


Deep Learning in Practice

Philipp Seegerer
Technische Universität Berlin - Machine Learning Group

Beginners Workshop Machine Learning 2019

Agenda

Hyperparameters and Optimization

Controlling Overfitting

Tips, Tricks and Misc

Summary

Motivation

- Large number of parameters → overfitting
- Non-convex, high-dimensional optimization with SGD → local optima
- Learning from small datasets
- **Important:** Generalization error

Agenda

Hyperparameters and Optimization

Controlling Overfitting

Tips, Tricks and Misc

Summary

Training Curve

- Visualize evolution of learning
- Performance metrics: loss, accuracy, precision, recall etc.
- Training and **validation** performance

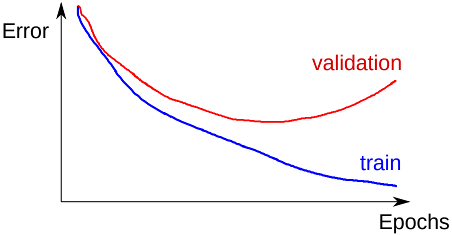

Figure: Typical SGD training curve for neural networks.

Learning Rate

- Far too high: divergence
- Too high: Fast initial improvement but convergence to bad optimum
- Appropriate: typically exponentially shaped
- Too low: Linear improvement
- Search in logarithmic space
- Typical value: 10^{-2}

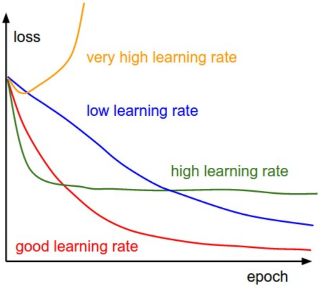

Figure: *Image taken from* `http://cs231n.github.io/neural-networks-3/`.

Learning Rate Scheduling

- High initial learning rate
- Decrease to find minimum more accurately
- Variants:
  - Step
  - Exponential
  - $^1/_t$
  - Reduction on plateau

Batch Size

- Control "stochasticity" of SGD $\rightarrow$ "noise" in training curve
- Influence mainly on training speed, not test performance
- Typical value: 64

Parameter Initialization

- Break symmetry by random initialization
- Size of weights should depend on number of input connections
- Method by Glorot and Bengio 2010 (“Xavier initialization”)

Momentum

- "Smooth out" oscillations in training curve
- Moving average of past gradient values
- Typical value: $\mu = 0.9$ (can be lower in the beginning)

$$\Lambda_{\text{mom}}^{t+1} = \mu\Lambda_{\text{mom}}^{t} - \lambda \cdot \nabla E\left(\Theta^{t}\right) \quad , \tag{1}$$

where μ is the momentum parameter, Λ is the parameter update, Θ is the parameter vector and λ is the learning rate.

Advanced Update Functions

Automatic adaptation of learning rate

- AdaGrad: Duchi, Hazan, and Singer 2011
- AdaDelta: Zeiler 2012
- Adam: Kingma and Ba 2014
- RMSprop: Tieleman and Hinton 2012
- Nesterov's Accelerated Gradient (NAG): Nesterov 1983

Advanced Update Functions

Automatic adaptation of learning rate

- AdaGrad: Duchi, Hazan, and Singer 2011
- AdaDelta: Zeiler 2012
- **Adam**: Kingma and Ba 2014
- RMSprop: Tieleman and Hinton 2012
- Nesterov's Accelerated Gradient (NAG): Nesterov 1983

Hyperparameter Search

- Grid search
- Random search (Bergstra and Bengio 2012)
- Bayesian hyperparameter optimization

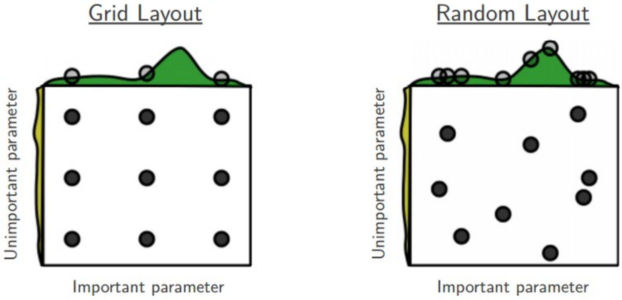

Figure: Image from Bergstra and Bengio 2012.

