

Regularization in Neural Networks

Vineeth N Balasubramanian

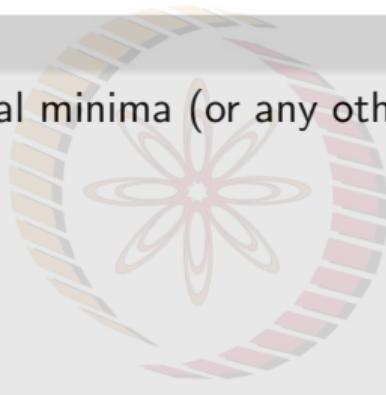
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad



Review

Questions

- How to know if you are in a local minima (or any other critical point on the loss surface)?

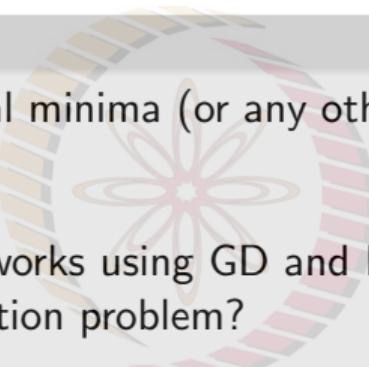


NPTEL

Review

Questions

- How to know if you are in a local minima (or any other critical point on the loss surface)?
We will see this in this lecture
- Why is training deep neural networks using GD and Mean-Squared Error (MSE) as cost function, a non-convex optimization problem?



NPTEL

Review

Questions

- How to know if you are in a local minima (or any other critical point on the loss surface)?
We will see this in this lecture
- Why is training deep neural networks using GD and Mean-Squared Error (MSE) as cost function, a non-convex optimization problem? **Non-linear activation functions can make the problem non-convex**
- Let us assume a linear deep neural network (only linear activation functions, $f(x) = x$). Would using GD and MSE still be a non-convex problem?

Review

Questions

- How to know if you are in a local minima (or any other critical point on the loss surface)?
We will see this in this lecture
- Why is training deep neural networks using GD and Mean-Squared Error (MSE) as cost function, a non-convex optimization problem? **Non-linear activation functions can make the problem non-convex**
- Let us assume a linear deep neural network (only linear activation functions, $f(x) = x$). Would using GD and MSE still be a non-convex problem? **Yes! Weight symmetry is a reason**

What is Regularization? An Intuitive View

What's my rule?

- 1 2 3 \Rightarrow satisfies rule
- 4 5 6 \Rightarrow satisfies rule
- 7 8 9 \Rightarrow satisfies rule
- 9 2 31 \Rightarrow does not satisfy rule



NPTEL

What is Regularization? An Intuitive View

What's my rule?

- 1 2 3 \Rightarrow satisfies rule
- 4 5 6 \Rightarrow satisfies rule
- 7 8 9 \Rightarrow satisfies rule
- 9 2 31 \Rightarrow does not satisfy rule

Plausible rules

- 3 consecutive single digits
- 3 consecutive integers
- 3 numbers in ascending order
- 3 numbers whose sum is less than 25
- 3 numbers < 10
- 1, 4, or 7 in first column
- “yes” to first 3 sequences, “no” to all others

What is Regularization? An Intuitive View

What's my rule?

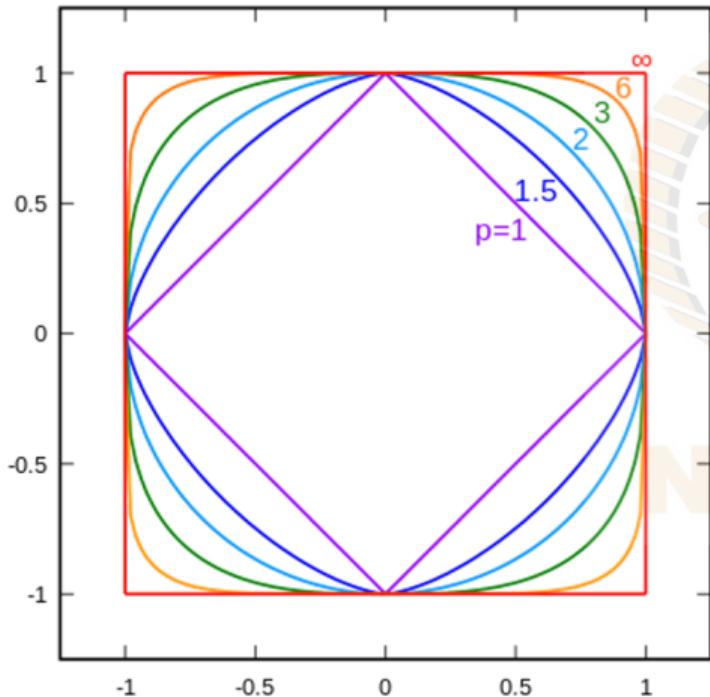
- 1 2 3 \Rightarrow satisfies rule
- 4 5 6 \Rightarrow satisfies rule
- 7 8 9 \Rightarrow satisfies rule
- 9 2 31 \Rightarrow does not satisfy rule

Plausible rules

- 3 consecutive single digits
- 3 consecutive integers
- 3 numbers in ascending order
- 3 numbers whose sum is less than 25
- 3 numbers < 10
- 1, 4, or 7 in first column
- “yes” to first 3 sequences, “no” to all others

Regularization corresponds to methods used in machine learning to improve **generalization performance** (avoid overfitting to training data)

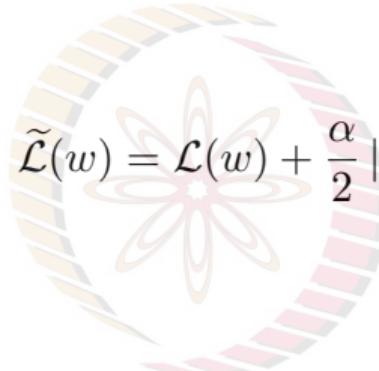
Review: L_p Norms



- All p -norms penalize larger weights
- $p < 2$ tends to create sparse (i.e. lots of 0 weights)
- $p > 2$ tends to like similar weights

L2 Regularization

- Penalty on L2-norm of parameters commonly known as **L_2 weight decay**, or simply **weight decay**
- Loss function defined by:

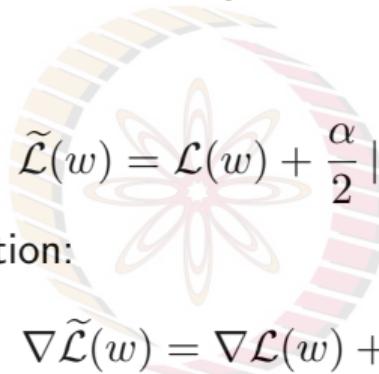
$$\tilde{\mathcal{L}}(w) = \mathcal{L}(w) + \frac{\alpha}{2} \|w\|^2$$


The NPTEL logo consists of a circular wreath made of alternating light blue and light orange segments. In the center is a stylized, symmetrical flower with eight petals, rendered in a light pink or peach color.

