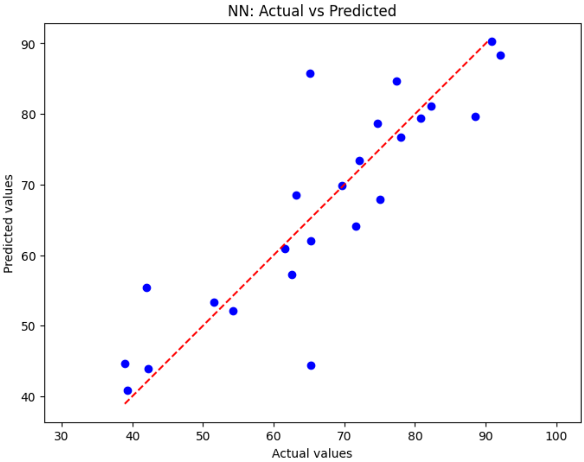
Why do overparameterized neural networks generalize?

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Tampere University

March 6 2024

A real-world example



- A very small noisy satellite data with 118 rows and 12 features. The picture shows the performance of a neural network with roughly six million parameters trained on 94 points.

Statistical learning theory

- **Goal:** given i.i.d. samples (measurements) $(x_1, y_1), \dots, (x_N, y_N)$ from an unknown probability distribution μ over $\mathcal{X} \times \mathcal{Y}$, find a predictor $f : \mathcal{X} \to \mathcal{Y}$ from a given set $\mathcal{F}$ for which the risk

$$R(f) := \mathbb{E}_{(x,y)\sim\mu}[\ell(f(x), y)]$$

is small. Here $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is some loss function.

Statistical learning theory

- **Goal:** given i.i.d. samples (measurements) $(x_1, y_1), \ldots, (x_N, y_N)$ from an unknown probability distribution μ over $\mathcal{X} \times \mathcal{Y}$, find a predictor $f : \mathcal{X} \to \mathcal{Y}$ from a given set $\mathcal{F}$ for which the risk

$$R(f) := \mathbb{E}_{(x,y)\sim\mu}[\ell(f(x), y)]$$

is small. Here $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is some loss function.

Example (Linear regression)

We have $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ and $\ell(y, y') := (y - y')^2$. Goal is to find a predictor f from

$$\mathcal{F} := \{\beta_1 x + \beta_0 \mid \beta_1, \beta_0 \in \mathbb{R}\}$$

with small risk.

Overfitting

- Since μ is unknown, we can not calculate $R(f)$ directly. In practice we estimate $R(f)$ by calculating its empirical risk with respect to our sample $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$:

$$R_S(f) := \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i)$$

Law of large numbers guarantees that almost surely $R_S(f) \to R(f)$ as $|S| \to \infty$.

Overfitting

- Since μ is unknown, we can not calculate $R(f)$ directly. In practice we estimate $R(f)$ by calculating its empirical risk with respect to our sample $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$:

$$R_S(f) := \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i)$$

  Law of large numbers guarantees that almost surely $R_S(f) \to R(f)$ as $|S| \to \infty$.

  - For example in the case of linear regression we minimize the mean squared error.

Overfitting

- Since μ is unknown, we can not calculate $R(f)$ directly. In practice we estimate $R(f)$ by calculating its empirical risk with respect to our sample $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$:

$$R_S(f) := \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i)$$

  Law of large numbers guarantees that almost surely $R_S(f) \to R(f)$ as $|S| \to \infty$.

  - For example in the case of linear regression we minimize the mean squared error.

- If $\mathcal{F}$ is sufficiently expressive w.r.t. S, the problem of finding $f \in \mathcal{F}$ with small $R_S(f)$ is ill-posed: there are many predictors in $\mathcal{F}$ that have a small empirical risk, but many of them will overfit.

Overfitting

- Since μ is unknown, we can not calculate $R(f)$ directly. In practice we estimate $R(f)$ by calculating its empirical risk with respect to our sample $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$:

$$R_S(f) := \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i)$$

  Law of large numbers guarantees that almost surely $R_S(f) \to R(f)$ as $|S| \to \infty$.