Agenda

Hyperparameters and Optimization

Controlling Overfitting

Tips, Tricks and Misc

Summary

Early Stopping

- Select model with best validation performance
- Stop training if validation performance has not improved for some epochs (“patience”)

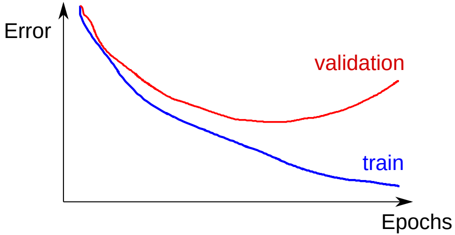

Figure: Early stopping.

Early Stopping

- Select model with best validation performance
- Stop training if validation performance has not improved for some epochs (“patience”)

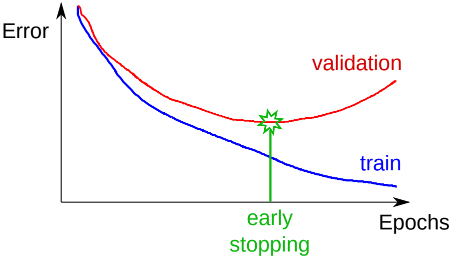

Figure: Early stopping.

Dropout

- Co-adapting neurons: “errors” of certain neurons may be compensated by others
- Dropout neurons randomly during training (Srivastava et al. 2014)
- Ensemble of pruned networks
- Averaging in test phase
- Takes longer to train (2-3 times)
- Dropout ratio ρ: probability that neuron is kept in SGD iteration
- Typical value: $\rho = 0.5$

Weight Regularization

- Regularize length of parameter vector
- L_1 and L_2
- "Weight decay": regularization strength
- Typical value: 10^{-3}

$$J = E(\Theta) + \omega \left\|\Theta\right\|_2^2 \rightarrow \min \ , \tag{2}$$

where J is the objective function, E is the cost function, ω is the weight decay and Θ are the parameters of the model.

Data Augmentation

- More data always reduces overfitting
- Generate synthetic data by applying transformations, e. g. for images:
  - random crops
  - mirroring
  - rotation
  - elastic deformations
  - color augmentation

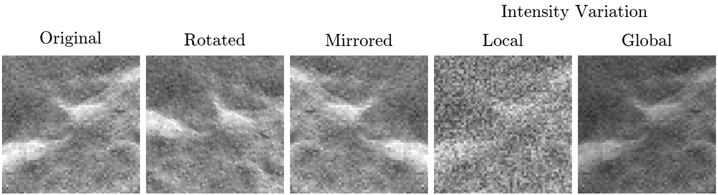

Figure: Image from Seegerer 2017.

Transfer Learning

- Extract knowledge from *source task* and apply to *target task*
- Pretraining and finetuning $\rightarrow$ pretrained parameters as initialization
- Unsupervised and supervised
- Narrow down exploration space
- Acts as regularizer

Agenda

Hyperparameters and Optimization

Controlling Overfitting

Tips, Tricks and Misc

Summary

Standardization

- Features should all be in same scale and centered around zero
- Express features in terms of standard deviations from the mean

$$x' = \frac{x - \mu}{\sigma} \tag{3}$$

- Images: μ and σ can be computed per:
  - feature $\rightarrow$ “mean image”
  - channel
  - sample
- Standardize validation and test set with training statistics

Batch Normalization

- Standardize input features in each layer by mini-batch statistics
- Enables higher learning rates
- Proposed by Ioffe and Szegedy 2015

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Ensembles

- Boost performance by averaging predictions of several models
- Average predictor's variance (bias-variance trade-off)
- Models should be as diverse as possible, e. g. :
  - Different architectures
  - Different initializations ($\rightarrow$ pretraining)
  - Different optimization
  - Best models of cross validation
  - Models that are not neural networks, e. g. SVMs, Random Forests

