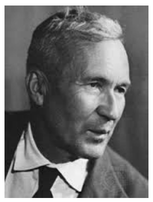
TTIC 31230, Fundamentals of Deep Learning

David McAllester, April 2017

Generalization and Regularization

Chomsky vs. Kolmogorov and Hinton

Noam Chomsky: Natural language grammar cannot be learned by a universal learning algorithm. This position is supported by the "no free lunch theorem".



Andrey Kolmogorov, Geoff Hinton: Universal learning algorithms exist. This position is supported by the "free lunch theorem".

The No Free Lunch Theorem

Without prior knowledge, such as universal grammar, it is impossible to make a prediction for an input you have not seen in the training data.

Proof: Select a predictor h uniformly at random from all functions from $\mathcal{X}$ to $\mathcal{Y}$ and then take the data distribution to draw pairs $(x, h(x))$ where x is drawn uniformly from $\mathcal{X}$. No learning algorithm can predict $h(x)$ where x does not occur in the training data.

The Free Lunch Theorem

Consider a classifier f written in C++ with an arbitrarily large standard library.

Let $|f|$ be the number of bits needed to represent f.

Let $\hat{E}(f)$ be the error rate on an IID training set and let $E(f)$ be the population error rate.

Theorem: With probability at least $1 - \delta$ over the draw of the training data the following holds simultaneously for all f.

$$E(f) \leq \frac{10}{9}\left(\hat{E}(f) + \frac{5}{N}\left((\ln 2)|f| + \ln\frac{1}{\delta}\right)\right)$$

Training Data, Validation Data and Test Data

Good performance on training data does not guarantee good performance on test data.

An nth order polynomial can fit any n (pure noise) data points.

For each order n we can fit a polynomial to the training data and select the order n with the best performance on validation data.

Ultimately performance should be measured on test data not used in any way during learning.

Train Data, Validation Data and Test Data

In general one designs algorithms and tunes hyper-parameters by training on training data and evaluating on validation data.

This is sometimes called graduate student descent.

Kaggle withholds test data until the final contest evaluation.

Loss Vs. Error Rate (or BLEU Score)

While SGD is generally done on cross entropy loss, one often wants minimum classification error or BLEU Score (for translation).

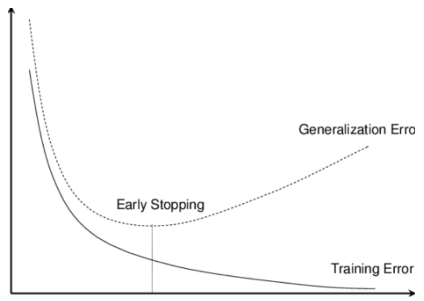
The term “loss” often refers to cross entropy loss as opposed to error rate.

SGD optimizes loss because error is not differentiable.

Later we will discuss attempts to directly optimize error.

But training on loss is generally effective.

Early Stopping



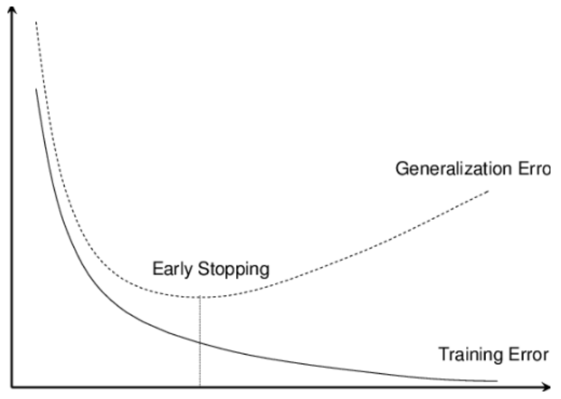
Claudia Perlich

During SGD one tracks validation loss or validation error.

One stops training when the validation error stops improving.

Empirically, loss reaches a minimum sooner than error.

Over Confidence



Validation error is larger than training error when we stop.

The model probabilities are tuned on training data statistics.

The probabilities are tuned to an unrealistically low lower error rate and are therefore over-confident.

This over-confidence occurs before the stopping point and damages validation loss.

Regularization

There is never harm in doing early stopping — one should always do early stopping.

Regularization is a modification to the training algorithm motivated by reducing the training-validation gap and, in this way, improving overall performance.

L_2 Regularization (Weight Decay or Shrinkage)

We will later theoretically motivate the following where N is the number of training examples.

$$\begin{aligned}\Phi^* &= \operatorname*{argmin}_\Phi \left(E_{(x,y)\sim\mathrm{Train}}\ \mathcal{L}(\Phi,x,y)\right) + \frac{\lambda}{2N}||\Phi||^2 \\ &= \operatorname*{argmin}_\Phi\ E_{(x,y)\sim\mathrm{Train}}\ \left(\mathcal{L}(\Phi,x,y) + \frac{\lambda}{2N}||\Phi||^2\right)\end{aligned}$$

This objective corresponds to a Gaussian prior bias

$$p(\Phi) \propto \exp\left(-\frac{1}{2}||\Phi||^2\right)$$

Applying SGD

$$\Phi^* = \operatorname*{argmin}_{\Phi} \; E_{(x,y)\sim\mathrm{Train}} \; \left(\mathcal{L}(\Phi, x, y) + \frac{\lambda}{2N}||\Phi||^2\right)$$

$$\nabla_\Phi \; E_{(x,y)\sim\mathrm{Train}} \; \left(\mathcal{L}(\Phi, x, y) + \frac{\lambda}{2N}||\Phi||^2\right)$$

$$= \; E_{(x,y)\sim\mathrm{Train}} \; \left(g(\Phi, x, y) + \frac{\lambda}{N}\Phi\right)$$

$$\Phi \mathrel{-=} \eta\left(\hat{g} + \frac{\lambda}{N}\Phi\right)$$

Conjugacy Considerations (TZ)

The update

$$\Phi \mathrel{-}= \eta\left(\hat{g} + \frac{\lambda}{N}\Phi\right)$$

is typically parameterized as

$$\Phi \mathrel{-}= \eta\hat{g} + \gamma\Phi$$

where γ is the weight decay parameter.

