Course Notes: Deep Learning for Visual Computing

Peter Wonka

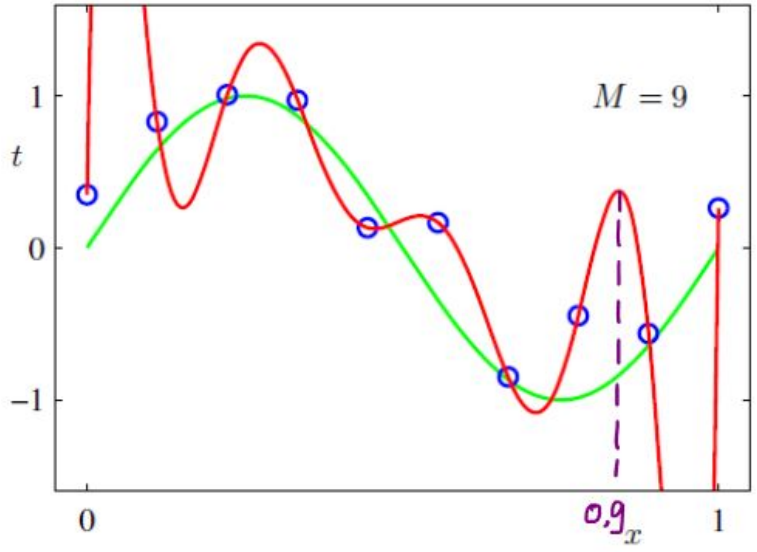
August 30, 2021

Contents

1 Network Regularization

1.1 Introduction

- Review: Story of polynomial fitting


$M = 9$
t
1
0
−1
0
0.9
x
1

- Blue points: training data
- Red Line: without regularization the fitted polynomial passes through all points
  - over swinging, between training samples values can be very small and very large
- Green Line: using L_2 regularization / weight decay

1.2 Design Choices

- Every layer can have a **different regularizer**
- Every layer can have a **different weight** for the regularizer
- Every parameter inside each layer can have a **different weight**, e.g. bias terms
- Regularizer sums up terms for each layer

$$R(\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_k) = \sum_{i=1}^{k} \lambda_i R_i(\boldsymbol{W}_i) \tag{1.1}$$

  - k number of layers, $\boldsymbol{W}_i$ parameter tensor of layer i
- Problems:
  - some layers have parameters that cannot be arranged in a tensor
  - not all parameters participate, different weights per parameters

- **Proposed notation**:

$$R(w_1, w_2, \ldots, w_k) = \sum_{i \in I} \ldots \tag{1.2}$$

  - I index set describing all parameters participating in the regularization
  - w_i individual scalar parameter
  - omit parameters λ_i (can be set per layer or per weight)

1.3 L2 regularization

- add an L_2 penalty to encourage small weights

$$R_{L2} = \sum_{i \in I} w_i^2 \tag{1.3}$$

- Comments
  - L_2 regularization is popular
  - L_2 regularization is also called **weight decay**
  - Check if implementation uses a factor of $\frac{1}{2}\lambda$ or just λ
    - $\frac{1}{2}$ is there to cancel the 2 when computing gradients

1.4 L1 regularization

- add an L_1 penalty to encourage small weights:

$$R_{L1} = \sum_{i \in I} |w_i| \tag{1.4}$$

1.5 Elastic Net regularization

- combine L_1 and L_2 regularization:
  - separately add an L_1 and L_2 regularizer with different weights

$$R_{EL} = \sum_{i \in I} \left(\lambda_1 |w_i| + \lambda_2 w_i^2 \right) \tag{1.5}$$

1.6 What parameters to regularize?

- What weights should be regularized using L_2 or L_1 regularization (What is I)?
  - common: multiplicative weights of linear layers (FC layers)
  - common: multiplicative weights of convolutional layers (CONV layers)
  - maybe / maybe not: bias terms in FC and CONV layers?
  - ???: batch normalization parameters
- How to set weights λ_i?
  - Vary as little as possible

