mesure choisie pour quantifier la justesse d'une prédiction.
- P est une distribution sur (x, y) (inconnue de l'algorithme)

Le but d'un algorithme est de trouver une fonction $h : \mathcal{X} \mapsto \mathcal{Y}$ qui minimise le *Risque*, définit comme

$$\mathcal{R}(h) := \mathbb{E}_{(x,y)\sim P}[\ell(h(x), y)]$$

Comme cette quantité est intractable, la stratégie employée par l'algorithme de ML sera de:

- Se limiter à l'exploration d'une famille de fonctions $\mathcal{H}$
- Sélectionner un échantillon $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ dans P
- Minimiser le *Risque Empirique* donné par

$$\mathcal{R}_S(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i) \text{ (Empirical Risk)}$$

1.1.2. Risque et VC-dimension

La minimisation du risque empirique pendant l'entraînement d'un algorithme de ML est justifiable par l'existence de la borne suivante, valide pour tout $\mathcal{H}$ avec une probabilité $\geq 1 - \eta$:

$$\sup_{h\in\mathcal{H}} |R(h) - R_S(h)| \leq \mathcal{O}(\sqrt{\frac{1}{2n}\log\left(\frac{2}{\eta}\right) + \frac{d_{\mathcal{H}}}{n}\log\left(\frac{n}{d_{\mathcal{H}}}\right)})$$

Il y a deux choses à remarquer de cette relation:

- Plus il y a de données (n), plus le risque empirique est proche du risque réel.
- On s'attend à ce que la différence entre le risque empirique et le risque réel croît avec $d_{\mathcal{H}}$.

[1]DIRO, Université de Montréal, ValerieLand, Canada. Correspondence to: Simon <simon.dufort-labbe@umontreal.ca >.

IFT 6269 project report - Université de Montréal

Finalement, considérons la décomposition suivante:

$$\begin{aligned}\mathcal{R}(h_S) &= \mathcal{R}_S(h_S) + \mathcal{R}(h_S) - \mathcal{R}_S(h_S) \\ &= \min_{h\in\mathcal{H}} \mathcal{R}_S(h) + \mathcal{R}(h_S) - \mathcal{R}_S(h_S) \\ &\leq \min_{h\in\mathcal{H}} \mathcal{R}_S(h) + \sup_{h\in\mathcal{H}} |\mathcal{R}(h) - \mathcal{R}_S(h)| \\ &\leq \underbrace{\min_{h\in\mathcal{H}} \mathcal{R}_S(h) + \mathcal{O}(\sqrt{\frac{1}{2n}\log\left(\frac{2}{\eta}\right) + \frac{d_\mathcal{H}}{n}\log\left(\frac{n}{d_\mathcal{H}}\right)})}_{\text{with probability } \geq 1-\eta}\end{aligned}$$

Qui permet d'exposer, encore une fois, un trade-off, du ML : Un plus grand $\mathcal{H}$ permet d'obtenir un plus petit $\mathcal{R}_S(h)$ mais *devrait mener à une plus grande différence entre le risque empirique et le risque réel.*

1.2. Capacité théorique des réseaux de neurones

On sait depuis 1989 (Cybenko, 1989) que les NN ont une capacité théorique qui leur permet d'estimer virtuellement n'importe quelle fonction. Comme ce résultat implique une taille potentiellement infinie pour le NN, il est très difficile d'obtenir un résultat pratique à partir de là.
Il est donc beaucoup plus intéressant de considérer le résultat suivant (Zhang et al., 2016):

Theorème 1 (Finite-Sample Expressivity) *There exists a two-layer neural network with ReLU activations and $2n + d$ weights that can represent any function on a sample of n examples in d dimensions.*

Ce qu'il faut comprendre ici, c'est que nous avons entre les mains une recette nous permettant de construire un NN pour lequel la VC-dimension de sa classe d'hypothèse (sa famille de fonctions) $\mathcal{H}$ est au moins égal à n ($d_\mathcal{H} \geq n$)!

