Generalisation in fully-connected neural networks

With applications to time series forecasting

Anastasia Borovykh

CWI Amsterdam, the Netherlands

Deep learning symposium, Delft, April 2019

Based on joint work with S.M. Bohte and C.W. Oosterlee

Outline of the talk

- An introduction into neural networks
- The goal: obtaining good generalization
- Generalization in neural networks: what does it mean?
- Metrics of generalization
  - The weight Hessian
  - The input Hessian
- Controlling the generalization
  - Learning rate
  - Batch size
  - Number of training iterations
- Numerical examples

Neural networks

- Given an input $x \in \mathbb{R}^{n_0}$ and some target $y \in \mathbb{R}^{n_L}$ such that $(x, y) \sim \mathcal{D}$, the first layer a neural network computes

$$a^{(1)} = f(W^{(1)}x),$$

  where $f(\cdot)$ is the non-linear activation function, $a^{(1)} \in \mathbb{R}^{n_1}$ is the activation of the first layer and $W^{(1)} \in \mathbb{R}_{n_1 \times n_0}$ are the weights.
- Each subsequent layer $l = 2, ..., L$ outputs, $a^{(l)} = f(z^{(l)}) = f\left(W^{(l)}a^{(l-1)}\right)$.
- The final layer output is then given by,

$$\hat{y}(x, w) = W^{(L)}f(W^{(L-1)}...f(W^{(1)}x))...).$$

The goal: obtaining good generalization

Definition (Generalization)

The ability of a network trained on dataset A to perform well on dataset B

The problem:

1. many current deep learning algorithms lack guarantees and understanding of the solution complexity and quality
2. there is no systematic way of choosing the hyperparameters of the network architecture and optimization method

The goal:

1. obtain some insight into the quality of the network output, and its ability to perform well on unseen data
2. obtain insight into the effects of certain hyperparameters.

Generalization and optimization

- The neural network architecture can *partly* explain the generalization capabilities, i.e. CNNs, pooling, dropout.
- The work "Understanding deep learning requires rethinking generalization" by Zhang et al. (2016) showed that:

Deep nets are capable of memorising noise (due to overparametrisation) and thus perform bad out of sample, while the same network can learn on real data and perform well out of sample.

- How can this be?
  1. Generalization is not implicit to the network architecture, but it is related to *the way we train a neural network*
  2. **Controls exist**: Early stopping, a large learning rate and a small batch size in stochastic gradient descent can be used to *control* the smoothness of your output function, and thus the generalization capabilities.

Previous work on generalization

- A recent line of work has related the neural network loss surface to Gaussian random fields [Choromanska et al (2015)], [Dauphin et al (2014)], [Becker et al (2018)]
- Different ways of measuring these generalization capabilities exist:
  - The flatness of a minimum can be a good indicator of its ability to generalize [Hochreiter, Schmidhuber (1997)], [Keskar et al (2016)], [Srebro et al (2017)]
  - The smoothness of the learned function can also be an indicator [Novak et al (2018)]
  - Pac-Bayes theory can be used to bound the generalization error [Dziugaite and Roy (2017)]
  - Various norms of the network parameters can be used to measure the capacity of neural networks [Bartlett, (1998)] [Srebro et al (2015)], [Bartlett et al (2017)] [Srebro et al (2017)]
- An implicit bias is said to exist in stochastic training which regularizes the output [Srebro et al (2015)] [Jastrzkebski et al (2017)]

Generalization in neural networks

- Remember, we train the network by minimizing a loss function on a *train dataset*; i.e. given the training data (x^i, y^i), $i = 1, ..., N$, we minimize the squared loss,

$$\hat{\mathcal{L}}(x, w, y) = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}(x^i, w) - y^i \right)^2 .$$

- generalization is the relationship between a trained networks' performance on train data versus its performance on test data and is measured by the discrepancy between the true error and the error on the sample dataset,

$$\mathcal{L}(x, w, y) - \hat{\mathcal{L}}(x, w, y).$$

Generalization in neural networks

- A trained network is able to generalize well when it is not overfitting on noise in the train dataset: one aims to extract a meaningful pattern in the data instead of learning a flexible function that is able to fit all training points.
- One way to define the generalization capability is to study the robustness of the network with respect to input perturbations.