1.7 Max Norm Constraint

- Ensure that the sum of weights for one output / neuron <= threshold $const$:
  - for each output in a linear layer /each conv filter check that:

$$\sum_{i \in NI} \left(w_i^2\right) \leq \textbf{const}^2 \tag{1.6}$$

- NI is the index set describing all the weights for a particular output / neuron
- If the weight exceeds the threshold $const$ project onto the ball with radius $const$

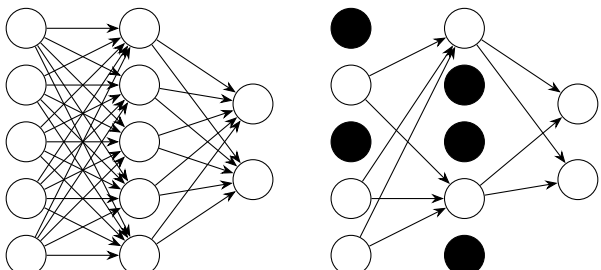
1.8 Dropout

- Literature:
  - original paper: Srivastava, Dropout: A Simple Way to Prevent Neural Networks from Overfitting
  - survey paper: Labach et al., Survey of Dropout Methods for Deep Neural Networks
- Idea:
  - **Parameter**: probability p of keeping a value
  - During **training**:
    - randomly set some values to 0 with probability $1-p$
    - Scale other parameters by $1/p$
  - During **inference**:
    - Eliminate the dropout layer

- Computation:
  - Easiest to imagine as a separate layer that receives an input tensor $\boldsymbol{X}_{in} \in R^{C,W,H}$ and outputs a modified tensor $\boldsymbol{X}_{out}$.
  - For each value in the tensor:

$$\boldsymbol{X}_{out}(c,w,h) = \frac{1}{p} \times [rand < p] \times \boldsymbol{X}_{in}(c,w,h) \tag{1.7}$$

  - $[rand < p]$ is the Iverson bracket



1.8.1 Dropout vs. Inverted Dropout

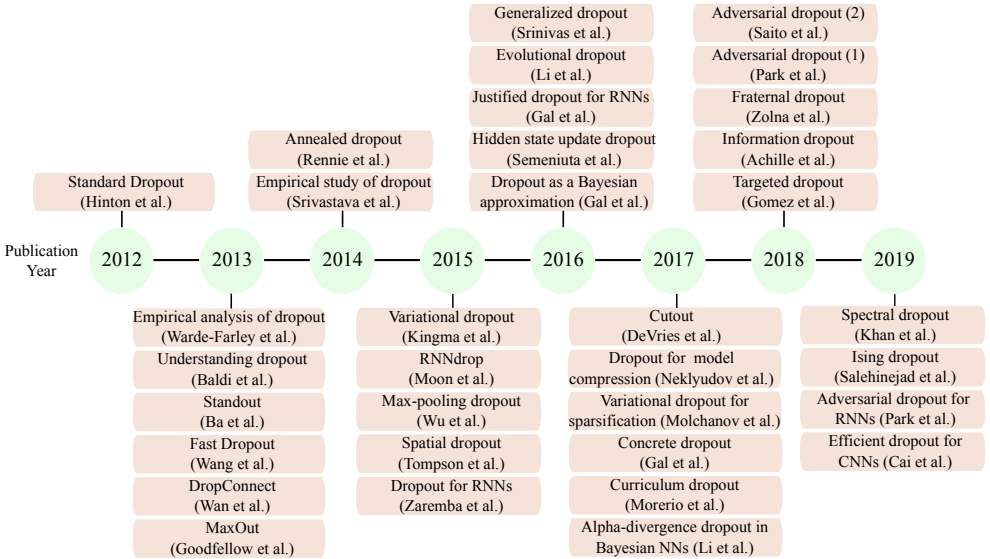
- Dropout changes the expected value of the output
- Two strategies
  - Dropout: scale the weights by p during inference
  - Inverted Dropout: scale the weights by $\frac{1}{p}$ during training after randomly deleting weights with propability p.
- Discussion:
- Inverted Dropout is better, because the code during inference remains the same

