Introduction to Deep Learning

Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
arijit@iitp.ac.in

Regularization

Introduction

- In machine learning, target is to make an algorithm performs well not only on training data but also on new data
- Many strategies exist to reduce test error at the cost of training error
- Any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error
- Objectives
  - To encode prior knowledge
  - Constraints and penalties are designed to express generic preference for simpler model

Regularization in DL

- In DL regularization works by trading increased bias for reduced variance
- Consider the following scenario
  - Excluded the true data generating process
    - Underfitting, inducing bias
  - Matched the true data generating process
    - Desired one
  - Included the generating process but also many other generating process
    - Overfitting, variance dominates
  - Goal of regularizer is to take an model overfit zone to desired zone

Norm penalties

- Most of the regularization approaches are based on limiting the capacity of the model
- Objective function becomes $\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$
  - α — Hyperparameter denotes relative contribution
  - Minimization of $\tilde{J}$ implies minimization of J
  - Ω penalizes only the weight of affine transform
    - Bias remain unregularized
    - Regularizing bias may lead to underfitting

L^2 parameter regularization

- Weights are closer to origin as $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$
  - Also known as ridge regression or Tikhonov regression
- Objective function $\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$

L^2 parameter regularization

- Weights are closer to origin as $\Omega(\theta) = \frac{1}{2}\|\mathbf{w}\|_2^2$
  - Also known as ridge regression or Tikhonov regression
- Objective function $\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \dfrac{\alpha}{2}\mathbf{w}^T\mathbf{w} + J(\mathbf{w}; \mathbf{X}, \mathbf{y})$
- Gradient $\nabla_{\mathbf{w}}\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha\mathbf{w} + \nabla_{\mathbf{w}}J(\mathbf{w}; \mathbf{X}, \mathbf{y})$

L^2 parameter regularization

- Weights are closer to origin as $\Omega(\theta) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$
  - Also known as ridge regression or Tikhonov regression
- Objective function $\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$
- Gradient $\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$
- New weights

$$\boldsymbol{w} = \boldsymbol{w} - \epsilon(\alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}))$$

L^2 parameter regularization

- Weights are closer to origin as $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$
  - Also known as ridge regression or Tikhonov regression
- Objective function $\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \dfrac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$
- Gradient $\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$
- New weights

$$\boldsymbol{w} \;=\; \boldsymbol{w} - \epsilon(\alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})) \;=\; \boldsymbol{w}(1 - \epsilon\alpha) - \epsilon\nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

L^2 parameter regularization

- **Weights are closer to origin as $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$**
  - **Also known as ridge regression or Tikhonov regression**
- **Objective function $\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$**
- **Gradient $\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$**
- **New weights**

$$\boldsymbol{w} = \boldsymbol{w} - \epsilon(\alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})) = \boldsymbol{w}(1 - \epsilon\alpha) - \epsilon\nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

- **Assume quadratic nature of curve in the neighborhood of $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$**
  - **$J(\boldsymbol{w})$ — unregularized cost**
  - **Perfect scenario for linear regression with MSE**

Jacobian & Hessian

- **Derivative of a function having single input and single output — $\frac{dy}{dx}$**
- **Derivative of function having vector input and vector output that is, $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$**
  - **Jacobian $\boldsymbol{J} \in \mathbb{R}^{n \times m}$ of f defined as $J_{i,j} = \frac{\partial}{\partial x_j} f(\boldsymbol{x})_i$**
- **Second derivative is also required sometime**
  - **For example, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\frac{\partial^2}{\partial x_i \partial x_j} f$**
  - **If second derivative is 0, then there is no curvature**
- **Hessian matrix $\boldsymbol{H}(f)(x)_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$**