Credit: Ali Ghodsi, University of Waterloo; Mitesh Khapra, IIT Madras

L2 Regularization

- Penalty on L2-norm of parameters commonly known as **L_2 weight decay**, or simply **weight decay**
- Loss function defined by:


$$\tilde{\mathcal{L}}(w) = \mathcal{L}(w) + \frac{\alpha}{2} \|w\|^2$$
$$\nabla \tilde{\mathcal{L}}(w) = \nabla \mathcal{L}(w) + \alpha w$$

NPTEL

Credit: Ali Ghodsi, University of Waterloo; Mitesh Khapra, IIT Madras

L2 Regularization

- Penalty on L2-norm of parameters commonly known as **L_2 weight decay**, or simply **weight decay**
- Loss function defined by:

$$\tilde{\mathcal{L}}(w) = \mathcal{L}(w) + \frac{\alpha}{2} \|w\|^2$$

- Gradient of total objective function:

$$\nabla \tilde{\mathcal{L}}(w) = \nabla \mathcal{L}(w) + \alpha w$$

- Update rule:

$$w_{t+1} = w_t - \eta \nabla \mathcal{L}(w_t) - \eta \alpha w_t$$

Credit: Ali Ghodsi, University of Waterloo; Mitesh Khapra, IIT Madras

L2 Regularization: Analysis

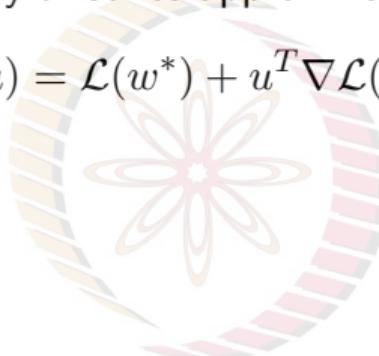
- Let w^* be optimal solution in absence of regularization [i.e. $\nabla \mathcal{L}(w^*) = 0$]



L2 Regularization: Analysis

- Let w^* be optimal solution in absence of regularization [i.e. $\nabla \mathcal{L}(w^*) = 0$]
- Consider $u = w - w^*$. Using Taylor series approximation upto 2nd order:

$$\mathcal{L}(w^* + u) = \mathcal{L}(w^*) + u^T \nabla \mathcal{L}(w^*) + \frac{1}{2} u^T H u$$



NPTEL

L2 Regularization: Analysis

- Let w^* be optimal solution in absence of regularization [i.e. $\nabla \mathcal{L}(w^*) = 0$]
- Consider $u = w - w^*$. Using Taylor series approximation upto 2nd order:

$$\begin{aligned}\mathcal{L}(w^* + u) &= \mathcal{L}(w^*) + u^T \nabla \mathcal{L}(w^*) + \frac{1}{2} u^T H u \\ \mathcal{L}(w) &= \mathcal{L}(w^*) + (w - w^*)^T \nabla \mathcal{L}(w^*) + \frac{1}{2} (w - w^*)^T H (w - w^*) \\ &= \mathcal{L}(w^*) + \frac{1}{2} (w - w^*)^T H (w - w^*) \quad (\because \nabla \mathcal{L}(w^*) = 0)\end{aligned}\tag{1}$$

NPTEL

L2 Regularization: Analysis

- Let w^* be optimal solution in absence of regularization [i.e. $\nabla \mathcal{L}(w^*) = 0$]
- Consider $u = w - w^*$. Using Taylor series approximation upto 2nd order:

$$\begin{aligned}\mathcal{L}(w^* + u) &= \mathcal{L}(w^*) + u^T \nabla \mathcal{L}(w^*) + \frac{1}{2} u^T H u \\ \mathcal{L}(w) &= \mathcal{L}(w^*) + (w - w^*)^T \nabla \mathcal{L}(w^*) + \frac{1}{2} (w - w^*)^T H (w - w^*) \\ &= \mathcal{L}(w^*) + \frac{1}{2} (w - w^*)^T H (w - w^*) \quad (\because \nabla \mathcal{L}(w^*) = 0)\end{aligned}\tag{1}$$

- Taking derivative:

$$\nabla \mathcal{L}(w) = \nabla \mathcal{L}(w^*) + H(w - w^*) = H(w - w^*)$$

L2 Regularization: Analysis

- Let w^* be optimal solution in absence of regularization [i.e. $\nabla \mathcal{L}(w^*) = 0$]
- Consider $u = w - w^*$. Using Taylor series approximation upto 2nd order:

$$\begin{aligned}\mathcal{L}(w^* + u) &= \mathcal{L}(w^*) + u^T \nabla \mathcal{L}(w^*) + \frac{1}{2} u^T H u \\ \mathcal{L}(w) &= \mathcal{L}(w^*) + (w - w^*)^T \nabla \mathcal{L}(w^*) + \frac{1}{2} (w - w^*)^T H (w - w^*) \\ &= \mathcal{L}(w^*) + \frac{1}{2} (w - w^*)^T H (w - w^*) \quad (\because \nabla \mathcal{L}(w^*) = 0)\end{aligned}\tag{1}$$

- Taking derivative:

$$\nabla \mathcal{L}(w) = \nabla \mathcal{L}(w^*) + H(w - w^*) = H(w - w^*)$$

- Gradient of total objective function (in presence of L2 regularization):

$$\nabla \tilde{\mathcal{L}}(w) = \nabla \mathcal{L}(w) + \alpha w = H(w - w^*) + \alpha w$$

Credit: Ali Ghodsi, Univ of Waterloo; Mitesh Khapra, IIT Madras

L2 Regularization: Analysis

- Let \tilde{w} be optimal solution in presence of regularization, i.e. $\nabla \tilde{\mathcal{L}}(\tilde{w}) = 0$



NPTEL

L2 Regularization: Analysis

- Let \tilde{w} be optimal solution in presence of regularization, i.e. $\nabla \tilde{\mathcal{L}}(\tilde{w}) = 0$

$$H(\tilde{w} - w^*) + \alpha \tilde{w} = 0$$



NPTEL

L2 Regularization: Analysis

- Let \tilde{w} be optimal solution in presence of regularization, i.e. $\nabla \tilde{\mathcal{L}}(\tilde{w}) = 0$

$$H(\tilde{w} - w^*) + \alpha \tilde{w} = 0$$

$$\therefore (H + \alpha \mathbb{I})\tilde{w} = Hw^*$$



L2 Regularization: Analysis

- Let \tilde{w} be optimal solution in presence of regularization, i.e. $\nabla \tilde{\mathcal{L}}(\tilde{w}) = 0$

$$H(\tilde{w} - w^*) + \alpha \tilde{w} = 0$$

$$\therefore (H + \alpha \mathbb{I})\tilde{w} = Hw^*$$

$$\implies \tilde{w} = (H + \alpha \mathbb{I})^{-1} Hw^*$$



L2 Regularization: Analysis



- Let \tilde{w} be optimal solution in presence of regularization, i.e. $\nabla \tilde{\mathcal{L}}(\tilde{w}) = 0$

$$H(\tilde{w} - w^*) + \alpha \tilde{w} = 0$$

$$\therefore (H + \alpha \mathbb{I})\tilde{w} = Hw^*$$

$$\implies \tilde{w} = (H + \alpha \mathbb{I})^{-1} Hw^*$$

- If $\alpha \rightarrow 0$, then $\tilde{w} \rightarrow w^*$; and therefore no regularization

NPTEL

L2 Regularization: Analysis

- What happens when $\alpha \neq 0$?

- Let \tilde{w} be optimal solution in presence of regularization, i.e. $\nabla \tilde{\mathcal{L}}(\tilde{w}) = 0$

$$H(\tilde{w} - w^*) + \alpha \tilde{w} = 0$$

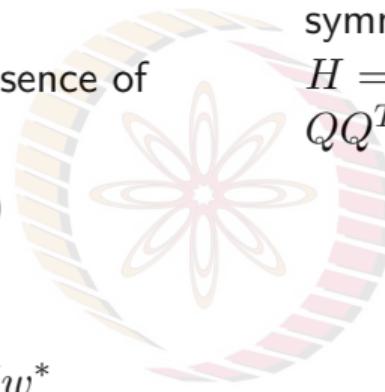
$$\therefore (H + \alpha \mathbb{I})\tilde{w} = Hw^*$$

$$\implies \tilde{w} = (H + \alpha \mathbb{I})^{-1} Hw^*$$



- If $\alpha \rightarrow 0$, then $\tilde{w} \rightarrow w^*$; and therefore no regularization