  - For example in the case of linear regression we minimize the mean squared error.

- If $\mathcal{F}$ is sufficiently expressive w.r.t. S, the problem of finding $f \in \mathcal{F}$ with small $R_S(f)$ is ill-posed: there are many predictors in $\mathcal{F}$ that have a small empirical risk, but many of them will overfit.

  - Consider for example the case where $\mathcal{X} = \mathcal{Y} = \mathbb{R}$, $\ell(y, y') = |y - y'|$ and

$$\mathcal{F} = \{\text{polynomials } \mathbb{R} \to \mathbb{R}\}.$$

    For finite $S \subseteq \mathbb{R}^2$ we can often find several $f \in \mathcal{F}$ for which $R_S(f) = 0$.

Overfitting

- Since μ is unknown, we can not calculate $R(f)$ directly. In practice we estimate $R(f)$ by calculating its empirical risk with respect to our sample $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$:

$$R_S(f) := \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i)$$

  Law of large numbers guarantees that almost surely $R_S(f) \to R(f)$ as $|S| \to \infty$.

  - For example in the case of linear regression we minimize the mean squared error.

- If $\mathcal{F}$ is sufficiently expressive w.r.t. S, the problem of finding $f \in \mathcal{F}$ with small $R_S(f)$ is ill-posed: there are many predictors in $\mathcal{F}$ that have a small empirical risk, but many of them will overfit.

  - Consider for example the case where $\mathcal{X} = \mathcal{Y} = \mathbb{R}, \ell(y, y') = |y - y'|$ and

$$\mathcal{F} = \{\text{polynomials } \mathbb{R} \to \mathbb{R}\}.$$

    For finite $S \subseteq \mathbb{R}^2$ we can often find several $f \in \mathcal{F}$ for which $R_S(f) = 0$.

- How to find a predictor which also has a small risk?

Example: support vector machines and ℓ_2-regularization

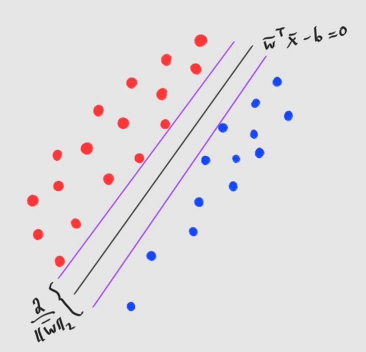
- Let $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{-1, 1\}$ and

$$\ell(y, y') = \begin{cases} 1 & \text{, if } y \neq y' \\ 0 & \text{, otherwise} \end{cases}$$

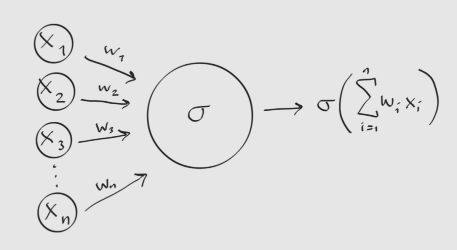
  Set

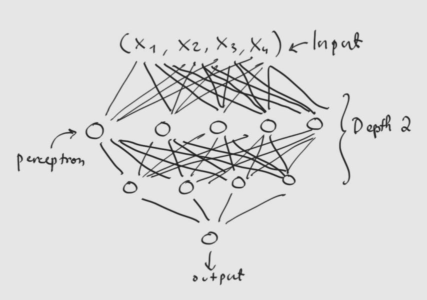
$$\mathcal{F} := \{\text{hyperplanes in } \mathbb{R}^d\}.$$

  If $S \subseteq \mathbb{R}^d \times \{-1, 1\}$ is linearly separable, then there are many predictors in $f \in \mathcal{F}$ for which $R_S(f) = 0$. The main idea in support vector machines is that we should select a hyperplane with a minimal norm.



Neural networks





How to train a neural network

- Neural network is always trained with fixed architecture, which leaves us with the task of learning good weights.