Jacobian & Hessian

- Derivative of a function having single input and single output — $\frac{dy}{dx}$
- Derivative of function having vector input and vector output that is, $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$
  - Jacobian $\boldsymbol{J} \in \mathbb{R}^{n \times m}$ of f defined as $J_{i,j} = \frac{\partial}{\partial x_j} f(\boldsymbol{x})_i$
- Second derivative is also required sometime
  - For example, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\frac{\partial^2}{\partial x_i \partial x_j} f$
  - If second derivative is 0, then there is no curvature
- Hessian matrix $\boldsymbol{H}(f)(x)_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$
  - Jacobian of gradient
  - Symmetric

Directional derivative

- **The directional derivative of a scalar function $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n)$ along a vector $\mathbf{v} = (v_1, \ldots, v_n)$ is given by**

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x})}{h}$$

- **If f is differentiable at point x then**

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v}$$

Taylor series expansion

- **A real valued function differentiable at point x_0 can be expressed as**

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f^{(3)}(x_0)}{3!}(x - x_0)^3 + \cdots .$$

Taylor series expansion

- **A real valued function differentiable at point x_0 can be expressed as**

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f^{(3)}(x_0)}{3!}(x - x_0)^3 + \cdots .$$

- **When input is a vector**

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^T \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^T \boldsymbol{H} (\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

  - **$\boldsymbol{g}$ — gradient at $\boldsymbol{x}^{(0)}$, $\boldsymbol{H}$ — Hessian at $\boldsymbol{x}^{(0)}$**

Taylor series expansion

- A real valued function differentiable at point x_0 can be expressed as

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f^{(3)}(x_0)}{3!}(x - x_0)^3 + \cdots .$$

- When input is a vector

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^T \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^T \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

  - $\boldsymbol{g}$ — gradient at $\boldsymbol{x}^{(0)}$, $\boldsymbol{H}$ — Hessian at $\boldsymbol{x}^{(0)}$

- If ϵ is the learning rate, then $f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) = f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^T \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g}$

Quadratic approximation

- Let $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$ optimum weights for minimal unregularized cost
- If the objective function is quadratic then $\hat{J}(\theta) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$
  - $\boldsymbol{H}$ is the Hessian matrix of J with respect to $\boldsymbol{w}$ at $\boldsymbol{w}^*$
  - No first order term as $\boldsymbol{w}^*$ is minimum
  - $\boldsymbol{H}$ is positive semidefinite
- Minimum of $\hat{J}$ occurs when $\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) = 0$
- With weight decay we have

$$\alpha\tilde{\boldsymbol{w}} + \boldsymbol{H}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^*) = 0 \Rightarrow (\boldsymbol{H} + \alpha\boldsymbol{I})\tilde{\boldsymbol{w}} = \boldsymbol{H}\boldsymbol{w}^* \Rightarrow \tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha\boldsymbol{I})^{-1}\boldsymbol{H}\boldsymbol{w}^*$$

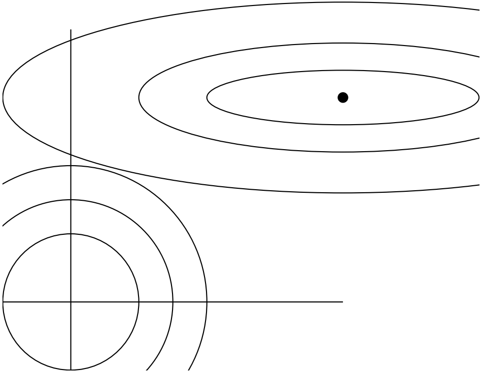
Quadratic approximation (contd)

- As $\alpha \to 0$, regularized solution $\hat{w}$ approaches to w^*
- As $\alpha \to \infty$
  - H is symmetric, therefore $H = Q\Lambda Q^T$. Now we have

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

  - Weight decay rescale w^* along the eigen vector of H
    - Component of w^* that is aligned to i-th eigen vector, will be rescaled by a factor of $\frac{\lambda_i}{\lambda_i + \alpha}$
    - $\lambda_i \gg \alpha$ — regularization effect is small