However, the η, λ, N parameterization seems more conjugate (independently optimizable) than η, γ.

Batch Scaling (TZ)

$$\Phi \mathrel{-}= \eta\left(\hat{g} + \frac{\lambda}{N}\Phi\right)$$

Setting $\eta = B\eta'$ will now handle batch size scaling using the batch size conjugate learning rate η'.

Efficiency and Numerical Stability

$$\Phi \mathrel{-}= \eta \left(\hat{g} + \frac{\lambda}{N} \Phi \right)$$

For N large the shrinkage is very small.

We can improve efficiency and numerical stability by separating shrinkage from SGD and performing

$$\Phi \mathrel{-}= \eta \Phi$$

once for every N/λ steps of SGD.

Shrinkage meets Early Stopping

Early stopping can limit $||\Phi||$.

But early stopping more directly limits $||\Phi - \Phi_{\mathrm{init}}||$.

Theoretical guarantees also work for $||\Phi - \Phi_{\mathrm{init}}||^2$.

$$\begin{aligned}\Phi^* &= \operatorname*{argmin}_{\Phi} \left(E_{(x,y)\sim\mathrm{Train}}\, \mathcal{L}(\Phi, x, y)\right) + \frac{\lambda}{2N}||\Phi - \Phi_{\mathrm{init}}||^2 \\ &= \operatorname*{argmin}_{\Phi}\; E_{(x,y)\sim\mathrm{Train}}\; \left(\mathcal{L}(\Phi, x, y) + \frac{\lambda}{2N}||\Phi - \Phi_{\mathrm{init}}||^2\right)\end{aligned}$$

A Generalization Guarantee Motivating Shrinkage

Assume $0 \leq \mathcal{L}(\Phi, x, y) \leq L_{\max}$.

Define:

$$\mathcal{L}(\Phi) = E_{(x,y)\sim\text{Pop},\ \epsilon\sim\mathcal{N}(0,\sigma)^d}\ \mathcal{L}(\Phi + \epsilon, x, y)$$

$$\hat{\mathcal{L}}(\Phi) = E_{(x,y)\sim\text{Train},\ \epsilon\sim\mathcal{N}(0,\sigma)^d}\ \mathcal{L}(\Phi + \epsilon, x, y)$$

Theorem: With probability at least $1 - \delta$ over the draw of training data the following holds **simultaneously** for all Φ.

$$\mathcal{L}(\Phi) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(\Phi) + \frac{5L_{\max}}{N}\left(\frac{||\Phi - \Phi_{\text{init}}||^2}{2\sigma^2} + \ln\frac{1}{\delta}\right)\right) \qquad \lambda = \frac{5L_{\max}}{\sigma^2}$$

PAC-Bayesian Guarantees

In the PAC-Bayesian framework we assume a prior distribution (or density) on models.

For any prior (true or not) selected before seeing the data, any model with sufficiently large prior probability is guaranteed to have the generalization loss near the training loss.

For the shrinkage bound the prior is $p(\Phi) \propto \exp\left(\frac{-||\Phi - \Phi_{\mathrm{init}}||^2}{2\sigma^2}\right)$.

$$\mathcal{L}(\Phi) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(\Phi) + \frac{5L_{\mathrm{max}}}{N}\left(\frac{||\Phi - \Phi_{\mathrm{init}}||^2}{2\sigma^2} + \ln\frac{1}{\delta}\right)\right)$$

A Simpler Theorem

Consider any prior probability $P(h)$ over an discrete class $\mathcal{H}$.

Assume $0 \leq \mathcal{L}(h, x, y) \leq L_{\max}$.

Define:

$$\mathcal{L}(h) \; = \; E_{(x,y)\sim\mathrm{Pop}}\, \mathcal{L}(h, x, y)$$

$$\hat{\mathcal{L}}(h) \; = \; E_{(x,y)\sim\mathrm{Train}}\, \mathcal{L}(h, x, y)$$

Theorem: With probability at least $1 - \delta$ over the draw of training data the following holds simultaneously for all h.

$$\mathcal{L}(h) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(h) + \frac{5L_{\max}}{N}\left(\ln\frac{1}{P(h)} + \ln\frac{1}{\delta}\right)\right)$$

Proof

Consider $L_{\max} = 1$ and define $\epsilon(h)$ by

$$\epsilon(h) = \sqrt{\frac{2\mathcal{L}(h)\left(\ln \frac{1}{P(h)} + \ln \frac{1}{\delta}\right)}{N}}.$$

