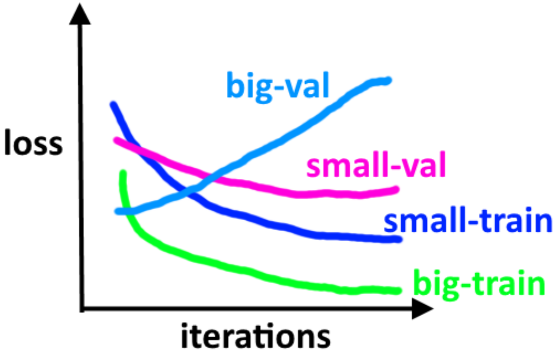
7.4 Regularization for Deep Learning

Machine Learning 1: Foundations

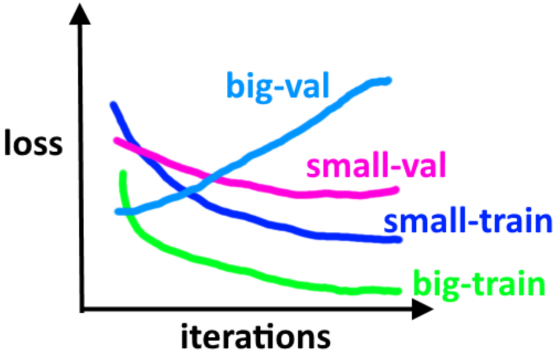
Marius Kloft *(TUK)*

Deep Neural Networks, Due to Their High Complexity, Are in Very High Danger of Overfitting



Adapted from `https://keras.rstudio.com/articles/tutorial_overfit_underfit.html`

Deep Neural Networks, Due to Their High Complexity, Are in Very High Danger of Overfitting



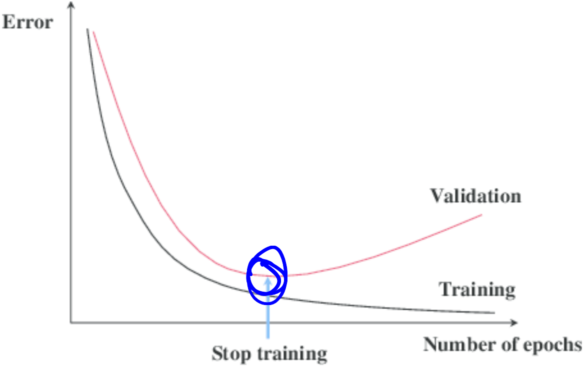
Deep learning requires specific regularization machinery—we will learn about some common strategies in the following

Adapted from `https://keras.rstudio.com/articles/tutorial_overfit_underfit.html`

Trick 1 for DL Regularization: Early Stopping

Idea:

- Split training set into a new (smaller) training set and a validation set
- Monitor the validation error every couple of mini-batch SGD iterations
- Stop SGD optimization early, when validation error goes up



A colleague just gave me this data ...

$$X := (\mathbf{x}_1, \ldots, \mathbf{x}_n) = \begin{pmatrix} 230.23 & 625.43 & 709.25 & 434.39 \\ 439.11 & 153.00 & 248.52 & 834.76 \\ \boxed{12.04} & \boxed{12.83} & \boxed{12.09} & \boxed{11.80} \\ 244.59 & 935.98 & 710.26 & 437.63 \\ \vdots & & & \vdots \end{pmatrix}$$

A colleague just gave me this data ...

$$X := (\mathbf{x}_1, \ldots, \mathbf{x}_n) = \begin{pmatrix} 230.23 & 625.43 & 709.25 & 434.39 \\ 439.11 & 153.00 & 248.52 & 834.76 \\ 12.04 & 12.83 & 12.09 & 11.80 \\ 244.59 & 935.98 & 710.26 & 437.63 \\ \vdots & & & \vdots \end{pmatrix}$$

What could be a problem with this data?