L2 Regularization: Analysis



- Let \tilde{w} be optimal solution in presence of regularization, i.e. $\nabla \tilde{\mathcal{L}}(\tilde{w}) = 0$

$$H(\tilde{w} - w^*) + \alpha \tilde{w} = 0$$

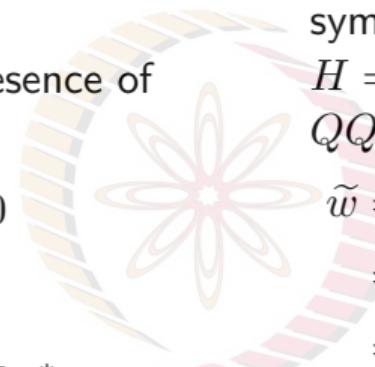
$$\therefore (H + \alpha \mathbb{I})\tilde{w} = Hw^*$$

$$\implies \tilde{w} = (H + \alpha \mathbb{I})^{-1} Hw^*$$

- If $\alpha \rightarrow 0$, then $\tilde{w} \rightarrow w^*$; and therefore no regularization

- What happens when $\alpha \neq 0$? If H is symmetric positive semidefinite, $H = Q\Lambda Q^T$, where Q is orthogonal i.e. $QQ^T = Q^TQ = \mathbb{I}$

L2 Regularization: Analysis



- Let \tilde{w} be optimal solution in presence of regularization, i.e. $\nabla \tilde{\mathcal{L}}(\tilde{w}) = 0$

$$H(\tilde{w} - w^*) + \alpha \tilde{w} = 0$$

$$\therefore (H + \alpha \mathbb{I})\tilde{w} = Hw^*$$

$$\implies \tilde{w} = (H + \alpha \mathbb{I})^{-1} Hw^*$$

- If $\alpha \rightarrow 0$, then $\tilde{w} \rightarrow w^*$; and therefore no regularization

- What happens when $\alpha \neq 0$? If H is symmetric positive semidefinite, $H = Q\Lambda Q^T$, where Q is orthogonal i.e. $QQ^T = Q^TQ = \mathbb{I}$

$$\begin{aligned}\tilde{w} &= (H + \alpha \mathbb{I})^{-1} Hw^* \\ &= (Q\Lambda Q^T + \alpha \mathbb{I})^{-1} Q\Lambda Q^T w^* \\ &= (Q\Lambda Q^T + \alpha Q\mathbb{I}Q^T)^{-1} Q\Lambda Q^T w^* \\ &= [Q(\Lambda + \alpha \mathbb{I})Q^T]^{-1} Q\Lambda Q^T w^* \\ &= Q^{T^{-1}} (\Lambda + \alpha \mathbb{I})^{-1} Q^{-1} Q\Lambda Q^T w^* \\ &= Q(\Lambda + \alpha \mathbb{I})^{-1} \Lambda Q^T w^* (\because Q^{T^{-1}} = Q)\end{aligned}$$

Credit: Ali Ghodsi, Univ of Waterloo; Mitesh Khapra, IIT Madras

L2 Regularization: Analysis

- Each element i of $Q^T w^*$ gets scaled by $\frac{\lambda_i}{\lambda_i + \alpha}$ before it is rotated back by Q



L2 Regularization: Analysis

- Each element i of $Q^T w^*$ gets scaled by $\frac{\lambda_i}{\lambda_i + \alpha}$ before it is rotated back by Q
- If $\lambda_i \gg \alpha$ then $\frac{\lambda_i}{\lambda_i + \alpha} = 1$



L2 Regularization: Analysis

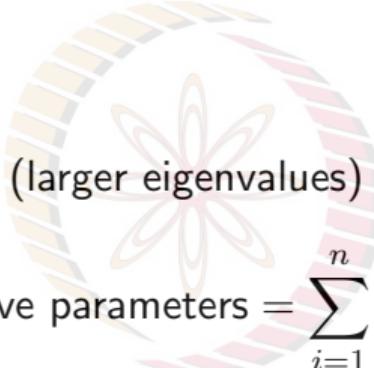
- Each element i of $Q^T w^*$ gets scaled by $\frac{\lambda_i}{\lambda_i + \alpha}$ before it is rotated back by Q
- If $\lambda_i \gg \alpha$ then $\frac{\lambda_i}{\lambda_i + \alpha} = 1$
- If $\lambda_i \ll \alpha$ then $\frac{\lambda_i}{\lambda_i + \alpha} = 0$



NPTEL

L2 Regularization: Analysis

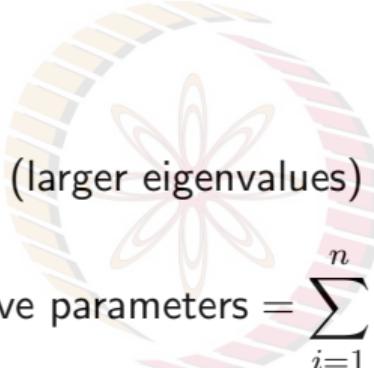
- Each element i of $Q^T w^*$ gets scaled by $\frac{\lambda_i}{\lambda_i + \alpha}$ before it is rotated back by Q
- If $\lambda_i \gg \alpha$ then $\frac{\lambda_i}{\lambda_i + \alpha} = 1$
- If $\lambda_i \ll \alpha$ then $\frac{\lambda_i}{\lambda_i + \alpha} = 0$
- Thus, only significant directions (larger eigenvalues) will be retained.


$$\text{Effective parameters} = \sum_{i=1}^n \frac{\lambda_i}{\lambda_i + \alpha} < n$$

NPTEL

L2 Regularization: Analysis

- Each element i of $Q^T w^*$ gets scaled by $\frac{\lambda_i}{\lambda_i + \alpha}$ before it is rotated back by Q
- If $\lambda_i \gg \alpha$ then $\frac{\lambda_i}{\lambda_i + \alpha} = 1$
- If $\lambda_i \ll \alpha$ then $\frac{\lambda_i}{\lambda_i + \alpha} = 0$
- Thus, only significant directions (larger eigenvalues) will be retained.


$$\text{Effective parameters} = \sum_{i=1}^n \frac{\lambda_i}{\lambda_i + \alpha} < n$$

- **Summary:** The weight vector (w^*) is getting rotated to \tilde{w} on using L2 regularization. All of its elements are shrinking but some are shrinking more than the others. This ensures that only important features are given high weights.

Credit: Ali Ghodsi, Univ of Waterloo; Mitesh Khapra, IIT Madras

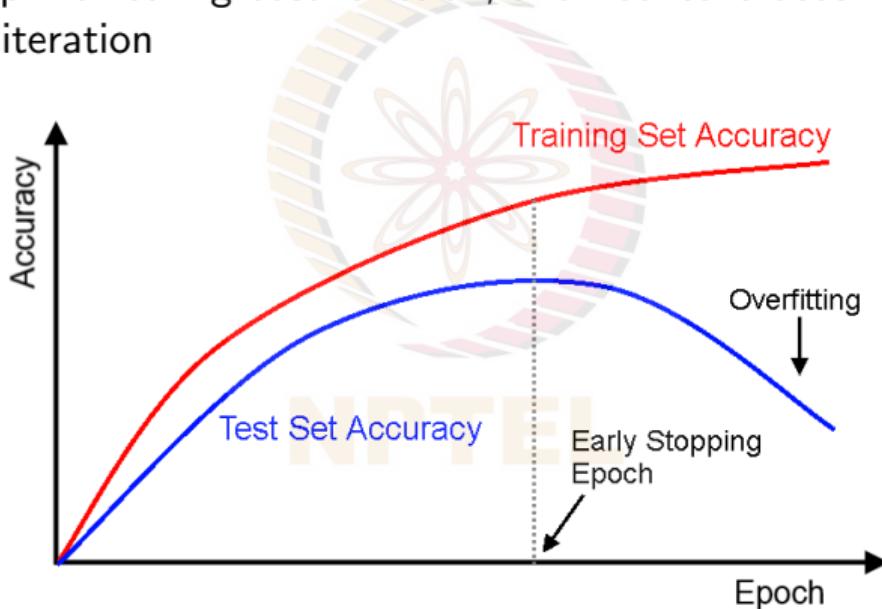
Early Stopping

- **Simple idea:** keep monitoring cost function, and not let it become too low consistently; stop at an earlier iteration



Early Stopping

- **Simple idea:** keep monitoring cost function, and not let it become too low consistently; stop at an earlier iteration



Early Stopping

When to stop?