By the relative Chernov bound we have

$$P_{\mathrm{Train}\sim\mathrm{Pop}}\left(\hat{\mathcal{L}}(h) \leq \mathcal{L}(h) - \epsilon(h)\right) \leq e^{-N\frac{\epsilon(h)^2}{2\mathcal{L}(h)}} = \delta P(h).$$

Proof

$$P_{\mathrm{Train}\sim\mathrm{Pop}}\left(\hat{\mathcal{L}}(h) \leq \mathcal{L}(h) - \epsilon(h)\right) \leq \delta P(h).$$

$$P_{\mathrm{Train}\sim\mathrm{Pop}}\left(\exists h\ \hat{\mathcal{L}}(h) \leq \mathcal{L}(h) - \epsilon(h)\right) \leq \sum_h \delta P(h) = \delta$$

$$P_{\mathrm{Train}\sim\mathrm{Pop}}\left(\forall h\ \mathcal{L}(h) \leq \hat{\mathcal{L}}(h) + \epsilon(h)\right) \geq 1 - \delta$$

Proof

$$\mathcal{L}(h) \leq \widehat{\mathcal{L}}(h) + \sqrt{\mathcal{L}(h)\left(\frac{2\left(\ln\frac{1}{P(h)} + \ln\frac{1}{\delta}\right)}{N}\right)}$$

using

$$\sqrt{ab} = \inf_{\lambda>0}\ \frac{a}{2\lambda} + \frac{\lambda b}{2}$$

we get

$$\mathcal{L}(h) \leq \widehat{\mathcal{L}}(h) + \frac{\mathcal{L}(h)}{2\lambda} + \frac{\lambda\left(\ln\frac{1}{P(h)} + \ln\frac{1}{\delta}\right)}{N}$$

Proof

$$\mathcal{L}(h) \leq \widehat{\mathcal{L}}(h) + \frac{\mathcal{L}(h)}{2\lambda} + \frac{\lambda\left(\ln\frac{1}{P(h)} + \ln\frac{1}{\delta}\right)}{N}$$

Solving for $\mathcal{L}(h)$ yields

$$\mathcal{L}(h) \leq \frac{1}{1-\frac{1}{2\lambda}}\left(\hat{\mathcal{L}}(h) + \frac{\lambda}{N}\left(\ln\frac{1}{P(h)} + \ln\frac{1}{\delta}\right)\right)$$

Setting $\lambda = 5$ and rescaling the loss gives the version on earlier slides.

A Model Compression Guarantee

Let $|\Phi|$ be the number of bits used to represent Φ under some fixed compression scheme.

Let $P(\Phi) = 2^{-|\Phi|}$

$$\mathcal{L}(\Phi) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(\Phi) + \frac{5L_{\max}}{N}\left((\ln 2)|\Phi| + \ln\frac{1}{\delta}\right)\right)$$

Adding Noise Simulates Limiting Precision

Assume $0 \leq \mathcal{L}(\Phi, x, y) \leq L_{\mathrm{max}}$.

Define:

$$\mathcal{L}(\Phi) = E_{(x,y)\sim\mathrm{Pop},\ \epsilon\sim\mathcal{N}(0,\sigma)^d}\ \mathcal{L}(\Phi+\epsilon, x, y)$$

$$\hat{\mathcal{L}}(\Phi) = E_{(x,y)\sim\mathrm{Train},\ \epsilon\sim\mathcal{N}(0,\sigma)^d}\ \mathcal{L}(\Phi+\epsilon, x, y)$$

Theorem: With probability at least $1-\delta$ over the draw of training data the following holds **simultaneously** for all Φ.

$$\mathcal{L}(\Phi) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(\Phi) + \frac{5L_{\mathrm{max}}}{N}\left(\frac{||\Phi-\Phi_{\mathrm{init}}||^2}{2\sigma^2} + \ln\frac{1}{\delta}\right)\right)$$

L_1 Regularization and Sparse Weights

$$p(\Phi) \propto e^{-||\Phi||_1} \qquad ||\Phi||_1 = \sum_i |\Phi_i|$$

$$\Phi^* = \operatorname*{argmin}_{\Phi} \;\; \hat{\mathcal{L}}(\Phi) \; + \; \lambda ||\Phi||_1$$

$$\begin{aligned} \Phi \;\; &\texttt{-=} \;\; \eta \nabla_{\Phi} \, \mathcal{L}_{\mathrm{train}}(\Phi) \\ \Phi_i \;\; &\texttt{-=} \;\; \eta\lambda \, \mathrm{sign}(\Phi_i) \qquad \text{(shrinkage)} \end{aligned}$$

At equilibrium (sparsity is difficult to achieve with SGD)

$$\begin{aligned} \Phi_i &= 0 & & \text{if } |\partial\mathcal{L}/\partial\Phi_i| < \lambda \\ \partial\mathcal{L}/\partial\Phi_i &= -\lambda\mathrm{sign}(\Phi_i) & & \text{otherwise} \end{aligned}$$

Ensembles

Train several models $\mathrm{Ens} = (\Phi_1,\ \ldots,\ \Phi_k)$ from different initializations and/or under different meta parameters.

We define the ensemble model by

$$P_{\mathrm{Ens}}(y|x) = \frac{1}{k}\sum_{j=1}^{k}\ P_{\Phi_i}(y|x)$$

Ensemble models almost always perform better than any single model.

We will explore some reasons for this.