L^2 Norm



Linear regression

- For linear regression cost function is $(\boldsymbol{Xw} - \boldsymbol{y})^T(\boldsymbol{Xw} - \boldsymbol{y})$
- Using L^2 regularization we have $(\boldsymbol{Xw} - \boldsymbol{y})^T(\boldsymbol{Xw} - \boldsymbol{y}) + \frac{1}{2}\alpha\boldsymbol{w}^T\boldsymbol{w}$

Linear regression

- For linear regression cost function is $(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$
- Using L^2 regularization we have $(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) + \frac{1}{2}\alpha\mathbf{w}^T\mathbf{w}$
- Solution for normal equation $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Linear regression

- For linear regression cost function is $(\boldsymbol{Xw} - \boldsymbol{y})^T(\boldsymbol{Xw} - \boldsymbol{y})$
- Using L^2 regularization we have $(\boldsymbol{Xw} - \boldsymbol{y})^T(\boldsymbol{Xw} - \boldsymbol{y}) + \frac{1}{2}\alpha\boldsymbol{w}^T\boldsymbol{w}$
- Solution for normal equation $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$
- Solution for with weight decay $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

L^1 regularization

- Formally it is defined as $\Omega(\theta) = \|\mathbf{w}\|_1 = \sum_i |w_i|$
- Regularized objective function will be $\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha\|\mathbf{w}\|_1 + J(\mathbf{w}; \mathbf{X}, \mathbf{y})$

L^1 regularization

- Formally it is defined as $\Omega(\theta) = \|\mathbf{w}\|_1 = \sum_i |w_i|$
- Regularized objective function will be $\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha \|\mathbf{w}\|_1 + J(\mathbf{w}; \mathbf{X}, \mathbf{y})$
- The gradient will be $\nabla_{\mathbf{w}} \tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha \mathbf{sign}(\mathbf{w}) + \nabla_{\mathbf{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y})$
  - Gradient does not scale linearly compared to L^2 regularization
- Taylor series expansion with approximation provides $\nabla_{\mathbf{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$
- Simplification can be made by assuming $\mathbf{H}$ to be diagonal
  - Apply PCA on the input dataset

L^1 regularization

- Quadratic approximation of L^1 regularization objective function becomes $\hat{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = J((\mathbf{w}^*; \mathbf{X}, \mathbf{y}) + \sum_i \left[\frac{1}{2} H_{i,i} (\mathbf{w}_i - \mathbf{w}_i^*)^2 + \alpha |w_i|\right]$
- So, analytical solution in each dimension will be $w_i = \textbf{sign}(w_i^*) \max \left\{ |w_i^*| - \frac{\alpha}{H_{i,i}}, 0 \right\}$
- Consider the situation when $w_i^* > 0$
  - If $w_i^* \leq \frac{\alpha}{H_{i,i}}$, optimal value for w_i will be 0 under regularization
  - If $w_i^* > \frac{\alpha}{H_{i,i}}$, w_i moves towards 0 with a distance equal to $\frac{\alpha}{H_{i,i}}$

Constrained optimization

- Cost function regularized by norm penalty is given by

$$\tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\boldsymbol{\theta})$$

- Let us assume $f(x)$ needs to be optimized under a set of equality constraints $g^{(i)}(x) = 0$ and inequality constraints $h^{(j)}(x) \leq 0$, then generalized Lagrangian is then defined as

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \sum_i \lambda_i g^{(i)}(\mathbf{x}) + \sum_j \alpha_j h^{(j)}(\mathbf{x})$$

- If there exists a solution then

$$\min_x \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha} \geq 0} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \min_x f(\mathbf{x})$$

  - This can be solved by $\nabla_{\mathbf{x},\lambda,\alpha} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = 0$

Constraint optimization (contd.)