Agenda

Hyperparameters and Optimization

Controlling Overfitting

Tips, Tricks and Misc

Summary

Recommendations

- Most important hyperparameters:
  - Initial learning rate
  - Learning rate schedule (step?)
  - Regularization strength (dropout and weight decay)
  - Batch size
- Random search better than grid search
- Overfit subset of data
- SGD + momentum or Adam
- Standardization is crucial!
- One validation fold better than k-fold cross validation for big data
- Get more data
- Get more GPUs

In a Nutshell

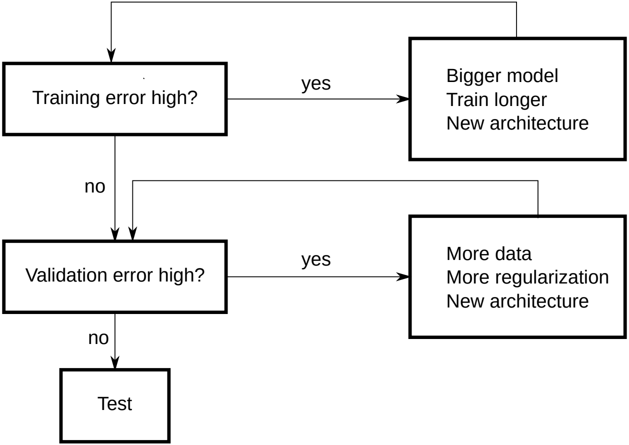

Figure: Adapted from Andrew Ng's NIPS 2016 talk "Nuts and Bolts of Applying DL".

References I

Bengio, Yoshua (2012). “Practical recommendations for gradient-based training of deep architectures”. In: **Neural Networks: Tricks of the Trade**. Springer, pp. 437–478.

Bergstra, James and Yoshua Bengio (2012). “Random search for hyper-parameter optimization”. In: **Journal of Machine Learning Research** 13.Feb, pp. 281–305.

Duchi, John, Elad Hazan, and Yoram Singer (July 2011). “Adaptive subgradient methods for online learning and stochastic optimization”. In: **Journal of Machine Learning Research** 12, pp. 2121–2159.

Erhan, Dumitru et al. (Feb. 2010). “Why does unsupervised pre-training help deep learning?” In: **Journal of Machine Learning Research** 11, pp. 625–660.

Glorot, Xavier and Yoshua Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: **Proceedings of the thirteenth international conference on artificial intelligence and statistics**, pp. 249–256.

References II

Ioffe, Sergey and Christian Szegedy (2015). “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: **arXiv preprint arXiv:1502.03167**.

Karpathy, Andrej (n.d.). **CS231n Convolutional Neural Networks for Visual Recognition**. `http://cs231n.github.io/neural-networks-3/`.

Kingma, Diederik and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: **arXiv preprint arXiv:1412.6980**.

Nesterov, Yurii (1983). “A method of solving a convex programming problem with convergence rate O (1/k2)”. In: **Soviet Mathematics Doklady**. Vol. 27. 2, pp. 372–376.

Pan, Sinno Jialin and Qiang Yang (2010). “A survey on transfer learning”. In: **IEEE Transactions on knowledge and data engineering** 22.10, pp. 1345–1359.

Seegerer, Philipp (Jan. 2017). “Breast Lesion Detection and Segmentation in DBT Images Using Deep Learning Techniques”. MA thesis. FAU Erlangen-Nurnberg.

References III

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting.". In: **Journal of Machine Learning Research** 15.1, pp. 1929–1958.

Tieleman, Tijmen and Geoffrey Hinton (2012). "Divide the gradient by a running average of its recent magnitude". In: **COURSERA: Neural Networks for Machine Learning** 4.2.

Zeiler, Matthew D (2012). "ADADELTA: an adaptive learning rate method". In: **arXiv preprint arXiv:1212.5701**.

Thank you!