A colleague just gave me this data ...

$$X := (\mathbf{x}_1, \ldots, \mathbf{x}_n) = \begin{pmatrix} 230.23 & 625.43 & 709.25 & 434.39 \\ 439.11 & 153.00 & 248.52 & 834.76 \\ 12.04 & 12.83 & 12.09 & 11.80 \\ 244.59 & 935.98 & 710.26 & 437.63 \\ \vdots & & & \vdots \end{pmatrix}$$

What could be a problem with this data?

Not all features are on the same scale.

This is a serious problem because:

A colleague just gave me this data ...

$$X := (\mathbf{x}_1, \ldots, \mathbf{x}_n) = \begin{pmatrix} 230.23 & 625.43 & 709.25 & 434.39 \\ 439.11 & 153.00 & 248.52 & 834.76 \\ 12.04 & 12.83 & 12.09 & 11.80 \\ 244.59 & 935.98 & 710.26 & 437.63 \\ \vdots & & & \vdots \end{pmatrix}$$

What could be a problem with this data?

Not all features are on the same scale.

This is a serious problem because:

- Consider a linear classifier: $f(\mathbf{x}) = \text{sign}(\sum_{j=1}^{d} w_j x_j + b)$

A colleague just gave me this data ...

$$X := (\mathbf{x}_1, \ldots, \mathbf{x}_n) = \begin{pmatrix} 230.23 & 625.43 & 709.25 & 434.39 \\ 439.11 & 153.00 & 248.52 & 834.76 \\ 12.04 & 12.83 & 12.09 & 11.80 \\ 244.59 & 935.98 & 710.26 & 437.63 \\ \vdots & & & \vdots \end{pmatrix}$$

What could be a problem with this data?

Not all features are on the same scale.

This is a serious problem because:

- Consider a linear classifier: $f(\mathbf{x}) = \text{sign}(\sum_{j=1}^{d} w_j x_j + b)$
- Small-scale features $x_j \approx 0$ contribute to $f(\mathbf{x})$

$w_1 x_1 + w_2 x_2 + w_3 x_3$

$+ \ldots$

A colleague just gave me this data ...

$$X := (\mathbf{x}_1, \ldots, \mathbf{x}_n) = \begin{pmatrix} 230.23 & 625.43 & 709.25 & 434.39 \\ 439.11 & 153.00 & 248.52 & 834.76 \\ 12.04 & 12.83 & 12.09 & 11.80 \\ 244.59 & 935.98 & 710.26 & 437.63 \\ \vdots & & & \vdots \end{pmatrix}$$

What could be a problem with this data?

Not all features are on the same scale.

This is a serious problem because:

- Consider a linear classifier: $f(\mathbf{x}) = \mathrm{sign}(\sum_{j=1}^{d} w_j x_j + b)$
- Small-scale features $x_j \approx 0$ contribute to $f(\mathbf{x})$ only if their weight w_j is very large (to compensate for the small x_j)

A colleague just gave me this data ...

$$X := (\mathbf{x}_1, \ldots, \mathbf{x}_n) = \begin{pmatrix} 230.23 & 625.43 & 709.25 & 434.39 \\ 439.11 & 153.00 & 248.52 & 834.76 \\ 12.04 & 12.83 & 12.09 & 11.80 \\ 244.59 & 935.98 & 710.26 & 437.63 \\ \vdots & & & \vdots \end{pmatrix}$$

What could be a problem with this data?

Not all features are on the same scale.