- Train n epochs; lower learning rate; train m epochs



Early Stopping

When to stop?

- Train n epochs; lower learning rate; train m epochs → **Bad idea**: can't assume one-size-fits-all approach

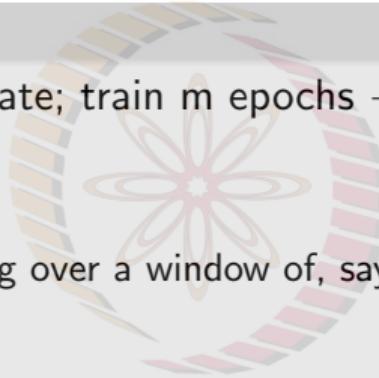


NPTEL

Early Stopping

When to stop?

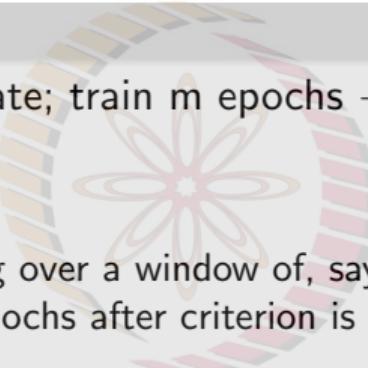
- Train n epochs; lower learning rate; train m epochs → **Bad idea**: can't assume one-size-fits-all approach
- **Error-change criterion:**
 - Stop when error isn't dropping over a window of, say, 10 epochs



Early Stopping

When to stop?

- Train n epochs; lower learning rate; train m epochs → **Bad idea**: can't assume one-size-fits-all approach
- **Error-change criterion:**
 - Stop when error isn't dropping over a window of, say, 10 epochs
 - Train for a fixed number of epochs after criterion is reached (possibly with lower learning rate)



NPTEL

Early Stopping

When to stop?

- Train n epochs; lower learning rate; train m epochs → **Bad idea**: can't assume one-size-fits-all approach
- **Error-change criterion:**
 - Stop when error isn't dropping over a window of, say, 10 epochs
 - Train for a fixed number of epochs after criterion is reached (possibly with lower learning rate)
- **Weight-change criterion:**
 - Compare weights at epochs $t - 10$ and t and test: $\max_i \|w_i^t - w_i^{t-10}\| < \rho$

Early Stopping

When to stop?

- Train n epochs; lower learning rate; train m epochs → **Bad idea**: can't assume one-size-fits-all approach
- **Error-change criterion:**
 - Stop when error isn't dropping over a window of, say, 10 epochs
 - Train for a fixed number of epochs after criterion is reached (possibly with lower learning rate)
- **Weight-change criterion:**
 - Compare weights at epochs $t - 10$ and t and test: $\max_i \|w_i^t - w_i^{t-10}\| < \rho$
 - Don't base on length of overall weight change vector

Early Stopping

When to stop?

- Train n epochs; lower learning rate; train m epochs → **Bad idea**: can't assume one-size-fits-all approach
- **Error-change criterion:**
 - Stop when error isn't dropping over a window of, say, 10 epochs
 - Train for a fixed number of epochs after criterion is reached (possibly with lower learning rate)
- **Weight-change criterion:**
 - Compare weights at epochs $t - 10$ and t and test: $\max_i \|w_i^t - w_i^{t-10}\| < \rho$
 - Don't base on length of overall weight change vector
 - Possibly express as a percentage of the weight

Dataset Augmentation

- Creation of data using some knowledge of task



Dataset Augmentation

- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image



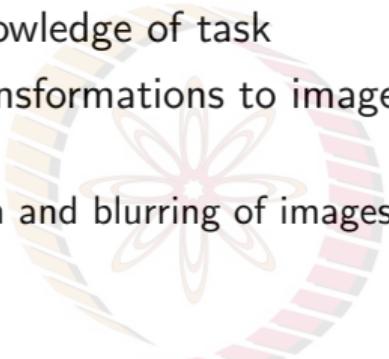
Dataset Augmentation

- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image
- Methods:



Dataset Augmentation

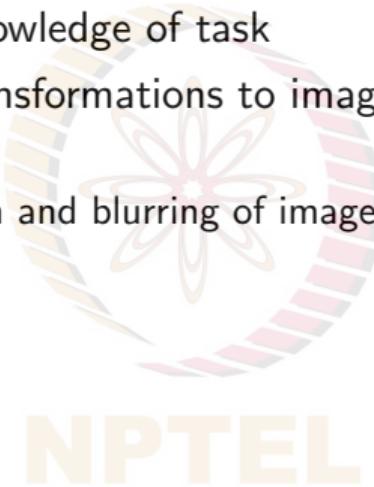
- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image
- Methods:
 - Data jittering (E.g. Distortion and blurring of images)



NPTEL

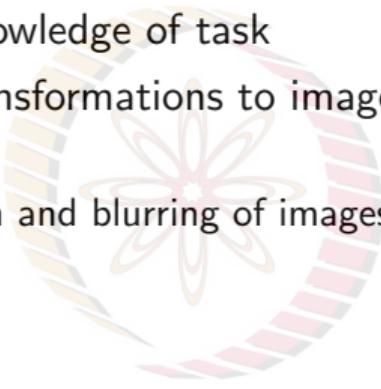
Dataset Augmentation

- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image
- Methods:
 - Data jittering (E.g. Distortion and blurring of images)
 - Rotations



Dataset Augmentation

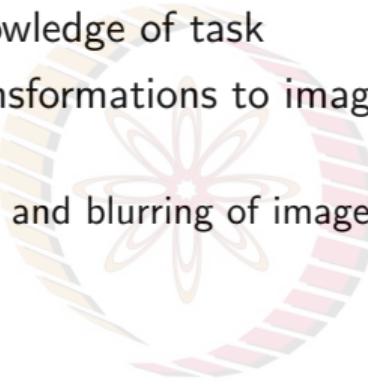
- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image
- Methods:
 - Data jittering (E.g. Distortion and blurring of images)
 - Rotations
 - Color changes



NPTEL

Dataset Augmentation

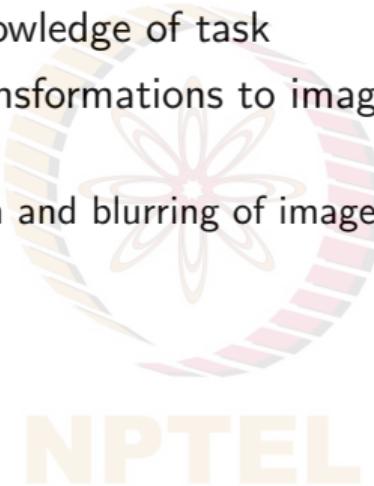
- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image
- Methods:
 - Data jittering (E.g. Distortion and blurring of images)
 - Rotations
 - Color changes
 - Noise injection



NPTEL

Dataset Augmentation

- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image
- Methods:
 - Data jittering (E.g. Distortion and blurring of images)
 - Rotations
 - Color changes
 - Noise injection
 - Mirroring



Dataset Augmentation

- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image
- Methods:
 - Data jittering (E.g. Distortion and blurring of images)
 - Rotations
 - Color changes
 - Noise injection
 - Mirroring
- Helps increase data; is useful when training data provided is less (DNNs need large amounts of training data to work!)