1.3. Implications

Rappelons qu'un grand $d_\mathcal{H}$ (et donc un grand $\mathcal{H}$) mène traditionnellement à un très petit $\min_{h\in\mathcal{H}} \mathcal{R}_S(h)$, mais augmente aussi la probabilité d'avoir un grand écart entre le risque empirique et le risque réel.

Comme le précédent résultat théorique nous indique que les réseaux de neurones ont potentiellement une très grande capacité, le comportement attendu est que la performance de généralisation devrait se *dégrader* à partir du moment où la capacité effective du modèle passe un certain seuil. Ce comportement attendu est illustré à la figure 1.

Pour bien comprendre cette figure, il est important de réaliser que l'algorithme explore graduellement l'espace $\mathcal{H}$, à partir de son point d'initialisation. Sa capacité effective augmente donc graduellement avec le temps, et $d_\mathcal{H}$ est donc initialement limité, mais augmente rapidement. Une dégradation dans la performance est observée lorsque $d_\mathcal{H}$ devient trop élevé par rapport au jeu de données d'entraînement.

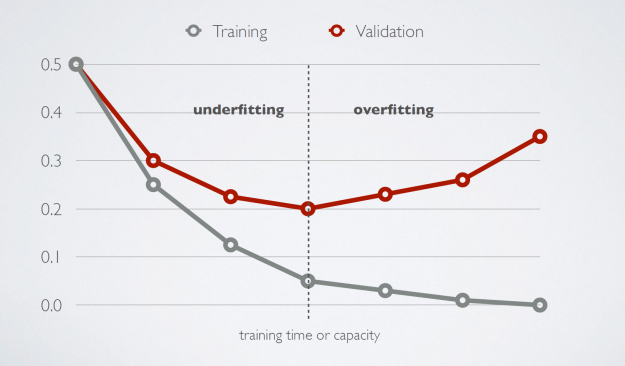

Figure 1. Comportement traditionnel attendu d'un algorithme de ML pour la performance de généralisation. Figure tirée des notes de cours du Dr Hugo Larochelle

2. Comportement observé

Toutefois, voici le comportement de généralisation observé pour un NN pour lequel on augmente graduellement la capacité:

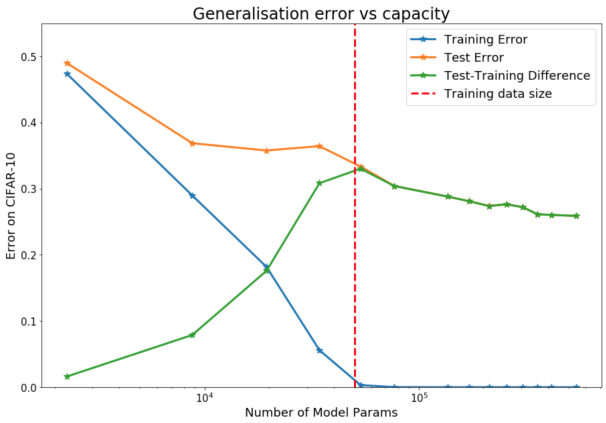

Figure 2. Dépendance de l'erreur de généralisation sur W (nombre total de paramètres) d'un modèle NN avec 5 couches de convolutions, entraîné sur un jeu fixe de N données d'entraînement. Aucune augmentation de données ni régularisation n'est effectuée. La théorie traditionnelle justifie le comportement de l'algorithme pour $N > W$, mais ne peut justifier qu'il n'y a pas de surentraînement lorsque $W \geq N$

Surprenamment, on observe qu'il n'y a pas de surentraînement alors que la capacité du modèle devient suffisamment élevée pour que la VC-dimension du modèle soit

au moins égale au cardinal de l'ensemble d'entraînement. Ce comportement ressemble grandement à celui d'un algorithme auquel on aurait inclus un régularisateur, limitant ainsi la capacité limite du NN.
Or, aucun régularisateur n'a été explicitement ajouté ici. Cela suggère la présence d'un régularisateur *implicite* dont nous aimerions identifié la source.