This is a serious problem because:

- Consider a linear classifier: $f(\mathbf{x}) = \text{sign}(\sum_{j=1}^{d} w_j x_j + b)$
- Small-scale features $x_j \approx 0$ contribute to $f(\mathbf{x})$ only if their weight w_j is very large (to compensate for the small x_j)
- But w_j cannot become large because the regularizer $\frac{1}{2} \|\mathbf{w}\|^2$ promotes small w_js

Solution: Standardization

Solution: Standardization

Notation:

- For a feature f, denote by $f_1, \ldots, f_n \in \mathbb{R}$ its values for the data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- Denote $\mu_f := \frac{1}{n} \sum_{i=1}^{n} f_i$ ("**mean**")
- and $\sigma_f := \sqrt{\frac{1}{n} \sum_{i=1}^{n} (f_i - \mu_f)^2}$ ("**standard deviation**")

1 2 3

1.5 3 4

2.875

Solution: Standardization

Notation:

- For a feature f, denote by $f_1, \ldots, f_n \in \mathbb{R}$ its values for the data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- Denote $\mu_f := \frac{1}{n} \sum_{i=1}^{n} f_i$ ("**mean**")
- and $\sigma_f := \sqrt{\frac{1}{n} \sum_{i=1}^{n} (f_i - \mu_f)^2}$ ("**standard deviation**")

Standardization

1. Center each feature: $\forall i = 1, \ldots, n: \ f_i \leftarrow f_i - \mu_f$
2. Normalize the spread of each feature: $\forall i = 1, \ldots, n: \ f_i \leftarrow f_i / \sigma_f$

Solution: Standardization

Notation:

- For a feature f, denote by $f_1, \ldots, f_n \in \mathbb{R}$ its values for the data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- Denote $\mu_f := \frac{1}{n} \sum_{i=1}^{n} f_i$ ("**mean**")
- and $\sigma_f := \sqrt{\frac{1}{n} \sum_{i=1}^{n} (f_i - \mu_f)^2}$ ("**standard deviation**")

Standardization

1. Center each feature: $\forall i = 1, \ldots, n: \; f_i \leftarrow f_i - \mu_f$
2. Normalize the spread of each feature:
   $\forall i = 1, \ldots, n: \; f_i \leftarrow f_i / \sigma_f$

After standardization, the mean of each feature is 0 and the standard deviation is 1.

Solution: Standardization

Notation:

- For a feature f, denote by $f_1, \ldots, f_n \in \mathbb{R}$ its values for the data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- Denote $\mu_f := \frac{1}{n}\sum_{i=1}^{n} f_i$ ("**mean**")
- and $\sigma_f := \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f_i - \mu_f)^2}$ ("**standard deviation**")

Standardization

1. Center each feature: $\forall i = 1, \ldots, n: \; f_i \leftarrow f_i - \mu_f$
2. Normalize the spread of each feature:
   $\forall i = 1, \ldots, n: \; f_i \leftarrow f_i / \sigma_f$

After standardization, the mean of each feature is 0 and the standard deviation is 1.

This motivates a form of regularization for deep learning

Trick 2 for DL Regularization: Batch Normalization

Batch normalization takes feature standardization to the next level and standardizes **every neuron**:

Batch Normalization

Denote, for a neuron f, its activation values on a mini-batch by $\mathbf{f} = (f_1, \ldots f_B)$. Then, for each mini-batch in the SGD optimization and every neuron in the ANN, do:

1. Center each neuron activation: $\forall i = 1, \ldots, B: \; f_i \leftarrow f_i - \mu_f$
2. Normalize the spread: $\quad f_i \leftarrow f_i / \sigma_f$

Trick 2 for DL Regularization: Batch Normalization

Batch normalization takes feature standardization to the next level and standardizes **every neuron**:

Batch Normalization

Denote, for a neuron f, its activation values on a mini-batch by $\mathbf{f} = (f_1, \ldots f_B)$. Then, for each mini-batch in the SGD optimization and every neuron in the ANN, do:

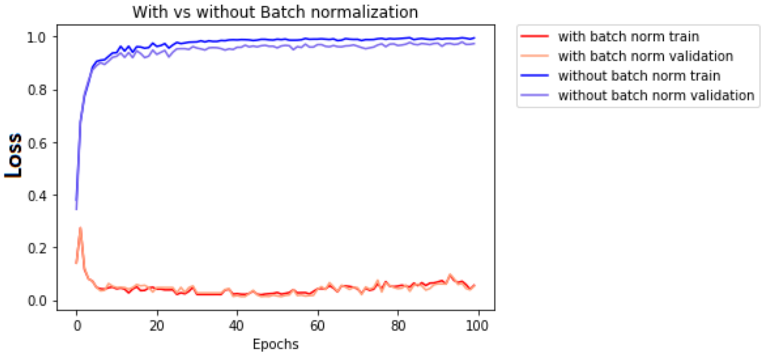
1. Center each neuron activation:
   $\forall i = 1, \ldots, B: \; f_i \leftarrow f_i - \mu_f$
2. Normalize the spread: $\quad f_i \leftarrow f_i / \sigma_f$

Overwrite the neuron's activation by:

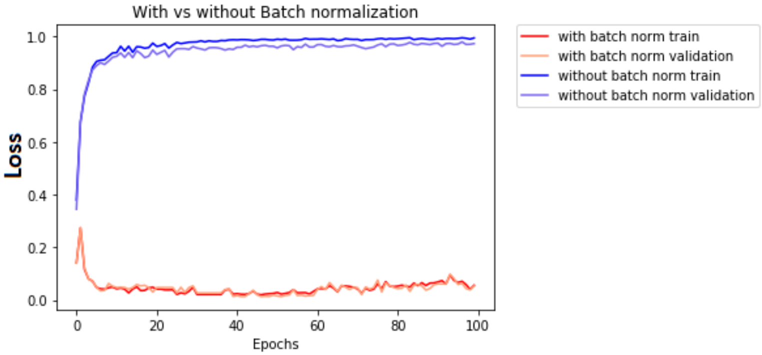
- $\gamma \mathbf{f} + \beta$
- Here $\gamma \in \mathbb{R}_+$ and $\beta \in \mathbb{R}$ are parameters that are learned during optimization

- The last step ensures that we can obtain the same range of activations in the optimization

Does Batch Normalization Work?

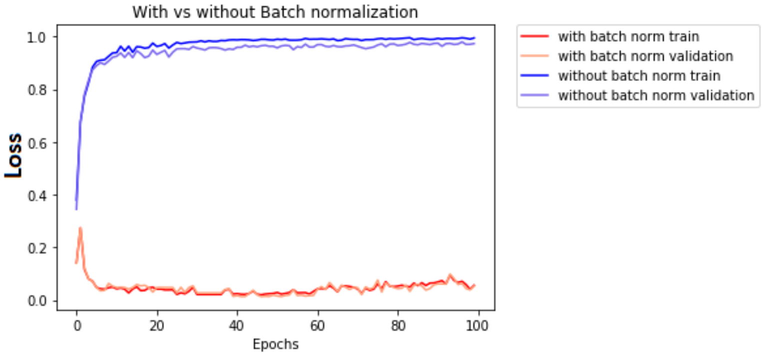


Does Batch Normalization Work?



Why does it work?

Does Batch Normalization Work?

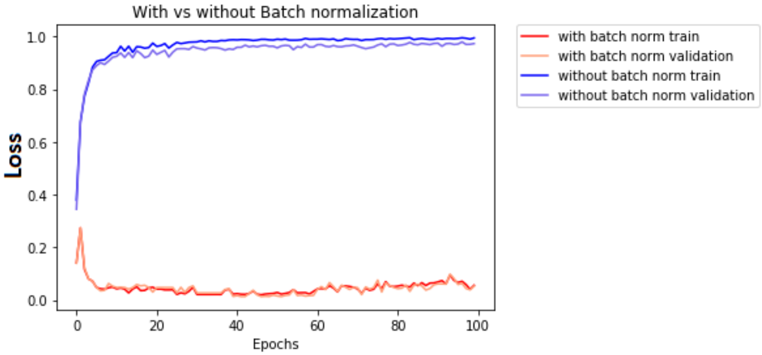


Why does it work?