Dataset Augmentation

- Creation of data using some knowledge of task
- Exploit the fact that certain transformations to image do not change label of image
- Methods:
 - Data jittering (E.g. Distortion and blurring of images)
 - Rotations
 - Color changes
 - Noise injection
 - Mirroring
- Helps increase data; is useful when training data provided is less (DNNs need large amounts of training data to work!)
- Also acts as regularizer (by avoiding overfitting to provided data)

Dataset Augmentation: Example¹



¹Wu, Ren, et al. "Deep image: Scaling up image recognition." arXiv 2015

Dataset Augmentation: Example¹



¹Wu, Ren, et al. "Deep image: Scaling up image recognition." arXiv 2015

Data Augmentation: Newer Methods

Mixup

- Create virtual training examples as below ($\lambda \in [0, 1]$):

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j$$

$$\tilde{y} = \lambda y_i + (1 - \lambda)y_j$$

where x_i, x_j are input vectors, and y_i, y_j are one-hot label encodings

- Variants:* Manifold Mixup, AugMix

CutOut

- Randomly mask out square regions of input during training



- Variants:* CutMix

Noise Injection

Injecting noise is another form of regularization; has shown impressive results in certain applications:

- **Data Noise**



²Bishop. Training with noise is equivalent to Tikhonov regularization. Neural Computation, 1995

Noise Injection

Injecting noise is another form of regularization; has shown impressive results in certain applications:

- **Data Noise**
 - Add noise to data while training



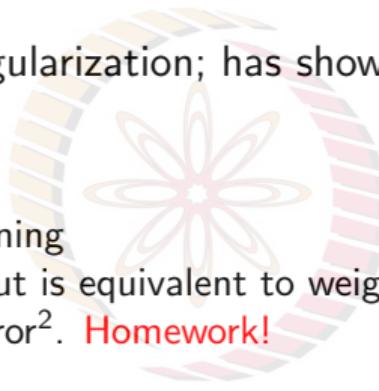
²Bishop. Training with noise is equivalent to Tikhonov regularization. Neural Computation, 1995

Noise Injection

Injecting noise is another form of regularization; has shown impressive results in certain applications:

- **Data Noise**

- Addd noise to data while training
- Adding Gaussian noise to input is equivalent to weight decay (L_2 regularisation) when loss function is sum-of-squared error². **Homework!**



NPTEL

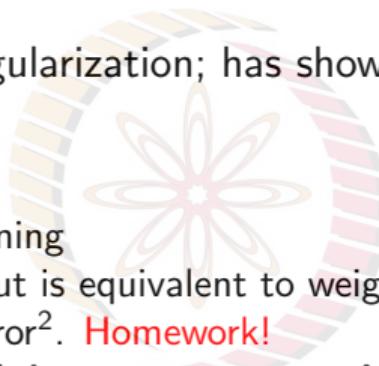
²Bishop. Training with noise is equivalent to Tikhonov regularization. Neural Computation, 1995

Noise Injection

Injecting noise is another form of regularization; has shown impressive results in certain applications:

- **Data Noise**

- Addd noise to data while training
- Adding Gaussian noise to input is equivalent to weight decay (L_2 regularisation) when loss function is sum-of-squared error². **Homework!**
- Can be interpreted as form of **dataset augmentation**.



NPTEL

²Bishop. Training with noise is equivalent to Tikhonov regularization. Neural Computation, 1995

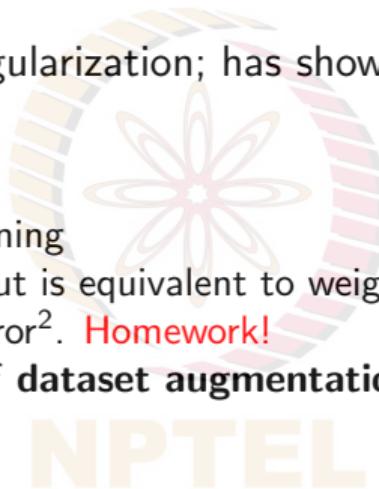
Noise Injection

Injecting noise is another form of regularization; has shown impressive results in certain applications:

- **Data Noise**

- Addd noise to data while training
- Adding Gaussian noise to input is equivalent to weight decay (L_2 regularisation) when loss function is sum-of-squared error². **Homework!**
- Can be interpreted as form of **dataset augmentation**.

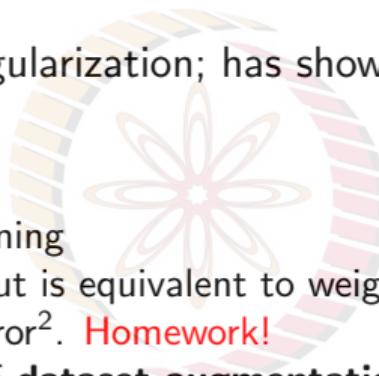
- **Label Noise**



²Bishop. Training with noise is equivalent to Tikhonov regularization. Neural Computation, 1995

Noise Injection

Injecting noise is another form of regularization; has shown impressive results in certain applications:



- **Data Noise**
 - Addd noise to data while training
 - Adding Gaussian noise to input is equivalent to weight decay (L_2 regularisation) when loss function is sum-of-squared error². **Homework!**
 - Can be interpreted as form of **dataset augmentation**.
- **Label Noise**
- **Gradient Noise**

²Bishop. Training with noise is equivalent to Tikhonov regularization. Neural Computation, 1995

Regularization through Label Noise³

- Disturb each training sample with probability α .
- For each disturbed sample, label randomly drawn from uniform distribution over $\{1, 2, \dots, C\}$, regardless of true label

Algorithm 1 DisturbLabel

```
1: Input:  $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ , noise rate  $\alpha$ .  
2: Initialization: a network model  $\mathbb{M}$ :  $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_0) \in \mathbb{R}^C$ ;  
3: for each mini-batch  $\mathcal{D}_t = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^M$  do  
4:   for each sample  $(\mathbf{x}_m, \mathbf{y}_m)$  do  
5:     Generate a disturbed label  $\tilde{\mathbf{y}}_m$  with Eqn (2);  
6:   end for  
7:   Update the parameters  $\boldsymbol{\theta}_t$  with Eqn (1);  
8: end for  
9: Output: the trained model  $\mathbb{M}'$ :  $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_T) \in \mathbb{R}^C$ .
```

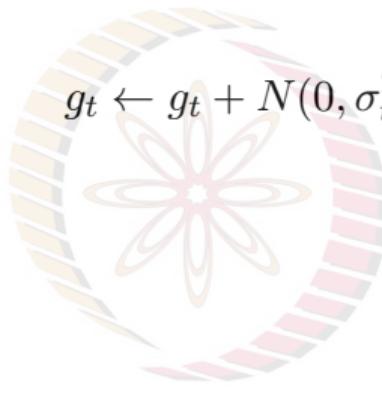
$$\begin{cases} \tilde{c} & \sim \mathcal{P}(\alpha), \\ \tilde{y}_{\tilde{c}} & = 1, \\ \tilde{y}_i & = 0, \quad \forall i \neq \tilde{c}. \end{cases} \quad (2)$$

³Xie, Lingxi, et al. "DisturbLabel: Regularizing CNN on the Loss Layer." CVPR 2016.