Pour trouver la provenance de ce régularisateur implicite, la structure de l'algorithme nous laisse, grosso modo, deux sources possibles : une affinité entre l'architecture du modèle et les données ou encore l'optimisateur utilisé pour l'entraînement (ici SGD). Or, ces expériences toutes simples nous permet d'écarter la première hypothèse: On constate

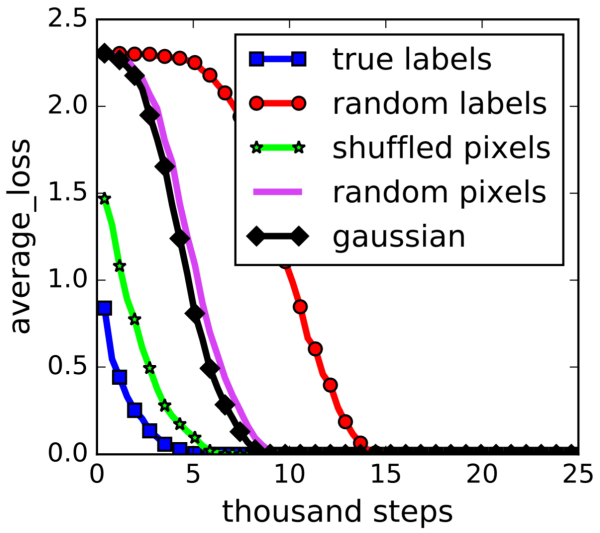

Figure 3. Entraînement d'un NN jusqu'au régime de surentraînement sur des jeux de données de plus en plus aléatoires. (Zhang et al., 2016)

que l'entraînement se déroule de la même manière (figure 3) lorsqu'il s'agit de modéliser des données de plus en plus aléatoires. La seule différence notoire est le temps requis (en époques) pour que le modèle parvienne au régime de surentraînement.
Mais surtout, on réalise que l'erreur de généralisation reste excellente lorsque le régime de surentraînement est atteint (figure 4), et ce même en présence de données qui sont de plus en plus corrompues.

Ces résultats suggèrent qu'il faut donc se tourner vers l'optimisateur pour trouver la source de la régularisation implicite.

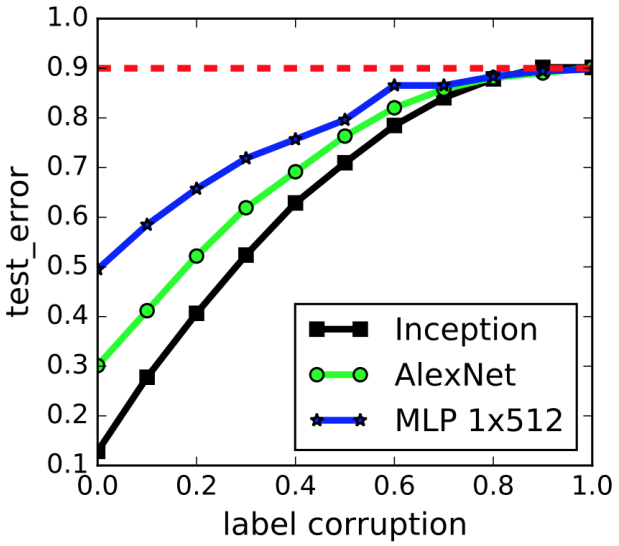

Figure 4. Croissance de l'erreur de généralisation pour différents modèles en régime de surentraînement selon le degré de corruption des données. (Zhang et al., 2016)