- Suppose $\Omega(\boldsymbol{\theta}) < k$ needs to be satisfied. Then regularization equation becomes

$$L(\boldsymbol{\theta}, \alpha; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha(\Omega(\boldsymbol{\theta}) - k)$$

  - Solution to the constrained problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \max_{\alpha > 0} L(\boldsymbol{\theta}, \alpha)$$

Dataset augmentation

- If data are limited, fake data can be added to training set
  - Computer vision problem
  - Speech recognition
- Easiest for classification problem
- Very effective in object recognition problem
  - Translating
  - Rotating
  - Scaling
    - Need to be careful for 'b' and 'd' or '6' and '9'
- Injecting noise to input data can be viewed as data augmentation

Multitask learning

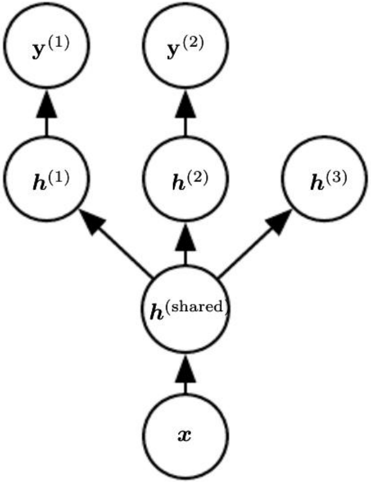


Image source: Deep Learning Book

Early stopping

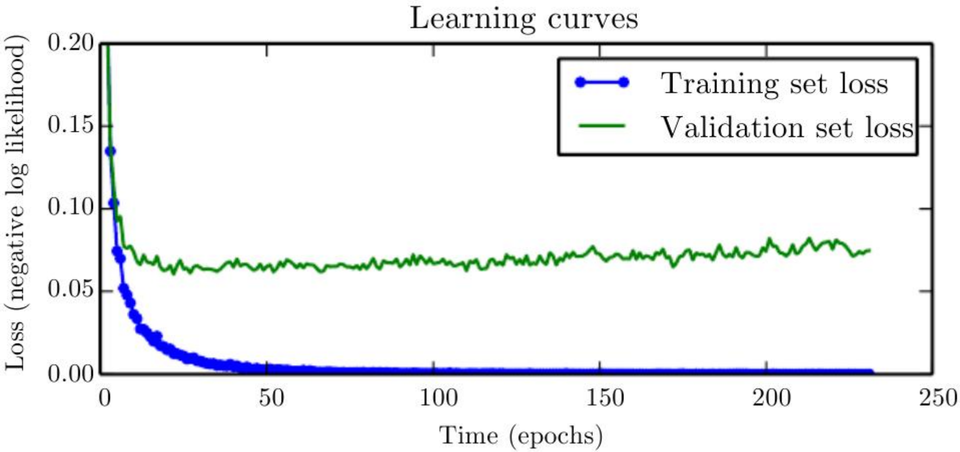


Image source: Deep Learning Book

Early stopping approach

- Initialize the parameters
- Run training algorithm for n steps and update $i = i + n$
- Compute error on the validation set (v')
- If v' is less than previous best, then update the same. Start step 2 again
- If v' is more than the previous best, then increment the count that stores the number of such occurrences. If the count is less than a threshold go to step 2, otherwise exit.

Early stopping (contd)

- Number of training step is a hyperparameter
  - Most hyperparameters that control model capacity have U-shaped curve
- Additional cost for this approach is to store the parameters
- Requires a validation set
  - It will have two passes
    - First pass uses only training data for update of the parameters
    - Second pass uses both training and validation data for update of the parameters
  - Possible strategies
    - Initialize the model again, retrain on all data, train for the same number of steps as obtained by early stopping in pass 1
    - Keep the parameters obtained from the first round, continue training using all data until the loss is less than the training loss at the early stopping point
- It reduces computational cost as it limits the number of iteration
- Provides regularization without any penalty

Early stopping as regularizer

- Let us assume τ training iteration, ϵ learning rate
  - $\epsilon\tau$ — measures effective capacity
- We have, $\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)\boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$ and $\nabla_{\boldsymbol{w}}\hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$
- Assume $\boldsymbol{w}^{(0)} = 0$

Early stopping as regularizer (contd.)

- Approximate behavior of gradient descent provides

Early stopping as regularizer (contd.)