How to train a neural network

- Neural network is always trained with fixed architecture, which leaves us with the task of learning good weights.
- One way to learn them iteratively is to use gradient descent.

How to train a neural network

- Neural network is always trained with fixed architecture, which leaves us with the task of learning good weights.
- One way to learn them iteratively is to use gradient descent. At each step of the iteration the current vector of weights $\overline{w}$ is updated using the rule

$$\overline{w} \leftarrow \overline{w} - \eta \nabla R_S(\overline{w}),$$

where η is the learning rate.

How to train a neural network

- Neural network is always trained with fixed architecture, which leaves us with the task of learning good weights.
- One way to learn them iteratively is to use gradient descent. At each step of the iteration the current vector of weights $\overline{w}$ is updated using the rule

$$\overline{w} \leftarrow \overline{w} - \eta \nabla R_S(\overline{w}),$$

  where η is the learning rate.
- The main optimization algorithm that is used to train modern neural networks is the stochastic gradient descent (SGD), which is a "noisy" variant of GD.

How to train a neural network

- Neural network is always trained with fixed architecture, which leaves us with the task of learning good weights.
- One way to learn them iteratively is to use gradient descent. At each step of the iteration the current vector of weights $\overline{w}$ is updated using the rule

$$\overline{w} \leftarrow \overline{w} - \eta \nabla R_S(\overline{w}),$$

  where η is the learning rate.
- The main optimization algorithm that is used to train modern neural networks is the stochastic gradient descent (SGD), which is a "noisy" variant of GD. At each step of the iteration the current vector of weights $\overline{w}$ are updated as follows: go through the points x in the sample S in a random order and for each of them do

$$\overline{w} \leftarrow \overline{w} - \eta \nabla R_{\{x\}}(\overline{w})$$

Why do big neural networks generalize?

- Modern neural network architectures have far more trainable parameters than the number of points in the sample that is used to train them and yet the training algorithms are able to learn models that generalize well.

Why do big neural networks generalize?

- Modern neural network architectures have far more trainable parameters than the number of points in the sample that is used to train them and yet the training algorithms are able to learn models that generalize well.
- This seems to be the case even without any form of explicit regularization.

Why do big neural networks generalize?

- Modern neural network architectures have far more trainable parameters than the number of points in the sample that is used to train them and yet the training algorithms are able to learn models that generalize well.
- This seems to be the case even without any form of explicit regularization.

Example

The CIFAR-10 image classification benchmark has 50000 training examples spread across 10 classes. The following table demonstrates the performance of a common architecture, called Inception, on CIFAR-10. Inception has more than 1.5 million parameters.

ℓ_2-regularization	Train accuracy	Test accuracy
Yes	100.0	86.03
No	100.0	85.75

Table: Inception on CIFAR-10.

Why do big neural networks generalize?

- Modern neural network architectures have far more trainable parameters than the number of points in the sample that is used to train them and yet the training algorithms are able to learn models that generalize well.
- This seems to be the case even without any form of explicit regularization.

Example

The CIFAR-10 image classification benchmark has 50000 training examples spread across 10 classes. The following table demonstrates the performance of a common architecture, called Inception, on CIFAR-10. Inception has more than 1.5 million parameters.

ℓ_2-regularization	Train accuracy	Test accuracy
Yes	100.0	86.03
No	100.0	85.75

Table: Inception on CIFAR-10.

- Furthermore, architectures that have less parameters than the number of sample points seem to be more prone to overfit.

Implicit regularization

- Since explicit regularization is not needed when training overparameterized neural networks, several researches have suggested that SGD performs some form of implicit regularization.

Implicit regularization

- Since explicit regularization is not needed when training overparameterized neural networks, several researches have suggested that SGD performs some form of implicit regularization.
- The nature of this regularization is not clear and it seems to depend on SGD having a good initialization. E.g. Liu et al. (2019) gave a simple way of initializing the weights in such a way that SGD does not learn a “simple” model. (This was also mentioned in Nakkiran et al. (2019).)