Generalization in neural networks

- A trained network is able to generalize well when it is not overfitting on noise in the train dataset: one aims to extract a meaningful pattern in the data instead of learning a flexible function that is able to fit all training points.
- One way to define the generalization capability is to study the robustness of the network with respect to input perturbations.
- For some input perturbation $\epsilon \in \mathbb{R}^{n_0}$, the change in the loss function should be small,

$$|\mathcal{L}(x+\epsilon, w, y) - \mathcal{L}(x, w, y)| < \delta.$$

- When the neural network is heavily overfitting on the noise, a small change in the input parameters might result in a large change in the neural network output.

Metrics for measuring generalization

- Generalization can thus be related to certain *smoothness* assumptions on the output function.
- It thus makes sense to look at metrics that measure this smoothness
- How can we measure the smoothness of a function? Using its derivatives.
- Intuition: if we want

$$|\mathcal{L}(x+\epsilon) - \mathcal{L}(x)| < \delta,$$

by a Taylor expansion,

$$\mathcal{L}(x+\epsilon) = \mathcal{L}(x) + \epsilon \nabla_x \mathcal{L}(x) + \frac{1}{2}\epsilon^2 \nabla_x^2 \mathcal{L}(x) + O(\epsilon^3),$$

so derivatives have to be small.

Metric one: the weight Hessian

- The Hessian with respect to the weights will be used in order to obtain insight on the noise robustness of the weights.
- The Hessian $H^w(\mathcal{L}) \in \mathbb{R}^{d \times d}$ of the loss function with respect to the weights has elements

$$h_{ij}^w = \partial_{w_i} \partial_{w_j} \mathcal{L}(x, w, y),$$

  which represents the rate of change of the derivative with respect to w_j in the direction of w_i.
- The weight Hessian gives insight into the *flatness* of a minimum. This in turn can be related to other generalization quantities such as minimum description length. [Hochreiter, 1997] or low information [Achille, 2018]
- We will use $Tr(H^w) = \sum_{i=1}^d h_{ii}^w$ as a metric of generalization.

Metric two: the input Hessian

- The Hessian with respect to the input measures the curvature of the output function or loss function with respect to a varying input.
- A Hessian with small eigenvalues means a smoother output function. Define the elements of the input Hessian $H^x(\mathcal{L}) \in \mathbb{R}^{n_0 \times n_0}$ of an input $x \in \mathbb{R}^{n_0 \times n_0}$ as,

$$h^x_{ij} = \partial_{x_i} \partial_{x_j} \mathcal{L}(x, w, y).$$

- The Hessian is averaged over the data samples x^i, $i = 1, ..., N$ in the train dataset in order to obtain an average sensitivity metric over the input space:

$$\mathbb{E}_x \left[Tr(H^x) \right] \approx \frac{1}{N} \sum_{i=1}^{N} Tr \left(H^{x^i} \right).$$

The relation between the input and weight Hessians

- Consider the output of a one-layer neural network $\hat{y}(x, w) = W^{(2)} f\left(W^{(1)} x\right)$.
- We have,

  $$\hat{y}(x+\epsilon, w) = W^{(2)} f(W^{(1)}(x+\epsilon)) = W^{(2)} f((W^{(1)} + \tilde{\epsilon})x), \quad \text{only if} \quad \tilde{\epsilon} = \frac{W^{(1)} \epsilon x^T}{||x||_2^2},$$

  where $\epsilon \in \mathbb{R}^{n_0}$ and $\tilde{\epsilon} \in \mathbb{R}^{n_0 \times n_1}$.
- Let $\hat{\epsilon}$ be the vectorised form of $\tilde{\epsilon}$. Then,

  $$\mathcal{L}(x+\epsilon, w, y) = \mathcal{L}(x, w + [\hat{\epsilon}, 0 I_{n_1}]^T, y), \tag{1}$$

  so that if the output is resistant to additive noise $\hat{\epsilon}$ in the first weight matrix $W^{(1)}$ then the output is resistant to additive noise ϵ in the input.
- Therefore, input noise resistance can be controlled through flatness in the weight space.