1.8.2 Dropout Code Example

```
"""
Inverted Dropout: recommended implementation
Drop and scale at train time and don't do anything at test time
"""

import numpy as np

# probability of keeping a unit active. higher = less dropout
p = 0.5

# simulated input tensor of size 3 x 4
Input = np.random.randn(3,4)

# dropout computation. Notice /p!
Mask = (np.random.rand(*Input.shape) < p) / p
Output = Input * Mask # drop!

print(Input)
```

```
19 print(Mask)
20 print(Output)
```

1.8.3 Survey


Publication Year
2012
2013
2014
2015
2016
2017
2018
2019
Standard Dropout (Hinton et al.)
Annealed dropout (Rennie et al.)
Empirical study of dropout (Srivastava et al.)
Generalized dropout (Srinivas et al.)
Evolutional dropout (Li et al.)
Justified dropout for RNNs (Gal et al.)
Hidden state update dropout (Semeniuta et al.)
Dropout as a Bayesian approximation (Gal et al.)
Adversarial dropout (2) (Saito et al.)
Adversarial dropout (1) (Park et al.)
Fraternal dropout (Zolna et al.)
Information dropout (Achille et al.)
Targeted dropout (Gomez et al.)
Empirical analysis of dropout (Warde-Farley et al.)
Understanding dropout (Baldi et al.)
Standout (Ba et al.)
Fast Dropout (Wang et al.)
DropConnect (Wan et al.)
MaxOut (Goodfellow et al.)
Variational dropout (Kingma et al.)
RNNdrop (Moon et al.)
Max-pooling dropout (Wu et al.)
Spatial dropout (Tompson et al.)
Dropout for RNNs (Zaremba et al.)
Cutout (DeVries et al.)
Dropout for model compression (Neklyudov et al.)
Variational dropout for sparsification (Molchanov et al.)
Concrete dropout (Gal et al.)
Curriculum dropout (Morerio et al.)
Alpha-divergence dropout in Bayesian NNs (Li et al.)
Spectral dropout (Khan et al.)
Ising dropout (Salehinejad et al.)
Adversarial dropout for RNNs (Park et al.)
Efficient dropout for CNNs (Cai et al.)

1.8.4 Dropout Discussion

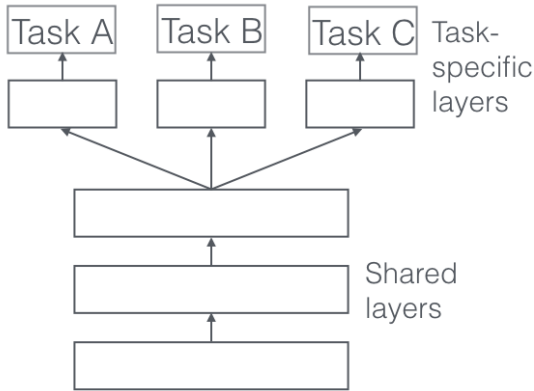
- Dropout can be interpreted as adding noise to the data
  - regularization by making the network robust to noise
- Dropout can be interpreted as simultaneously training an ensemble of networks (disputed)
- Dropout is typically only applied to **certain layers**
  - probability p changes per layer
  - more common in later layers of image classification
- Dropout was very popular before 2015
  - Some networks don't use dropout

1.9 Early Stopping

- When to stop training?
  - One Idea: plot the error on **training set** and **validation set**
  - stop training when error on validation set starts to increase
- Early stopping has a similar effect to L_2 regularization

1.10 Multi-task Learning

- Develop a network to solve **multiple tasks** / problems at once
- Example: predict **segmentation**, **normals**, **depth** for each pixel in an image



1.11 Adding Noise to Network Outputs

- Example: **Label Smoothing**
- Assume a classification problem with one hot encoding of ground truth labels

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{1.8}$$

- Use smoothed labels, ϵ smoothing parameter, k = number of classes - 1

$$\begin{pmatrix} \epsilon/k \\ \epsilon/k \\ 1-\epsilon \\ \epsilon/k \end{pmatrix} \tag{1.9}$$