Implicit regularization

- Since explicit regularization is not needed when training overparameterized neural networks, several researches have suggested that SGD performs some form of implicit regularization.
- The nature of this regularization is not clear and it seems to depend on SGD having a good initialization. E.g. Liu et al. (2019) gave a simple way of initializing the weights in such a way that SGD does not learn a "simple" model. (This was also mentioned in Nakkiran et al. (2019).)

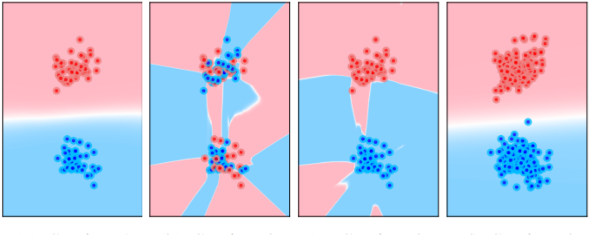


Figure: Picture from Liu et al. (2019)

Flat minimas

- Several researchers (e.g. Keskar et al. (2017) and Smith and Le (2018)) have suggested that SGD is able to find good models because its “noise” drives it towards “flat” minimas, which intuitively correspond to low complexity models.

Flat minimas

- Several researchers (e.g. Keskar et al. (2017) and Smith and Le (2018)) have suggested that SGD is able to find good models because its "noise" drives it towards "flat" minimas, which intuitively correspond to low complexity models.
- The correspondence between flat minimas and good models was emphasized already by Hochreiter and Schmidhuber (1997), who designed a learning algorithm that explicitly preferred flat minimas.

Flat minimas

- Several researchers (e.g. Keskar et al. (2017) and Smith and Le (2018)) have suggested that SGD is able to find good models because its “noise” drives it towards “flat” minimas, which intuitively correspond to low complexity models.
- The correspondence between flat minimas and good models was emphasized already by Hochreiter and Schmidhuber (1997), who designed a learning algorithm that explicitly preferred flat minimas.
- As far as I can tell, there is no water-proof explanation for why SGD is able to find flat minimas in the overparameterized region.

Nice loss landscape

- Somewhat surprisingly, Chiang et al. (2023) demonstrated empirically that in the over-parameterized regime "most" weights that work on the sample work also outside the sample. That is, in principle one can replace SGD with a random guessing.

Nice loss landscape

- Somewhat surprisingly, Chiang et al. (2023) demonstrated empirically that in the over-parameterized regime "most" weights that work on the sample work also outside the sample. That is, in principle one can replace SGD with a random guessing.
- More generally, they argue that the use of gradient-based optimizers is not the main source of generalization behavior of neural networks.

Nice loss landscape

- Somewhat surprisingly, Chiang et al. (2023) demonstrated empirically that in the over-parameterized regime “most” weights that work on the sample work also outside the sample. That is, in principle one can replace SGD with a random guessing.
- More generally, they argue that the use of gradient-based optimizers is not the main source of generalization behavior of neural networks.
- The experiments were limited to the small-sample regime, so verifying them in a more realistic setting is an interesting research direction.

That's all folks!

References

Chiang, P., Ni, R., Miller, D. Y., Bansal, A., Geiping, J., Goldblum, M., and Goldstein, T. (2023). Loss landscapes are all you need: Neural network generalization can be explained without the implicit bias of gradient descent. In *The Eleventh International Conference on Learning Representations*.

Hochreiter, S. and Schmidhuber, J. (1997). Flat Minima. *Neural Computation*, 9(1):1–42.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*.

Liu, S., Papailiopoulos, D., and Achlioptas, D. (2019). Bad global minima exist and SGD can reach them. In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*.

Nakkiran, P., Kaplun, G., Kalimeris, D., Yang, T., Edelman, B. L., Zhang, F., and Barak, B. (2019). SGD on neural networks learns functions of increasing complexity. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*.

Smith, S. L. and Le, Q. V. (2018). A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*.