Regularization through Gradient Noise⁴

- **Idea:** Add noise to gradient

$$g_t \leftarrow g_t + N(0, \sigma_t^2)$$



⁴Neelakantan, Arvind, et al. "Adding gradient noise improves learning for very deep networks." arXiv preprint arXiv:1511.06807 (2015)

Regularization through Gradient Noise⁴

- **Idea:** Add noise to gradient

$$g_t \leftarrow g_t + N(0, \sigma_t^2)$$

- Annealed Gaussian noise by decaying variance

$$\sigma_t^2 = \frac{\eta}{(1+t)^\gamma}$$

NPTEL

⁴Neelakantan, Arvind, et al. "Adding gradient noise improves learning for very deep networks." arXiv preprint arXiv:1511.06807 (2015)

Regularization through Gradient Noise⁴

- **Idea:** Add noise to gradient

$$g_t \leftarrow g_t + N(0, \sigma_t^2)$$

- Annealed Gaussian noise by decaying variance

$$\sigma_t^2 = \frac{\eta}{(1+t)^\gamma}$$

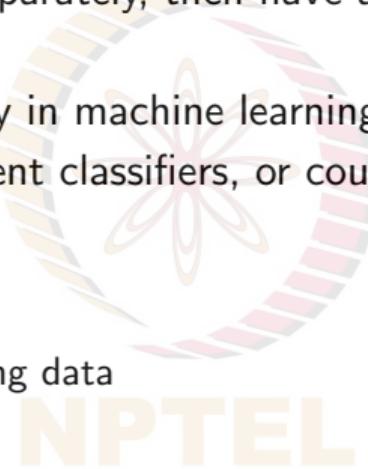


- Showed significant improvement in performance in certain applications

⁴Neelakantan, Arvind, et al. "Adding gradient noise improves learning for very deep networks." arXiv preprint arXiv:1511.06807 (2015)

Ensemble Methods

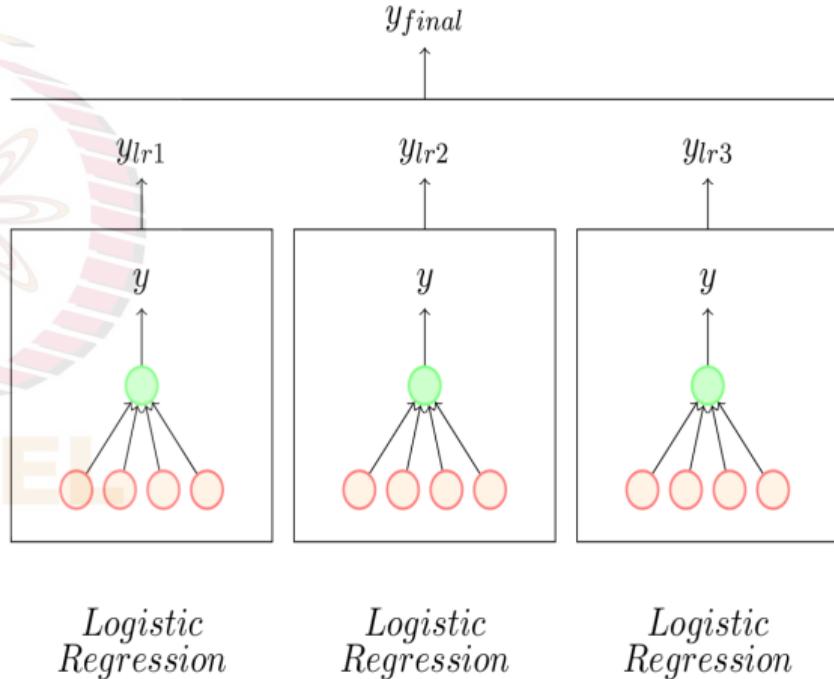
- Train several different models separately, then have all models vote on the output for test examples
- An example of a general strategy in machine learning called **model averaging**
- Models can correspond to different classifiers, or could be different instances of same classifier trained with:
 - different hyperparameters
 - different features
 - different samples of the training data



Credit: Ali Ghodsi, Univ of Waterloo; Mitesh Khapra, IIT Madras

Ensemble Methods

- **Bagging** (short for bootstrap aggregating): reduces generalization error by forming an ensemble using different instances of same classifier
- For e.g., consider a set of k logistic regression models
- Given a dataset, construct multiple training sets by sampling with replacement (T_1, T_2, \dots, T_k)
- Train i^{th} instance of classifier using training set T_i



Credit: Ali Ghodsi, Univ of Waterloo; Mitesh Khapra, IIT Madras

Ensemble Methods

- Suppose that each model makes an error ε_i on a test example

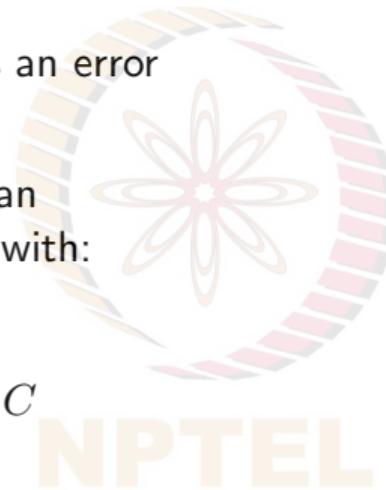


Ensemble Methods

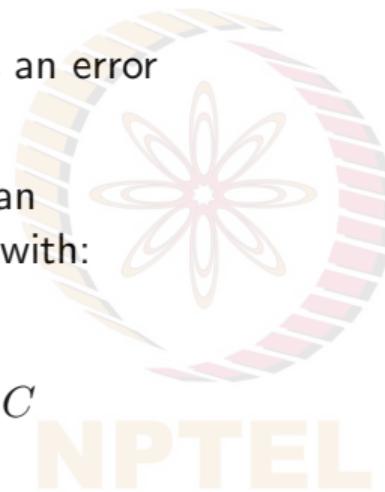
- Suppose that each model makes an error ε_i on a test example
- Let ε_i be drawn from a zero-mean multivariate normal distribution with:

$$\text{Variance} = \mathbb{E}[\varepsilon_i^2] = V$$

$$\text{Covariance} = \mathbb{E}[\varepsilon_i \varepsilon_j] = C$$



Ensemble Methods



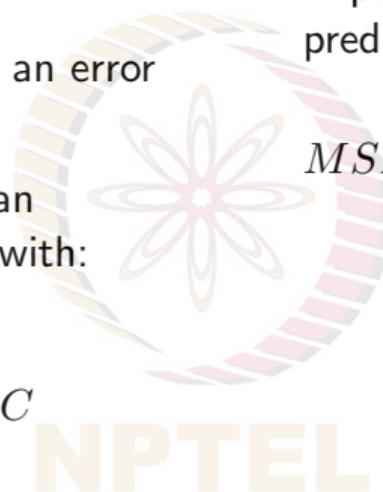
- Suppose that each model makes an error ε_i on a test example
- Let ε_i be drawn from a zero-mean multivariate normal distribution with:

$$\text{Variance} = \mathbb{E}[\varepsilon_i^2] = V$$

$$\text{Covariance} = \mathbb{E}[\varepsilon_i \varepsilon_j] = C$$

- Error made by average prediction of all models is $\frac{1}{k} \sum_i \varepsilon_i$

Ensemble Methods



- Suppose that each model makes an error ε_i on a test example
- Let ε_i be drawn from a zero-mean multivariate normal distribution with:

$$\text{Variance} = \mathbb{E}[\varepsilon_i^2] = V$$

$$\text{Covariance} = \mathbb{E}[\varepsilon_i \varepsilon_j] = C$$

- Error made by average prediction of all models is $\frac{1}{k} \sum_i \varepsilon_i$

- Expected squared error of ensemble predictor is:

$$\begin{aligned} MSE &= \mathbb{E}\left[\left(\frac{1}{k} \sum_i \varepsilon_i\right)^2\right] \\ &= \frac{1}{k^2} \mathbb{E}\left[\sum_i \sum_{j=i} \varepsilon_i \varepsilon_j + \sum_i \sum_{j \neq i} \varepsilon_i \varepsilon_j\right] \\ &= \frac{1}{k^2} \mathbb{E}\left[\sum_i \varepsilon_i^2 + \sum_i \sum_{j \neq i} \varepsilon_i \varepsilon_j\right] \\ &= \frac{1}{k^2}(kV + k(k-1)C) \\ &= \frac{1}{k}V + \frac{k-1}{k}C \end{aligned}$$

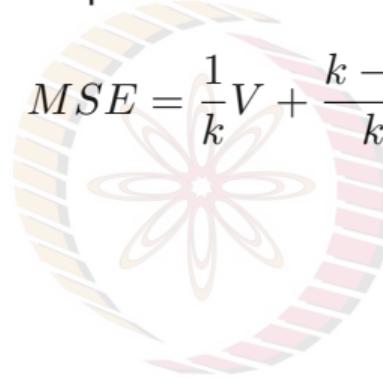
Credit: Ali Ghodsi, Univ of Waterloo; Mitesh Khapra, IIT Madras

Ensemble Methods

- Expected squared error of ensemble predictor:

$$MSE = \frac{1}{k}V + \frac{k-1}{k}C$$

What does this tell you?