3. SGD préfère les minimas plus lisses?

L'argument mis de l'avant pour justifier que la SGD préférera une solution plus lisse (Zhang et al., 2017) est le rapprochement possible entre la mise à jour des paramètres via SGD et l'équation de Langevin.
En considérant la SGD comme étant la récursion suivante:

$$\theta_{k+1} = \theta_k - \alpha_k g(\theta_k, \xi_k)$$

où θ_k représente la valeur des paramètres du modèle à l'itération k, α_k est le taux d'apprentissage, choisi comme étant une séquence décroissante, et $g(\theta_k, \xi_k)$ est le gradient du coût θ_k pour une *mini-batch* sélectionnée aléatoirement dans l'ensemble d'entraînement. On peut réécrire la précédente récursion comme:

$$\theta_{k+1} = \theta_k - \alpha_k(\nabla F(\theta_k) + \rho_k)$$

avec

$$\rho_k = g(\theta_k, \xi_k) - \nabla F(\theta_k)$$

Et où $\nabla F(\theta_k)$ est le gradient calculé sur l'entièreté de l'ensemble d'entraînement. De là, on peut facilement constater la ressemblance avec l'équation discrète de Langevin, décrivant le mouvement stochastique d'une particule soumise à un bruit Brownien:

$$z_{t+1} = z_t + \epsilon \nabla F(z_t) + \zeta_t$$

Où ζ_t est le bruit qui se doit d'avoir *un comportement Brownien*. Sa forme la plus commune est une gaussienne centrée

autour de 0.
En supposant que $-\alpha_k \rho_k$ se comporte aussi comme un mouvement Brownien, le comportement asymptotique de la SGD devrait tendre vers la solution de l'équation de Langevin, soit une distribution de Boltzmann donnée par:

$$p(\theta) = \frac{1}{Z} \exp\left(-\frac{F(\theta)}{T}\right)$$

où Z est une constante de normalisation, $F(\theta)$ est le coût et T est relié à l'intensité du bruit.

Un tel comportement favoriserait effectivement les solutions situées dans une région optimale lisse (vis-à-vis un optimum de même valeur mais qui serait dans une région plus étroite) puisque la probabilité d'aboutir dans une telle région serait plus élevée. Ce comportement est illustré à la figure 5.

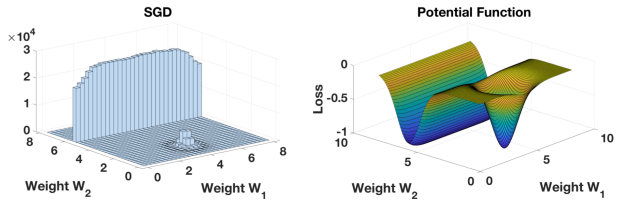

Figure 5. Histogramme de plusieurs optimisation via SGD (SGDL) sur la fonction de potentiel (coût) visible à droite. (Zhang et al., 2017)

3.0.1. Comportement Brownien du bruit

L'argument mis de l'avant pour justifier que la composante identifiée comme étant le bruit dans la récursion de la SGD agit de manière Brownienne est une conséquence du Théorème limite central (CLT). En effet, puisque $g(\theta_k, \xi_k)$ est une somme sur les variables aléatoire de la mini-batch, on doit s'attendre à ce que la distribution de $g(\theta_k, \xi_k)$ soit approximativement gaussienne.
L'expérience illustrée à la figure 6 montre que cette approximation tient dans certaines circonstances.

3.0.2. Effet sur la robustesse

L'intérêt d'atterrir dans une région de minima plus lisse et l'effet de régularisation qui s'ensuivrait repose sur le résultat suivant:

Theorème 2 *Assumons que pour une W_0 donné il existe un ensemble $\mathcal{X} \subseteq \mathbb{R}^d$ ainsi qu'un ensemble $\mathcal{W}' \subseteq \mathcal{W}$, $W \in \mathcal{W}'$ tel que:*

- $||\nabla^2_W F_W(x)||_\sigma \leq M \ \forall W \in \mathcal{W}', x \in \mathcal{X}$ ($||A||_\sigma$ *dénote la norme spectrale de A).*