Controlling the generalization capabilities

- A standard way of optimising neural networks is through stochastic gradient descent (SGD):

$$w(t+1) = w(t) - \eta \nabla_w \mathcal{L}^S(x, w, y),$$

where $\mathcal{L}^S(x,w,y) = \frac{1}{M}\sum_{i\in\mathcal{S}}(\hat{y}(x^i,w) - y^i)^2$; in other words the gradient is considered over a *subset* $\mathcal{S}$ of the dataset.

- By the central limit theorem, if $(x^i, y^i) \sim \mathcal{D}$ i.i.d. then,

$$w(t+1) = w(t) = \eta \nabla_w \mathcal{L}(x, w, y) - \frac{\eta}{\sqrt{M}}\epsilon,$$

where $\epsilon \sim \mathcal{N}(0, K)$ with K being the covariance matrix of the noise, a function of the full and sample gradients.

- Note: the *amount* of noise is thus controlled by the learning rate η and the batch size M.

Controlling the generalization capabilities

- It is known that adding noise during the training algorithm can result in a *smoother* output function. Can we give some insight into why this happens?
- If convergence has been reached: $|\mathcal{L}(w(t+1) - w(t)| < \delta$.
- By a Taylor expansion for the loss function evaluated on the full training data we have

$$\mathcal{L}(w_{t+1}) = \mathcal{L}(w_t) - \left(\eta g - \frac{\eta}{\sqrt{M}}\epsilon\right)^T \nabla_w \mathcal{L}(w) + \frac{1}{2}\left(\eta g - \frac{\eta}{\sqrt{M}}\epsilon\right)^T H^w(\mathcal{L}(w_t))\left(\eta g - \frac{\eta}{\sqrt{M}}\epsilon\right),$$

  where H^w denotes the Hessian of the total loss w.r.t. the weights.
- Then using the convergence criteria we can obtain:

$$H^w(\mathcal{L}(w_t)) \approx \frac{\delta + 2\nabla_w \mathcal{L}(w_t)^T\left(\eta g - \frac{\eta}{\sqrt{M}}\epsilon\right)}{\left|\eta g - \frac{\eta}{\sqrt{M}}\epsilon\right|^2} = \frac{\delta + 2\nabla_w \mathcal{L}(w_t)^T\left(g - \frac{1}{\sqrt{M}}\epsilon\right)}{\eta\left|g - \frac{1}{\sqrt{M}}\epsilon\right|^2}.$$

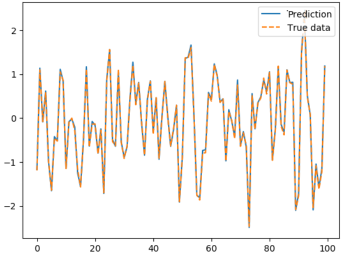
- In other words: at convergence the weight Hessian is small if a *large learning rate* or *small batch size* has been used. The learning rate and the batch size can thus be used to control the generalization.

Time series forecasting

- We have seen what generalization means, how to measure it and how to control this during training
- We are interested in applying these observations to time series forecasting, where we use the history to predict the future
- Time series have some properties that distinguish them from the other datasets:
  - non-i.i.d. data,
  - signal to noise ratio can be low,
  - non-stationary,
  - amount of data can be limited.

Numerical results: random noise

Consider data points x_i, $i = 1, ..., 100$ sampled from $\mathcal{N}(0,1)$. We use $(x_{i-5}, ..., x_i)$ to learn x_{i+1}.