Ensembles Under Cross Entropy Loss

For log loss we average the probability vectors.

$$P(y|x) = \frac{1}{k}\sum_i \; P_i(y|x)$$

$-\log P$ is a convex function of P. For any convex $\mathcal{L}(P)$ Jensen's inequality states that

$$\mathcal{L}\left(\frac{1}{k}\sum_i P_i\right) \leq \frac{1}{k}\sum_i \mathcal{L}(P_i)$$

This implies that the loss of the average model cannot be worse (can only be better) than the average loss of the models.

Implicit Regularization

Any stochastic learning algorithm, such as SGD, determines a stochastic mapping from training data to models.

The algorithm can implicitly incorporate a preference or bias for models.

For example, solving linear least squares regression with SGD maintains the invariant that Φ is a linear combination of training vectors.

It is not hard to show that SGD finds the zero training error solution minimizing $||\Phi||$.

So solving least squares regression by SGD has an implicit weight norm regularization.

An Implicit Regularization Generalization Guarantee

Let $\mathcal{H}$ be a discrete set of classifiers.

Let A be an algorithm mapping a training set to a classifier.

Let $P(h|A,\mathrm{Pop})$ be the probability over the draw of the training data that $A(\mathrm{Train}) = h$.

Theorem: With probability at least $1-\delta$ over the draw of the training data we have

$$\mathrm{Err}(A(\mathrm{Train})) \leq \frac{10}{9}\left(\begin{array}{l}\hat{\mathrm{Err}}(A(\mathrm{Train})) \\ +\frac{5}{N}\left(\ln\frac{1}{P(A(\mathrm{Train})|A,\mathrm{Pop})} + \ln\frac{1}{\delta}\right)\end{array}\right)$$

Dropout

Dropout can be viewed as an ensemble method.

To draw a model from the ensemble we randomly select a mask μ with

$$\begin{cases} \mu_i = 0 \text{ with probability } \alpha \\ \\ \mu_i = 1 \text{ with probability } 1 - \alpha \end{cases}$$

Then we use the model $(\Phi,\ \mu)$ with weight layers defined by

$$y_i = \mathrm{Relu}\left(\sum_j W_{i,j}\ \mu_j x_j\right)$$

Dropout Training

Repeat:

- Select a random dropout mask μ
- Φ `-=` $\nabla_\Phi \mathcal{L}(\Phi, \mu)$

Backpropagation must use the same mask μ used in the forward computation.

Test Time Scaling

At train time we have

$$y_i = \mathrm{Relu}\left(\sum_j W_{i,j}\ \mu_j x_j\right)$$

At test time we have

$$y_i = \mathrm{Relu}\left((1-\alpha)\sum_j W_{i,j}\ x_j\right)$$

At test time we use the "average network".

Dropout for Least Squares Regression

Consider simple least square regression

$$\begin{aligned}
\Phi^* &= \operatorname*{argmin}_{\Phi} \ \ \mathrm{E}_{(x,y)}\, E_\mu \left(y - \Phi\cdot(\mu \odot x)\right)^2 \\
&= \mathrm{E}\left[(\mu \odot x)(\mu \odot x)^\top\right]^{-1} \mathrm{E}\left[y(\mu \odot x)\right] \\
&= \operatorname*{argmin}_{\Phi} \ \ \mathrm{E}_{(x,y)}(y - (1-\alpha)\Phi\cdot x)^2 + \sum_i \frac{1}{2}(\alpha - \alpha^2)\mathrm{E}\left[x_i^2\right]\Phi_i^2
\end{aligned}$$

In this case dropout is equivalent to a form of L_2 regularization — see Wager et al. (2013).

Model Compression

Deep Compression: Compressing Deep Neural Networks With Pruning, Trained Quantization and Huffman Coding, Han et al., ICLR 2016.

- Compressed Models can be downloaded to mobile devices faster and fit in lower-power CPU memory. (The motivation of this paper).
- Sparse models may be more interpretable than dense models.
- Model size is a measure of model complexity and can be viewed as a form of regularization.

VGG-16 is reduced by $49\times$ from 552MB to 11.3MB with no loss of accuracy.

Three Stages

- Sparsification by simple weight thresholding. ($10\times$ reduction).

- Trained Quantization ($6\times$ reduction).

- Huffman coding (40% reduction).