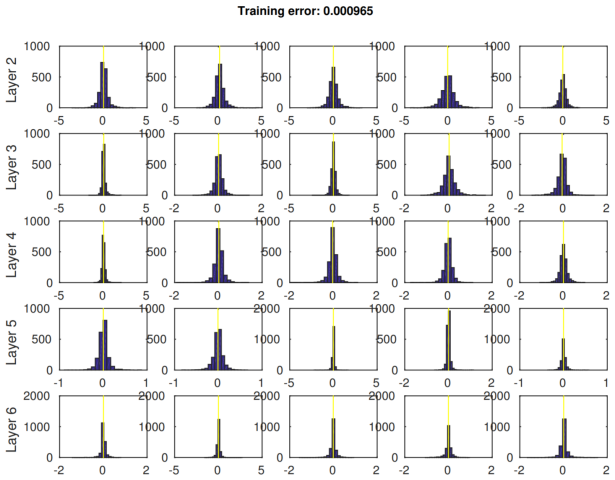

Figure 6. Histogrammes de quelques composants de $\nabla F(\theta_k, x_i)$ sur i pour un k fixé dans le régime asymptotique. (Zhang et al., 2017)

- *Il existe r, δ telle que la proposition suivante est vraie. Pour tout $x \in \mathcal{X}$ tel que $F_W(x)$ est différentiable vis-à-vis x et W pour tout W dans une ensemble ouvert U_x contenant W, il existe une boule $B(W_0, r) \subseteq \mathcal{W}'$ (mesurée selon la norme Euclidienne dans $\mathcal{W}$), tel que pour tout $W' \in B(w_0, r)$:*

$$|F'_W(x) - F_{W_0}(x)| \leq \delta$$

C'est à dire: une résistance à la variabilité des poids entraînerait une résistance à la variabilité des données et donc un effet de régularisation. Il est évident qu'une solution lisse serait plus résistante à une variation des poids (pour certaines dimensions) lorsque comparé à une solution à la SGD plus "pointue".

3.0.3. Critiques

Une critique majeure doit être apportée à ces conclusions: Si l'espace des paramètres comportent plusieurs minimas étroits (pointus) équivalents et très peu de minimas lisses et plats, alors il est possible que la probabilité de finir dans un minima étroit soit supérieure à celle de finir dans un minima lisse. Il est donc extrêmement hasardeux de tenter de conclure que la régularisation implicite est due à la présence de minima lisses sans en savoir plus sur les caractéristiques de l'espace des paramètres. Or, détailler l'entièreté de l'espace des paramètres est un problème infaisable à haute dimension.

Aussi, il a été démontré qu'on peut reparamétriser l'espace des paramètres de telle sorte qu'une solution identifiée comme ayant une bonne capacité de généralisation devient aussi étroite que désirée ((Dinh et al., 2017)). Cette

reparamétrisation préserve évidemment la capacité de généralisation de la solution.

4. SGD tend vers la solution de norme minimale

Le résultat suivant, démontré par (Soudry et al., 2017), se révèle beaucoup plus solide pour tenter d'expliquer la régularisation implicite de la SGD:

Theorème 3 *Pour tout ensemble de données linéairement séparable, pour toute fonction β-lisse de coût (inclut l'erreur logistique et peut être étendu aux moindres carrés) avec une queue exponentielle (peut être mise en sandwich entre deux fonctions exponentielles de la forme $(1+\exp(-ux))$ la descente de gradient convergera vers la solution de norme minimale:*

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 \quad \text{s.t. } \mathbf{w}^T\mathbf{x}_n > 1$$

De plus, ce résultat peut être étendu aux réseaux de neurones avec des fonctions d'activation ReLU (sous la condition que les ReLU se stabilisent après l'itération k_0 (cessent de changer de signe)):

$$\hat{\mathbf{w}}_l = \arg\min_{\mathbf{w}_l} ||\mathbf{w}_l||^2 \quad \text{s.t. } y_n u_n(\mathbf{w}) > 1$$

avec $u_n = \mathbf{W}_L\sigma(\mathbf{W}_{L-1}\sigma(...\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}_n)))$
(Voir (Soudry et al., 2017) pour la preuve)