1.12 Batch-normalization as Regularizer

- Batch normalization can be interpretated as follows
  - We estimate mean μ and standard deviation σ from a mini-batch
  - μ and σ are an estimate of the **true data + noise**
  - Noise: difference between true μ and σ and and mini-batch μ and σ
  - Batch normalization is a regularizer

1.13 DropConnect

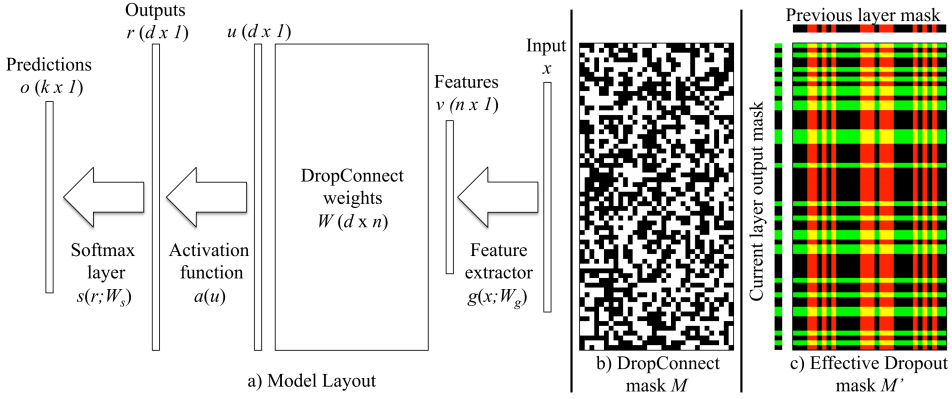
- Dropout: randomly set tensor values to zero
- DropConnect: randomly set network weights to zero
  - Results in a better random pattern
- Dropout example for a linear layer, $\boldsymbol{W}$ parameters, input $\mathbf{x}_{in}$, output $\mathbf{x}_{out}$

$$\mathbf{x}_{out} = \boldsymbol{W}\mathbf{x}_{in} \tag{1.10}$$

$$\begin{pmatrix} 1 & 3 & 4 & 7 \\ 2 & -2 & 5 & 3 \\ 2 & 9 & -1 & -9 \end{pmatrix} \begin{pmatrix} 2 \\ 5 \\ 6 \\ 8 \end{pmatrix} \tag{1.11}$$

$$\begin{pmatrix} 1 & 3 & 4 & 7 \\ 2 & -2 & 5 & 3 \\ 2 & 9 & -1 & -9 \end{pmatrix} \begin{pmatrix} 4 \\ \mathbf{0} \\ 12 \\ \mathbf{0} \end{pmatrix} \tag{1.12}$$

$$\begin{pmatrix} 2 & \mathbf{0} & 8 & \mathbf{0} \\ 4 & \mathbf{0} & 10 & \mathbf{0} \\ 4 & \mathbf{0} & -2 & \mathbf{0} \end{pmatrix} \begin{pmatrix} 2 \\ 5 \\ 6 \\ 8 \end{pmatrix} \tag{1.13}$$

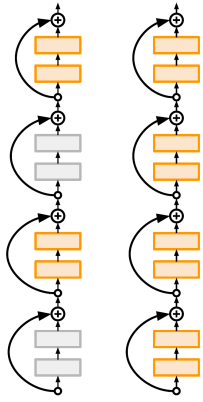


- Figure shows a single DropConnect layer.
- After running feature extractor g() on input x, a random mask M masks out the weight matrix $\boldsymbol{W}$.
- Dropout comparison:

- effective weight mask for dropout applied to the previous layer's output (red columns)
- effective weight mask for dropout applied to this layer's output (green rows)

1.14 Stochastic Depth

- Literature: Huang et al., Deep Networks with Stochastic Depth
- Training: randomly skip layers
- Testing: use all layers



1.15 Data Augmentation

- Discussed Later