We observe:

- BN just re-parametrizes the ANN optimization problem (same optimal solution)

Does Batch Normalization Work?



Why does it work?

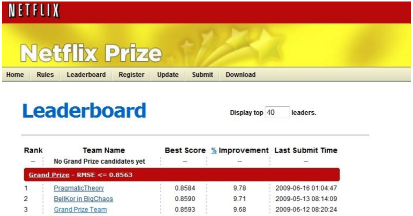
We observe:

- BN just re-parametrizes the ANN optimization problem (same optimal solution)

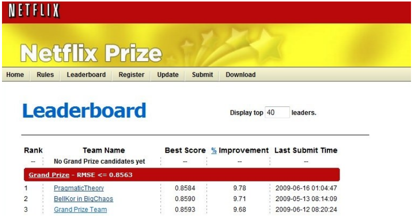
Unclear why it works... this is a pressing research topic.

Netflix Price

Netflix Price

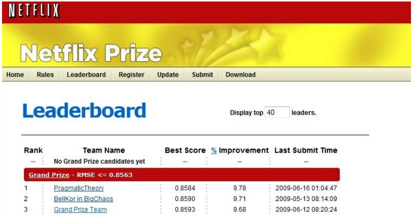


Netflix Price



How did they win the challenge?

Netflix Price



NETFLIX

Netflix Prize

Home | Rules | Leaderboard | Register | Update | Submit | Download

Leaderboard

Display top 40 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
Grand Prize - RMSE <= 0.8563				
1	PragmaticTheory	0.8584	9.78	2009-06-16 01:04:47
2	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
3	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24

How did they win the challenge?

The Netflix challenge and many others have been won using a technique called **ensembling**.

Ensemble Methods

Ensemble methods are methods that aggregate the prediction from multiple predictors

Ensemble Methods

Ensemble methods are methods that aggregate the prediction from multiple predictors

Example of a simple ensemble method:

1. Train k many of machine learning algorithms, resulting in prediction functions $f_1, \ldots, f_k$
2. Predict by majority vote: $f(\mathbf{x}) = +1$ if $|i : f_i(\mathbf{x}) = +1| \geq |i : f_i(\mathbf{x}) = -1|$ and $f(\mathbf{x}) = -1$ elsewise

Ensemble Methods

Ensemble methods are methods that aggregate the prediction from multiple predictors

Example of a simple ensemble method:

1. Train k many of machine learning algorithms, resulting in prediction functions $f_1, \ldots, f_k$
2. Predict by majority vote: $f(\mathbf{x}) = +1$ if $|i : f_i(\mathbf{x}) = +1| \geq |i : f_i(\mathbf{x}) = -1|$ and $f(\mathbf{x}) = -1$ elsewise

How can we use ensembling for deep learning?

Ensemble Methods

Ensemble methods are methods that aggregate the prediction from multiple predictors

Example of a simple ensemble method:

1. Train k many of machine learning algorithms, resulting in prediction functions $f_1, \ldots, f_k$
2. Predict by majority vote: $f(\mathbf{x}) = +1$ if $|i : f_i(\mathbf{x}) = +1| \geq |i : f_i(\mathbf{x}) = -1|$ and $f(\mathbf{x}) = -1$ elsewise

How can we use ensembling for deep learning?

Problem: too costly to train multiple deep ANNs

Trick 3 for DL Regularization: Dropout

Dropout regularization

Dropout regularization randomly removes in each mini-batch SGD iteration a fixed percentage of randomly selected neurons of the network.

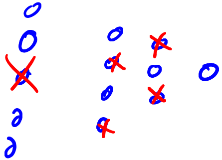
Trick 3 for DL Regularization: Dropout

Dropout regularization

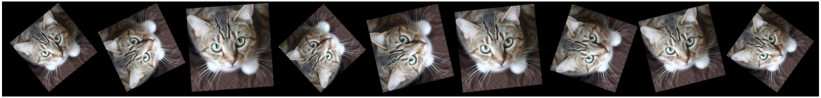
Dropout regularization randomly removes in each mini-batch SGD iteration a fixed percentage of randomly selected neurons of the network.