Cette convergence vers la solution de norme minimale agit évidemment comme une régularisation implicite, et émule en fait les effets de la "ridge regression". Finalement, notons que la solution de norme minimale est aussi celle qui maximise la marge géométrique

$$\gamma^i = \frac{y^i(\mathbf{w}^T\mathbf{x}^i + b)}{||\mathbf{w}||_2}$$

sur un ensemble de données linéairement séparable (Voir figure 7). Une marge plus large peut expliquer la robustesse des NN entraînés avec la SGD, en offrant une plus grande marge d'erreur lors de la classification de nouveaux points situés dans cette marge.

5. Conclusion

La propriété de certains NN à tendre vers la solution de norme minimale s'avère extrêmement intéressante, surtout que cette propriété pourrait être généralisable à plusieurs architectures différentes. En soit, il s'agit d'un argument en faveur de l'utilisation de la SGD par rapport à d'autres optimisateurs, incluant Adam.

Aussi, bien que l'argument voulant que les NN favorisent les solutions se trouvant dans des régions de minimas lisses

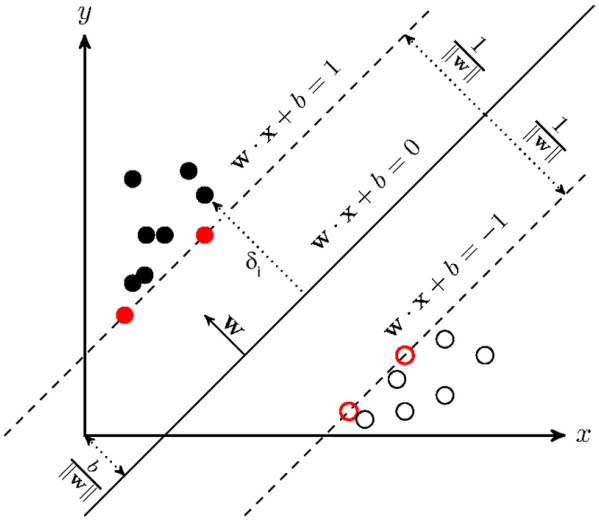

Figure 7. Exactement comme dans un SVM, la maximisation de la marge géométrique augmente la robustesse du modèle.

soit discutable, ce résultat pourrait tenir pour certaines architectures. Il serait très intéressant d'identifier le type d'architectures qui posséderaient cette propriété.

Finalement, il n'est pas encore possible de trancher définitivement sur l'origine de la régularisation implicite observée en apprentissage profond. Et il ne faut pas éliminer la possibilité qu'une multitude de facteurs pourraient s'accumuler pour expliquer complètement les capacités surprenantes de généralisation des NN entraînés avec la SGD.

Je reste convaincu qu'il s'agit d'un domaine de recherche pouvant se révéler encore très fécond, d'autant plus que de nouveaux optimisateurs restent certainement à découvrir et qu'il faudra étudier si eux aussi possèdent des caractéristiques leur permettant de présenter une régularisation implicite.

Code disponible

Le code ayant permis de produire la figure 2 est disponible via le Colab suivant: https://colab.research.google.com/drive/14E9st4LKcl36i15tJ6b8SDE-tLfsr6Xr

References

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989. ISSN 0932-4194. doi: 10.1007/BF02551274. URL http:

//dx.doi.org/10.1007/BF02551274.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets, 2017.

Shalev-Shwartz, S. and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014. ISBN 1107057132.

Soudry, D., Hoffer, E., and Srebro, N. The implicit bias of gradient descent on separable data. *J. Mach. Learn. Res.*, 19:70:1–70:57, 2017.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *ArXiv*, abs/1611.03530, 2016.

Zhang, C., Liao, Q., Rakhlin, A., Sridharan, K., Miranda, B., Golowich, N., and Poggio, T. Musings on deep learning: Properties of sgd. 04/2017 2017.