Quantization

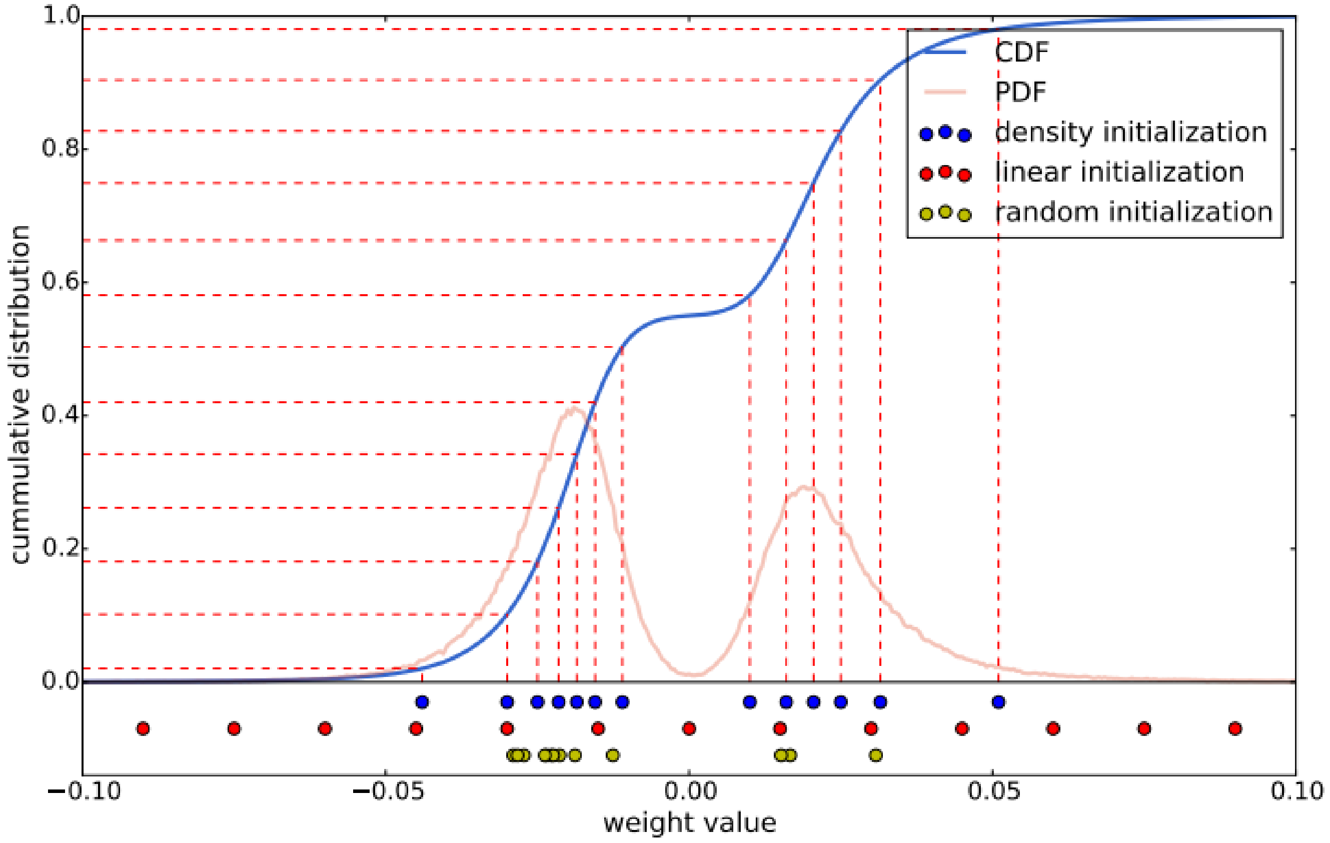
They use 5 bits of numerical precision for the weights.

This is done by having a table of the 32 possible weight values.

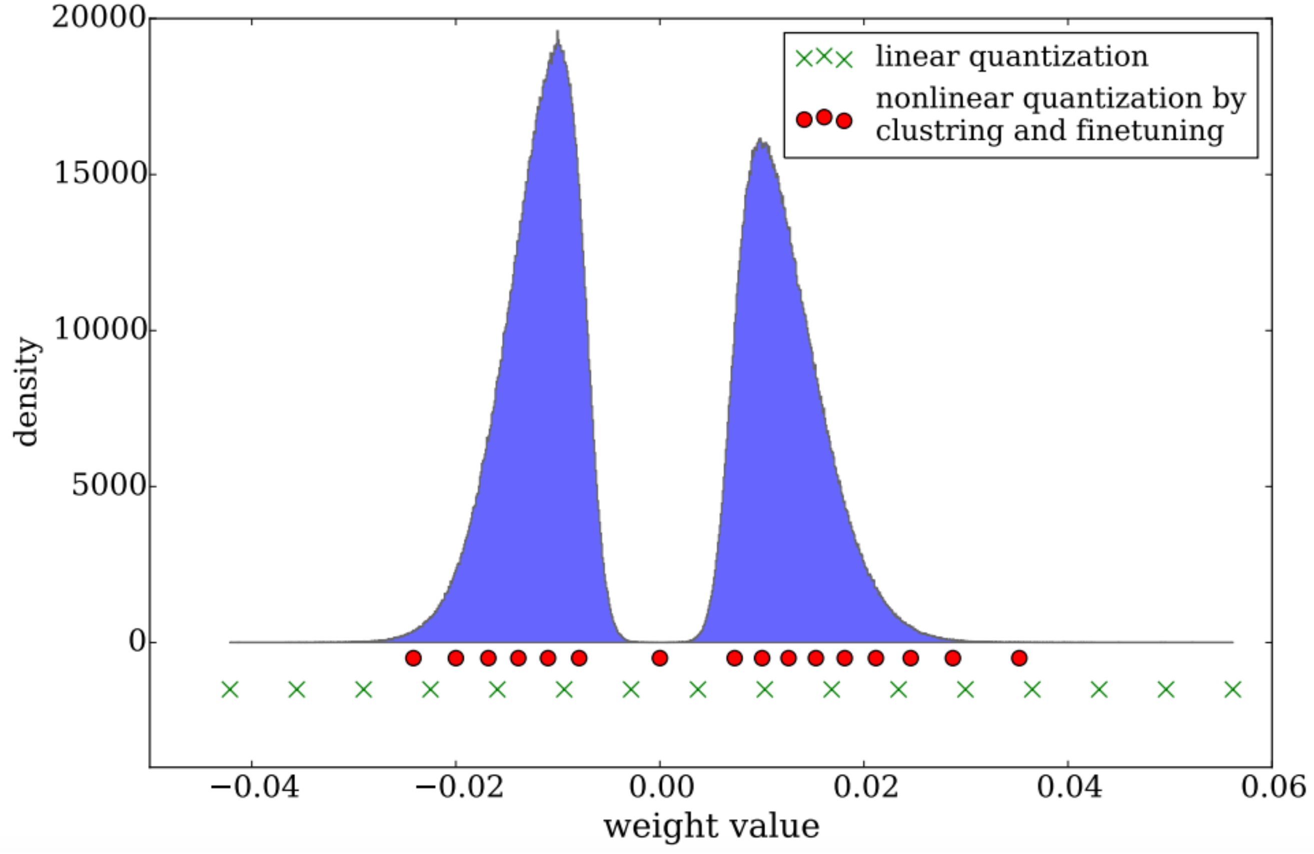
We have to cluster the weights into 32 groups and decide on a **centroid value** for each weight.