NPTEL

Ensemble Methods

- Expected squared error of ensemble predictor:

$$MSE = \frac{1}{k}V + \frac{k-1}{k}C$$

What does this tell you?

- If errors of model are perfectly correlated, then $V = C$ and $MSE = V$, i.e. bagging does not help: the MSE of ensemble is as bad as individual models

NPTEL

Ensemble Methods

- Expected squared error of ensemble predictor:

$$MSE = \frac{1}{k}V + \frac{k-1}{k}C$$

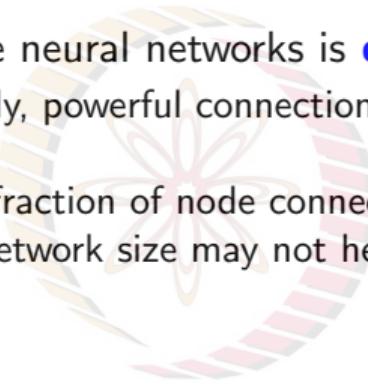
What does this tell you?

- If errors of model are perfectly correlated, then $V = C$ and $MSE = V$, i.e. bagging does not help: the MSE of ensemble is as bad as individual models
- If errors of model are independent or uncorrelated, then $C = 0$ and MSE of ensemble reduces to $\frac{1}{k}V$
- On average, **ensemble will perform at least as well as its individual members**

Credit: Ali Ghodsi, Univ of Waterloo; Mitesh Khapra, IIT Madras

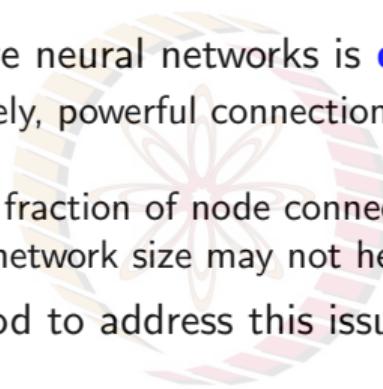
Dropout

- One major issue in learning large neural networks is **co-adaptation**
 - As network is trained iteratively, powerful connections are learned more while weaker ones are ignored
 - After many iterations, only a fraction of node connections participate
 - Therefore, expanding neural network size may not help



Dropout

- One major issue in learning large neural networks is **co-adaptation**
 - As network is trained iteratively, powerful connections are learned more while weaker ones are ignored
 - After many iterations, only a fraction of node connections participate
 - Therefore, expanding neural network size may not help
- **Dropout**: Regularization method to address this issue

The logo consists of the letters "NPTEL" in a bold, sans-serif font. The letters are colored in a gradient: N and P are orange, T is yellow, E is light green, and L is teal.

NPTEL

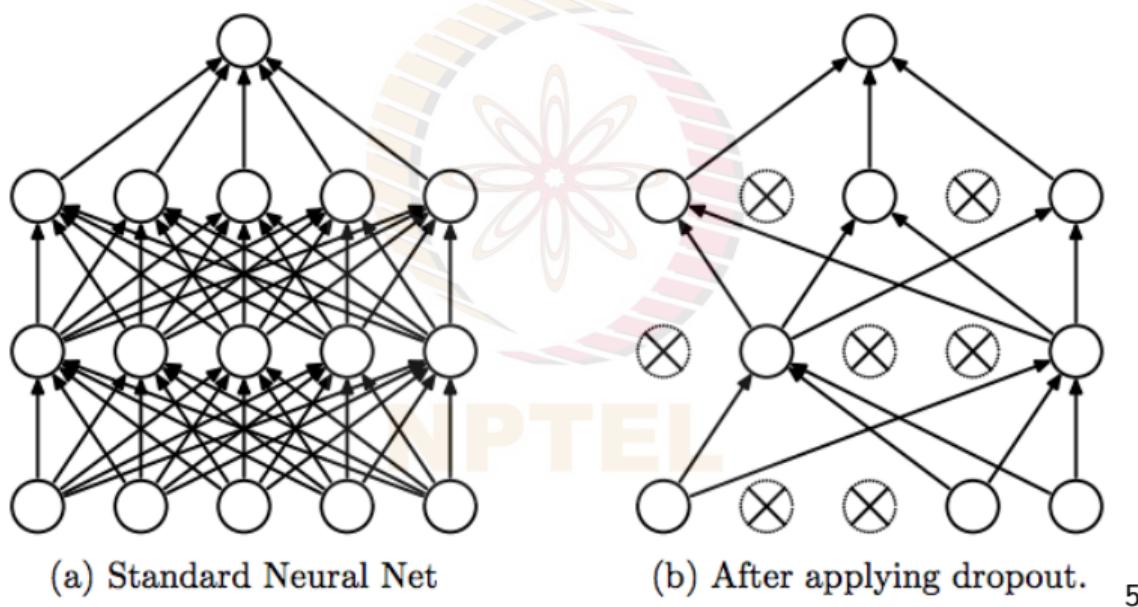
Dropout

- One major issue in learning large neural networks is **co-adaptation**
 - As network is trained iteratively, powerful connections are learned more while weaker ones are ignored
 - After many iterations, only a fraction of node connections participate
 - Therefore, expanding neural network size may not help
- **Dropout**: Regularization method to address this issue
- **Training Phase**: For each hidden layer, for each training sample, for each iteration, ignore (zero out) a random fraction, p , of nodes (and corresponding activations)

Dropout

- One major issue in learning large neural networks is **co-adaptation**
 - As network is trained iteratively, powerful connections are learned more while weaker ones are ignored
 - After many iterations, only a fraction of node connections participate
 - Therefore, expanding neural network size may not help
- **Dropout**: Regularization method to address this issue
- **Training Phase**: For each hidden layer, for each training sample, for each iteration, ignore (zero out) a random fraction, p , of nodes (and corresponding activations)
- **Test Phase**: Use all activations, but reduce them by factor p (to account for missing activations during training)

Dropout



⁵Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

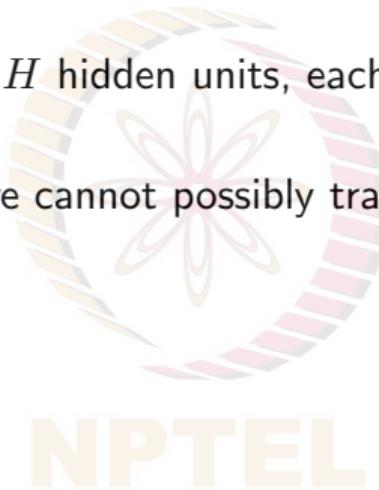
Dropout

- Recall ensemble methods: with H hidden units, each of which can be dropped, we can have 2^H possible models



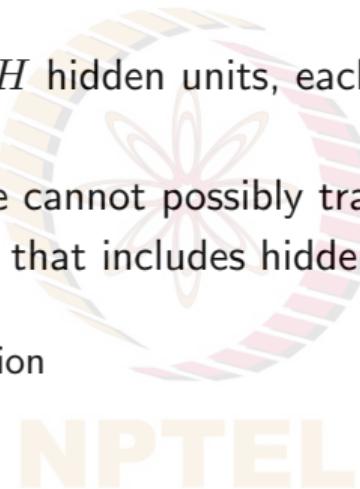
Dropout

- Recall ensemble methods: with H hidden units, each of which can be dropped, we can have 2^H possible models
- This is prohibitively large and we cannot possibly train so many networks