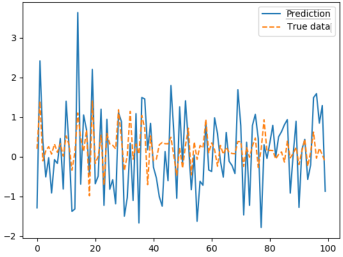
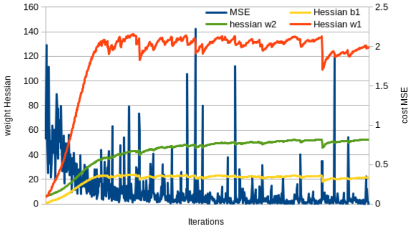


Figure: An overparametrized network can perfectly fit random noise. Left: after training; right: on the test set.

Numerical results: random noise

Consider data points x_i, $i = 1, ..., 100$ sampled from $\mathcal{N}(0, 1)$. We use $(x_{i-5}, ..., x_i)$ to learn x_{i+1}.



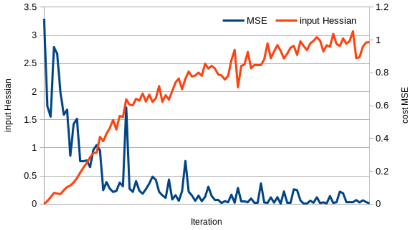
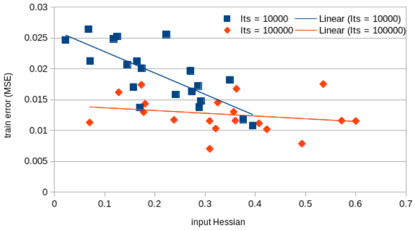


Figure: However, in order to learn this noise, the function *complexity*, measured here through the trace of the weight and input Hessian, has to significantly increase.

Numerical results: sine with noise

Consider now $x_i = \sin(0.1 t_i) + c\epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 1)$ and $t_i = 1, ..., 100$. We use $(x_{i-5}, ..., x_i)$ to learn x_{i+1}.



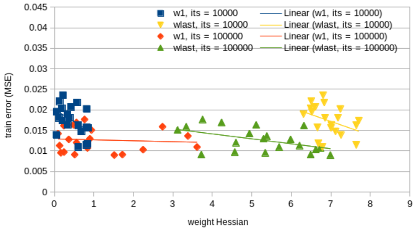
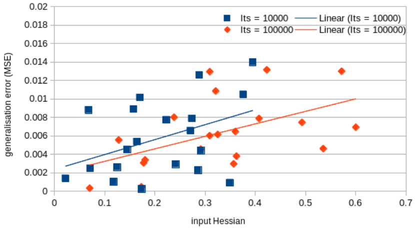


Figure: The higher the input or weight Hessian, the smaller the train error. Training longer results in functions of higher complexity.

Numerical results: sine with noise

Consider now $x_i = \sin(0.1 t_i) + c\epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0,1)$ and $t_i = 1, \ldots, 100$. We use $(x_{i-5}, \ldots, x_i)$ to learn x_{i+1}.



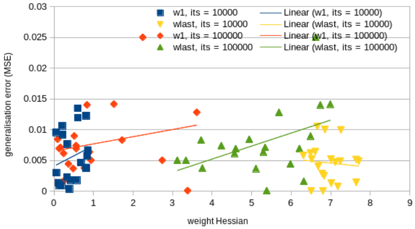


Figure: The higher the input or weight Hessian, the larger the generalization error. In other words, a large trace of the Hessian means worse noise resistance and higher function complexity. Training longer results in functions of higher complexity and thus worse generalization.

Numerical results: index forecasting

Consider the S&P500, the volatility index and the CBOE interest rate data.
The network output is the prediction for the next day return r_{t+1} of the S&P500 index using $n = 5$ historical daily return observations of the other time series.

N_{it}	N_b	MSE	hit rate	$Tr(H^x)$	$Tr(H^{W^{(1)}})$	$Tr(H^{W^{(3)}})$
10000	100	2.80	0.49	0.73	5.60	45.16
5000	100	2.65	0.513	0.73	4.86	45.48
10000	300	2.76	0.500	0.83	4.81	47.26
5000	300	2.71	0.505	0.74	4.50	46.60

Table: A high trace results in lower hit rate and higher MSE. Training longer, or using smaller batches results in a higher trace.