This is done with K-means clustering.

Initialization of Centroids



After Running K-means



Retrain to Adjust Centroids

Run over the data again doing backpropagation to adjust the table of the 32 possible weights.

This leaves the 5-bit code of each weight in the model unchanged.

Huffman Coding

Different 5-bit numerical codes have different frequencies.

This can be viewed as distribution over the 32 code words.

We can reduce the average number of bits per weight using fewer bits to code the more common weight values.

Huffman coding is applied to both the 5 bit weight coding and a three bit code used in the sparse representation of the weight matrices.

This results in about 5 bits per **nonzero** weight in a **sparse** coding of the weight matrices.

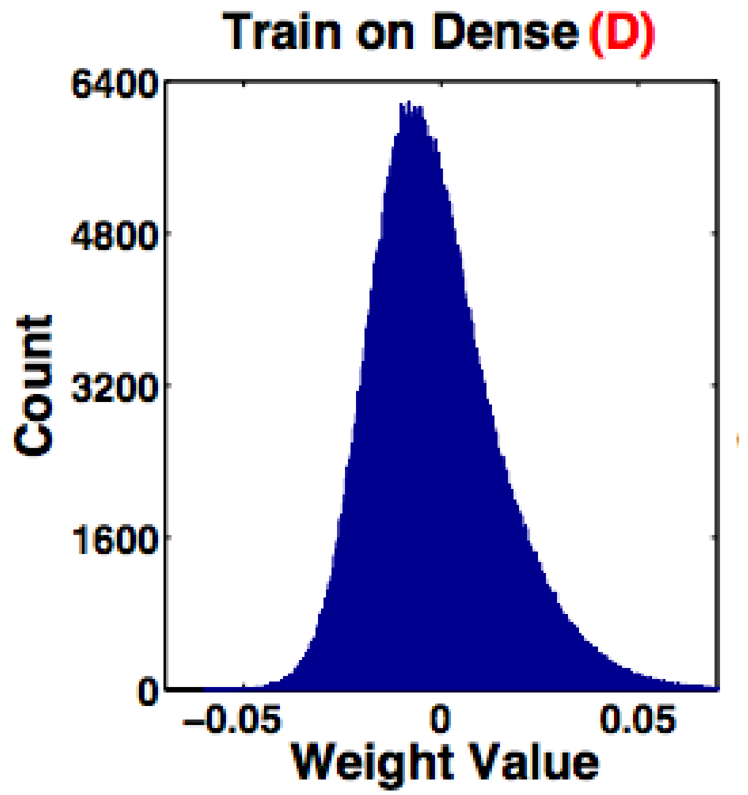
Dense-Sparse-Dense

DSD: Dense-Sparse-Dense Training for Deep Neural Networks, Han et al., ICLR 2017

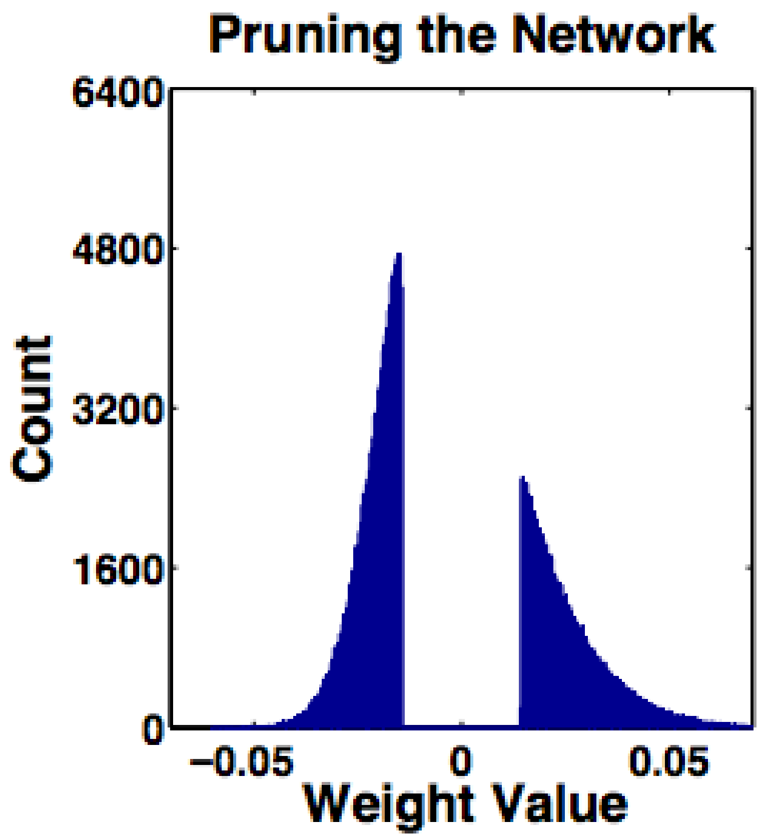
1. Train a model.
2. Make the model sparse by weight thresholding.
3. Retrain the model holding the sparsity pattern fixed (still 32 bits per weight).
4. Go back to a dense model with all pruned weights initialized to zero.
5. Retrain the dense model.