- **Approximate behavior of gradient descent provides**

$$\boldsymbol{w}^{(\tau)} = \boldsymbol{w}^{(\tau-1)} - \epsilon \nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}^{(\tau-1)})$$

Early stopping as regularizer (contd.)

- Approximate behavior of gradient descent provides

$$\begin{aligned}\boldsymbol{w}^{(\tau)} &= \boldsymbol{w}^{(\tau-1)} - \epsilon \nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}^{(\tau-1)}) \\ \boldsymbol{w}^{(\tau)} &= \boldsymbol{w}^{(\tau-1)} - \epsilon \boldsymbol{H}(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^{*})\end{aligned}$$

Early stopping as regularizer (contd.)

- **Approximate behavior of gradient descent provides**

$$\begin{aligned}
\mathbf{w}^{(\tau)} &= \mathbf{w}^{(\tau-1)} - \epsilon \nabla_{\mathbf{w}} \hat{J}(\mathbf{w}^{(\tau-1)}) \\
\mathbf{w}^{(\tau)} &= \mathbf{w}^{(\tau-1)} - \epsilon \mathbf{H}(\mathbf{w}^{(\tau-1)} - \mathbf{w}^{*}) \\
\mathbf{w}^{(\tau)} - \mathbf{w}^{*} &= (\mathbf{I} - \epsilon \mathbf{H})(\mathbf{w}^{(\tau-1)} - \mathbf{w}^{*})
\end{aligned}$$

Early stopping as regularizer (contd.)

- Approximate behavior of gradient descent provides

$$
\begin{aligned}
\mathbf{w}^{(\tau)} &= \mathbf{w}^{(\tau-1)} - \epsilon \nabla_{\mathbf{w}} \hat{J}(\mathbf{w}^{(\tau-1)}) \\
\mathbf{w}^{(\tau)} &= \mathbf{w}^{(\tau-1)} - \epsilon \mathbf{H}(\mathbf{w}^{(\tau-1)} - \mathbf{w}^{*}) \\
\mathbf{w}^{(\tau)} - \mathbf{w}^{*} &= (\mathbf{I} - \epsilon \mathbf{H})(\mathbf{w}^{(\tau-1)} - \mathbf{w}^{*}) \\
\mathbf{w}^{(\tau)} - \mathbf{w}^{*} &= (\mathbf{I} - \epsilon \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^{T})(\mathbf{w}^{(\tau-1)} - \mathbf{w}^{*})
\end{aligned}
$$

Early stopping as regularizer (contd.)

- **Approximate behavior of gradient descent provides**

$$
\begin{aligned}
\mathbf{w}^{(\tau)} &= \mathbf{w}^{(\tau-1)} - \epsilon \nabla_{\mathbf{w}} \hat{J}(\mathbf{w}^{(\tau-1)}) \\
\mathbf{w}^{(\tau)} &= \mathbf{w}^{(\tau-1)} - \epsilon \mathbf{H}(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*) \\
\mathbf{w}^{(\tau)} - \mathbf{w}^* &= (\mathbf{I} - \epsilon \mathbf{H})(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*) \\
\mathbf{w}^{(\tau)} - \mathbf{w}^* &= (\mathbf{I} - \epsilon \mathbf{Q} \Lambda \mathbf{Q}^T)(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*) \\
\mathbf{Q}^T(\mathbf{w}^{(\tau)} - \mathbf{w}^*) &= (\mathbf{I} - \epsilon \Lambda)\mathbf{Q}^T(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)
\end{aligned}
$$

Early stopping as regularizer (contd.)