Numerical results: weather forecasting

We train a network on data of the daily minimum temperature in Melbourne, Australia.
The input data will consist of $n = 20$ historical daily observations of the temperature.

N_{it}	N_b	MSE	$Tr(H^x)$	$Tr(H^{W^{(1)}})$	$Tr(H^{W^{(3)}})$
10000	10	0.44	0.078	6.05	34.9
5000	10	0.36	0.023	5.85	27.5
10000	100	0.49	0.11	36.7	40.2
5000	100	0.36	0.030	38.4	31.9
10000	200	0.50	0.10	42.6	40.9
5000	200	0.37	0.032	56.7	31.7

Table: A higher trace of the input or weight Hessian corresponds to a worse test set MSE due to overfitting on the noise. As usual, training longer and using larger batch sizes resuls in more overfitting.

Conclusions

- The loss surface can be related to a Gaussian random field, for which in a high-dimensional setting a certain structure of critical points exists: low train error means high complexity.
- generalization in neural networks refers to a trade-off between function complexity and the function fit on the train data.
- We measure generalization capabilities through a certain noise-resistance or smoothness using two metrics: *the input and the weight Hessians.*
- We show that these metrics are good indicators of generalization capabilities.
- Certain hyperparameters used in the optimisation algorithm, in particular the learning rate, batch size and number of iterations, can be used as a control for the output function smoothness.

Future work

- Some theoretical proofs showing that the size of the Hessian is related to generalization capabilities exists (e.g. [Neyshabur et al (2017)] where they relate sharpness to PAC-Bayes theory).
- Extending these proofs to the non-i.i.d. case (e.g. assuming some dependencies such as mixing [Agarwal, Duchi (2013)]) could be of interest to obtain a theoretical proof of why sharpness is a good metric in the non-i.i.d. case
- The trace of the Hessian does not tell the full story [Dinh et al (2017)]; it is a *summarizing* statistic that discards some information: some directions can be sharp, others can be flat, what does this mean?
- Going beyond the trace and studying noise robustness and its relation to flatness in *specific* directions in weight space can be of interest.

References

- A. Borovykh, C.W. Oosterlee, S.M. Bohte, generalization in fully-connected neural networks for time series forecasting, (2019)
- A. Achille and S. Soatto, Emergence of invariance and disentanglement in deep representations, Journal of Machine Learning Research, 19 (2018), pp. 1-34.
- A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, The loss surfaces of multilayer networks, (2015), pp. 192-204.
- S. Hochreiter and J. Schmidhuber, Flat minima, Neural Computation, 9 (1997), pp. 1-42.
- N. Tishby and N. Zaslavsky, Deep learning and the information bottleneck principle, in Information Theory Workshop (ITW), 2015 IEEE, IEEE, 2015, pp. 1-5.
- S. Becker, Y. Zhang, and A. A. Lee, Geometry of energy landscapes and the optimizability of deep neural networks, arXiv preprint arXiv:1808.00408, (2018).
- R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, Sensitivity and generalization in neural networks: an empirical study, arXiv preprint arXiv:1802.08760, (2018).
- S. Jastrzkebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey, Three factors influencing minima in sgd, arXiv preprint arXiv:1711.04623, (2017)
- L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, Sharp minima can generalize for deep nets, arXiv preprint arXiv:1703.04933, (2017).

References

- B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks, Proceeding of the 28th Conference on Learning Theory (COLT), (2015)
- B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning Proceeding of the International Conference on Learning Representations workshop track, (2015)
- B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, Exploring generalization in deep learning, in Advances in Neural Information Processing Systems, (2017), pp. 5947-5956.
- P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. arXiv preprint arXiv:1706.08498, (2017)
- P. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE transactions on Information Theory, 44(2):525-536, (1998).
- G. K. Dziugaite and D. M. Roy, Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data, arXiv preprint arXiv:1703.11008, (2017).
- A. Agarwal and J. C. Duchi, The generalization ability of online algorithms for dependent data, IEEE Transactions on Information Theory, 59 (2013), pp. 573-587.