Results in significant performance improvements in a wide variety of models.

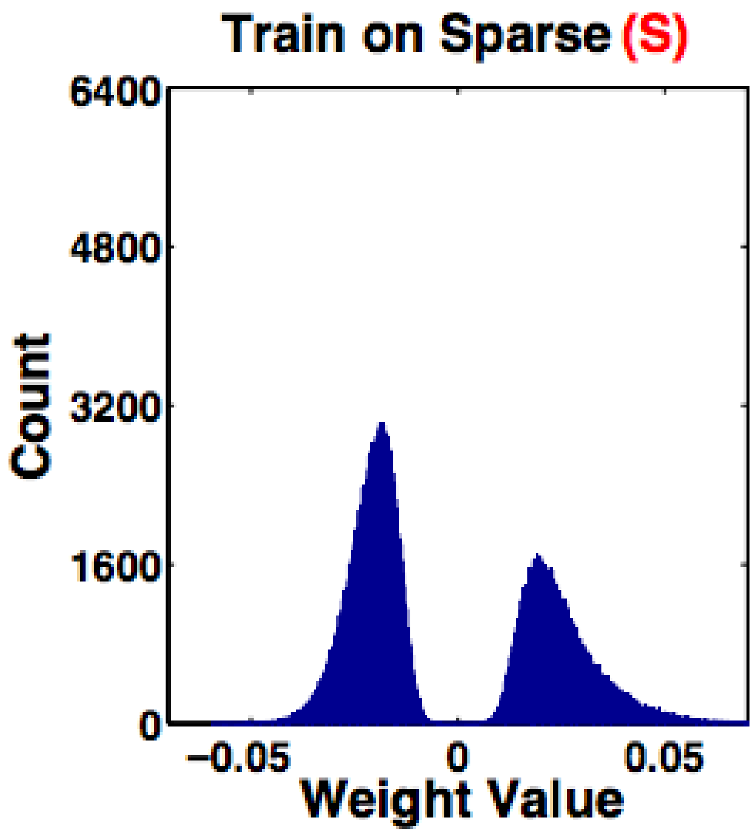
Step 1


Train on Dense (D)
6400
4800
3200
1600
0
Count
−0.05
0
0.05
Weight Value

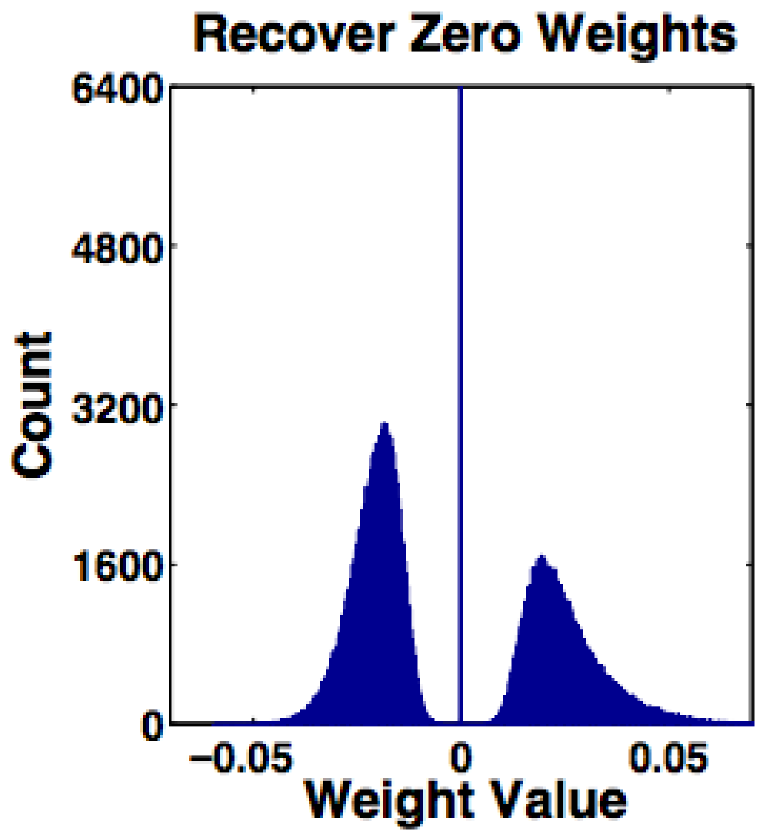
Step 2


Pruning the Network
6400
4800
3200
1600
0
Count
−0.05
0
0.05
Weight Value

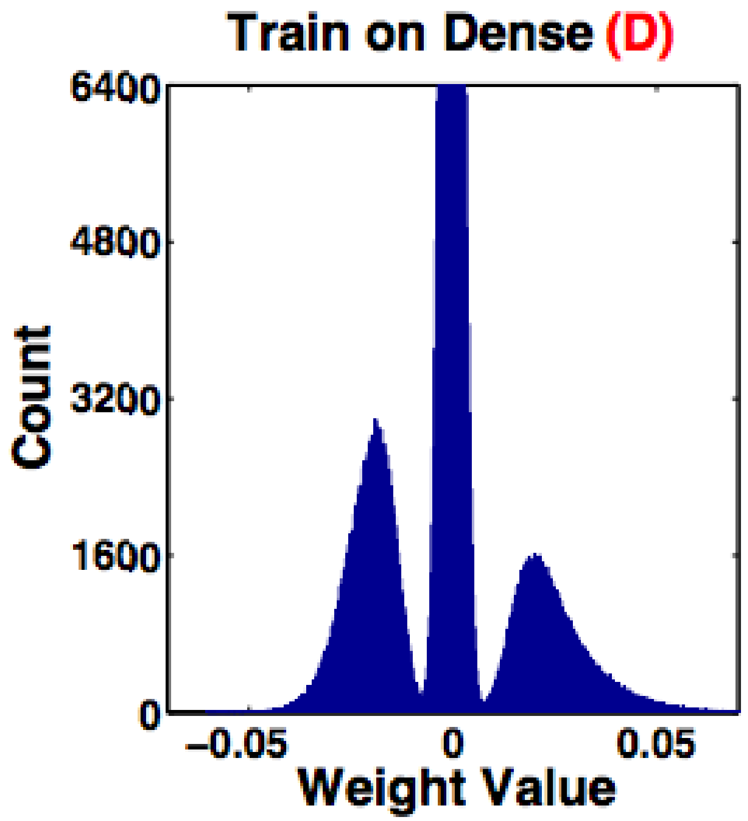
Step 3


Train on Sparse (S)
6400
4800
3200
1600
0
Count
−0.05
0
0.05
Weight Value

Step 4

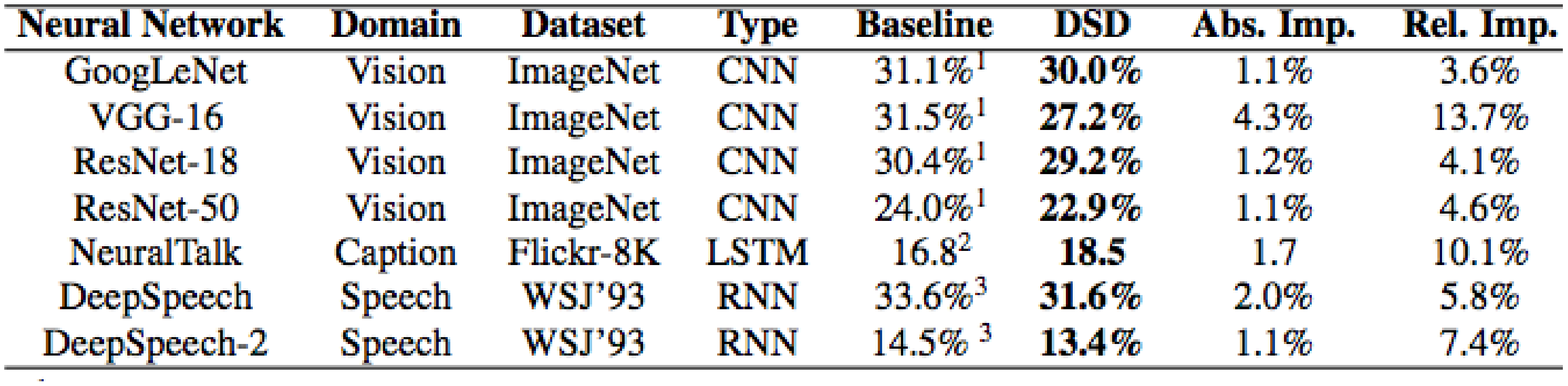

Recover Zero Weights
6400
4800
3200
1600
0
Count
−0.05
0
0.05
Weight Value

Step 5



Train on Dense (D)
Count
6400
4800
3200
1600
0
−0.05
0
0.05
Weight Value

Results



Neural Network	Domain	Dataset	Type	Baseline	DSD	Abs. Imp.	Rel. Imp.
GoogLeNet	Vision	ImageNet	CNN	31.1%[1]	**30.0%**	1.1%	3.6%
VGG-16	Vision	ImageNet	CNN	31.5%[1]	**27.2%**	4.3%	13.7%
ResNet-18	Vision	ImageNet	CNN	30.4%[1]	**29.2%**	1.2%	4.1%
ResNet-50	Vision	ImageNet	CNN	24.0%[1]	**22.9%**	1.1%	4.6%
NeuralTalk	Caption	Flickr-8K	LSTM	16.8[2]	**18.5**	1.7	10.1%
DeepSpeech	Speech	WSJ'93	RNN	33.6%[3]	**31.6%**	2.0%	5.8%
DeepSpeech-2	Speech	WSJ'93	RNN	14.5%[3]	**13.4%**	1.1%	7.4%

Summary

There is never harm in doing early stopping — one should always do early stopping.

Regularization is any modification to the training algorithm motivated by reducing the training-validation gap.

While regularization modifications to training can be inspired by theory, the theory is weak.

Regularization modifications to training should be evaluated empirically.

END