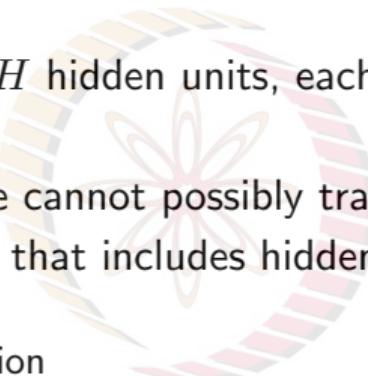
Dropout

- Recall ensemble methods: with H hidden units, each of which can be dropped, we can have 2^H possible models
- This is prohibitively large and we cannot possibly train so many networks
- **Trick:** Each of the 2^{H-1} models that includes hidden unit h must share the same weight for the unit
 - serves as a form of regularization
 - makes the models cooperate



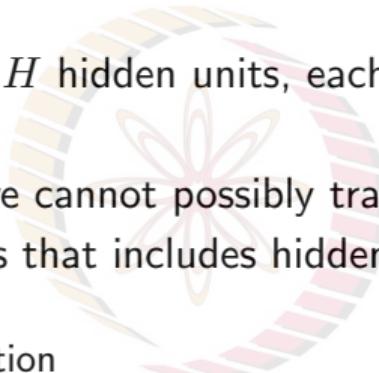
Dropout

- Recall ensemble methods: with H hidden units, each of which can be dropped, we can have 2^H possible models
- This is prohibitively large and we cannot possibly train so many networks
- **Trick:** Each of the 2^{H-1} models that includes hidden unit h must share the same weight for the unit
 - serves as a form of regularization
 - makes the models cooperate
- Including all hidden units at test time with scaling of $\frac{1}{2}$ is equivalent to computing geometric mean of all 2^H models.



Dropout

- Recall ensemble methods: with H hidden units, each of which can be dropped, we can have 2^H possible models
- This is prohibitively large and we cannot possibly train so many networks
- **Trick:** Each of the 2^{H-1} models that includes hidden unit h must share the same weight for the unit
 - serves as a form of regularization
 - makes the models cooperate
- Including all hidden units at test time with scaling of $\frac{1}{2}$ is equivalent to computing geometric mean of all 2^H models. (How?)



Dropout for Single Non-Linear Unit

- Consider a logistic sigmoidal function on input x :

$$o = \sigma(x) = \frac{1}{1 + ce^{-\lambda x}} \quad c \geq 0$$



Dropout for Single Non-Linear Unit

- Consider a logistic sigmoidal function on input x :

$$o = \sigma(x) = \frac{1}{1 + ce^{-\lambda x}} \quad c \geq 0$$

- 2^n possible sub-networks indexed by k



Dropout for Single Non-Linear Unit

- Consider a logistic sigmoidal function on input x :

$$o = \sigma(x) = \frac{1}{1 + ce^{-\lambda x}} \quad c \geq 0$$

- 2^n possible sub-networks indexed by k
- Geometric mean of outputs from all sub-networks: $G = \prod_k o_k^{\frac{1}{2^n}}$



Dropout for Single Non-Linear Unit

- Consider a logistic sigmoidal function on input x :

$$o = \sigma(x) = \frac{1}{1 + ce^{-\lambda x}} \quad c \geq 0$$

- 2^n possible sub-networks indexed by k
- Geometric mean of outputs from all sub-networks: $G = \prod_k o_k^{\frac{1}{2^n}}$
- Geometric mean of complementary outputs: $G' = \prod_k (1 - o_k)^{\frac{1}{2^n}}$



Dropout for Single Non-Linear Unit

- Consider a logistic sigmoidal function on input x :

$$o = \sigma(x) = \frac{1}{1 + ce^{-\lambda x}} \quad c >= 0$$

- 2^n possible sub-networks indexed by k
- Geometric mean of outputs from all sub-networks: $G = \prod_k o_k^{\frac{1}{2^n}}$
- Geometric mean of complementary outputs: $G' = \prod_k (1 - o_k)^{\frac{1}{2^n}}$

- Normalized geometric mean $NGM = \frac{G}{G+G'}$. Hence:

$$\begin{aligned} NGM &= \frac{1}{1 + \left(\prod_k \frac{1-\sigma(x_k)}{\sigma(x_k)} \right)^{\frac{1}{2^n}}} \\ &= \frac{1}{1 + \left(\prod_k ce^{-\lambda x_k} \right)^{\frac{1}{2^n}}} \left(\because \frac{1 - \sigma(x)}{\sigma(x)} = ce^{-\lambda x} \right) \\ &= \frac{1}{1 + c[e^{-\lambda \sum_k x_k / 2^n}]} = \sigma(\mathbb{E}[x]) \end{aligned}$$

where $\mathbb{E}[x] = \sum_k x_k / 2^n$

Dropout for Single Non-Linear Unit

- Consider a logistic sigmoidal function on input x :

$$o = \sigma(x) = \frac{1}{1 + ce^{-\lambda x}} \quad c >= 0$$

- 2^n possible sub-networks indexed by k
- Geometric mean of outputs from all sub-networks: $G = \prod_k o_k^{\frac{1}{2^n}}$
- Geometric mean of complementary outputs: $G' = \prod_k (1 - o_k)^{\frac{1}{2^n}}$

NGM is equivalent to output of overall network with weights divided by two

- Normalized geometric mean $NGM = \frac{G}{G+G'}$. Hence:

$$\begin{aligned} NGM &= \frac{1}{1 + \left(\prod_k \frac{1-\sigma(x_k)}{\sigma(x_k)} \right)^{\frac{1}{2^n}}} \\ &= \frac{1}{1 + \left(\prod_k ce^{-\lambda x_k} \right)^{\frac{1}{2^n}}} \left(\because \frac{1 - \sigma(x)}{\sigma(x)} = ce^{-\lambda x} \right) \\ &= \frac{1}{1 + c[e^{-\lambda \sum_k x_k / 2^n}]} = \sigma(\mathbb{E}[x]) \end{aligned}$$

where $\mathbb{E}[x] = \sum_k x_k / 2^n$

Homework

Readings

- Deep Learning book, [Chapter 7](#), Sections 7.1-7.5, 7.8, 7.12
- [Tutorial on Dropout](#) for more information (Optional)

Exercise

- Show that adding Gaussian noise (with zero mean) to input is equivalent to L_2 weight decay when loss function is MSE.

References I

-  Geoffrey E. Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580 (2012). arXiv: [1207.0580](https://arxiv.org/abs/1207.0580).
-  Pierre Baldi and Peter Sadowski. "The dropout learning algorithm". In: *Artificial intelligence* 210 (2014), pp. 78–122.
-  Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.
-  David Warde-Farley et al. "An empirical analysis of dropout in piecewise linear networks". In: *CoRR* abs/1312.6197 (2014).
-  Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
-  Terrance DeVries and Graham W Taylor. "Improved Regularization of Convolutional Neural Networks with Cutout". In: *arXiv preprint arXiv:1708.04552* (2017).

References II



Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018.



Ghodsi, Ali, *STAT 946 - Deep Learning (Fall 2015)*. URL:
https://uwaterloo.ca/data-analytics/sites/ca.data-analytics/files/uploads/files/f15_stat946_deep_learning_outline_1.pdf (visited on 06/05/2020).



Khapra, Mitesh M., *CS 7015 - Deep Learning (Spring 2019)*. URL:
<https://www.cse.iitm.ac.in/~miteshk/CS7015.html> (visited on 06/03/2020).

NPTEL