Common choice:

- input neurons are dropped out with probability $p = 0.2$
- hidden neurons are dropped out with probability $p = 0.5$



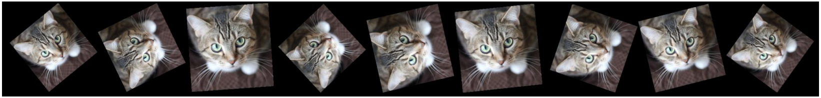
Trick 4 for DL Regularization: Data Augmentation



Idea:

- Say we wanna classify images, e.g., cats vs. dogs
- If we turn an image showing a cat upside down, it will still show a cat—just with the cat being upside down
- More generally, whatever an image shows, it will still show it after we apply a rotation or flip

Trick 4 for DL Regularization: Data Augmentation

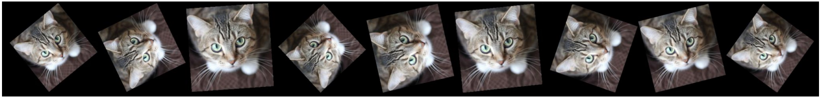


Idea:

- Say we wanna classify images, e.g., cats vs. dogs
- If we turn an image showing a cat upside down, it will still show a cat—just with the cat being upside down
- More generally, whatever an image shows, it will still show it after we apply a rotation or flip

This means we can generate, from our training points, many more training points by applying transformations.

Trick 4 for DL Regularization: Data Augmentation



Idea:

- Say we wanna classify images, e.g., cats vs. dogs
- If we turn an image showing a cat upside down, it will still show a cat—just with the cat being upside down
- More generally, whatever an image shows, it will still show it after we apply a rotation or flip

This means we can generate, from our training points, many more training points by applying transformations.

This regularization strategy is known as **data augmentation**

Trick 5 for DL Regularization: Transfer Learning

Again, say we wanna classify images, but we are given only a small dataset

Trick 5 for DL Regularization: Transfer Learning

Again, say we wanna classify images, but we are given only a small dataset

Idea: there are huge amounts of images on the web (e.g., flickr)

Trick 5 for DL Regularization: Transfer Learning

Again, say we wanna classify images, but we are given only a small dataset

Idea: there are huge amounts of images on the web (e.g., flickr)

How can we exploit this?

Trick 5 for DL Regularization: Transfer Learning

Again, say we wanna classify images, but we are given only a small dataset

Idea: there are huge amounts of images on the web (e.g., flickr)

How can we exploit this?

1. Pre-training: Download an ANN from the web that was pre-trained on huge amounts of images
2. Fine-tuning: Run SGD optimization on our small dataset, but using the ANN from the web as starting point of the optimization ("fine-tuning")

Trick 5 for DL Regularization: Transfer Learning

Again, say we wanna classify images, but we are given only a small dataset

Idea: there are huge amounts of images on the web (e.g., flickr)

How can we exploit this?

1. Pre-training: Download an ANN from the web that was pre-trained on huge amounts of images
2. Fine-tuning: Run SGD optimization on our small dataset, but using the ANN from the web as starting point of the optimization ("fine-tuning")

This regularization strategy is known as **transfer learning**—that is, transfering information into a learning problem from a related problem

Conclusion

The problem: overfitting dilemna

- do not know whether simple or complex model is adequate

The solution:

- choose complex model and use regularization

Regularization techniques:

- Norm regularization
  - makes solutions more smooth
- Batch normalization
  - standardize each neuron
- Dropout
  - randomly removing neurons from the architecture in each SGD iteration

Outlook:

- Another type of regularization: adversarial training