- **Approximate behavior of gradient descent provides**

$$
\begin{aligned}
\boldsymbol{w}^{(\tau)} &= \boldsymbol{w}^{(\tau-1)} - \epsilon \nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}^{(\tau-1)}) \\
\boldsymbol{w}^{(\tau)} &= \boldsymbol{w}^{(\tau-1)} - \epsilon \boldsymbol{H}(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^{*}) \\
\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^{*} &= (\boldsymbol{I} - \epsilon \boldsymbol{H})(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^{*}) \\
\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^{*} &= (\boldsymbol{I} - \epsilon \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{T})(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^{*}) \\
\boldsymbol{Q}^{T}(\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^{*}) &= (\boldsymbol{I} - \epsilon \boldsymbol{\Lambda})\boldsymbol{Q}^{T}(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^{*}) \\
\boldsymbol{Q}^{T}\boldsymbol{w}^{(\tau)} &= [\boldsymbol{I} - (\boldsymbol{I} - \epsilon \boldsymbol{\Lambda})^{\tau}]\boldsymbol{Q}^{T}\boldsymbol{w}^{*}
\end{aligned}
$$

Early stopping as regularizer (contd)

- Assuming $\mathbf{w}^{(0)} = 0$ and ϵ is small value such that $|1 - \epsilon\lambda_i| < 1$
- From L^2 regularization, we have

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

- Therefore we have, $(\mathbf{I} - \epsilon\mathbf{\Lambda})^{\tau} = (\mathbf{\Lambda} + \alpha\mathbf{I})^{-1}\alpha$
- Hence, $\tau \approx \frac{1}{\epsilon\alpha}$, $\alpha \approx \frac{1}{\tau\epsilon}$

Bagging

- Also known as Bootstrap aggregating
- Reduces generalization error by combining several models
- Train multiple models then vote on output for the test example
  - Also known as model averaging, ensemble method
- Suppose we have k regression model and each model makes an error ϵ_i such that $\mathbb{E}(\epsilon_i) = 0$, $\mathbb{E}(\epsilon_i^2) = v$, $\mathbb{E}(\epsilon_i \epsilon_j) = c$
- Error made by average prediction $\frac{1}{k} \sum_i \epsilon_i$

Bagging (contd.)

- **Expected mean square error**

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i\left(\epsilon_i^2 + \sum_{i\neq j}\epsilon_i\epsilon_j\right)\right] = \frac{v}{k} + \frac{k-1}{k}c$$

Dropout

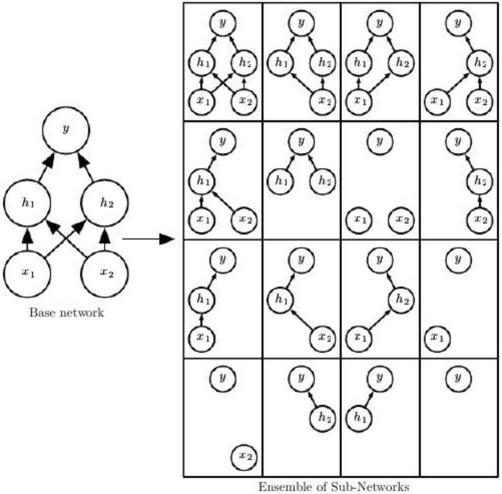


Image source: Deep Learning Book

Dropout

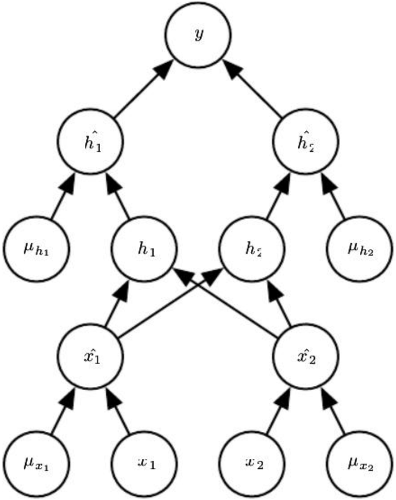


Image source: Deep Learning Book

Adversarial training

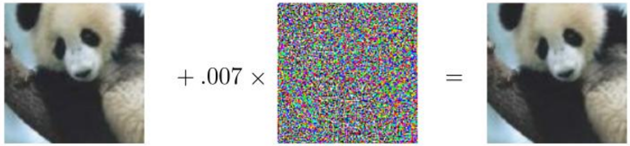


Image source: Deep Learning Book