CS551: Introduction to Deep Learning

Regularization

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

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

Trade off Bias and Variance

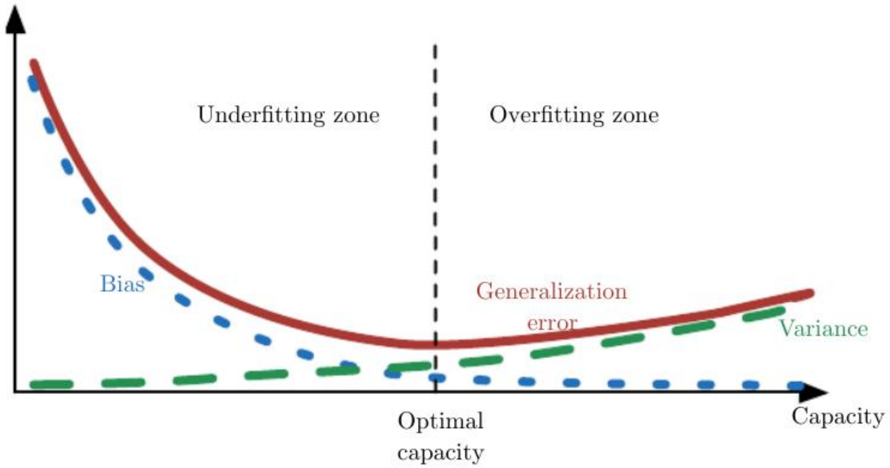
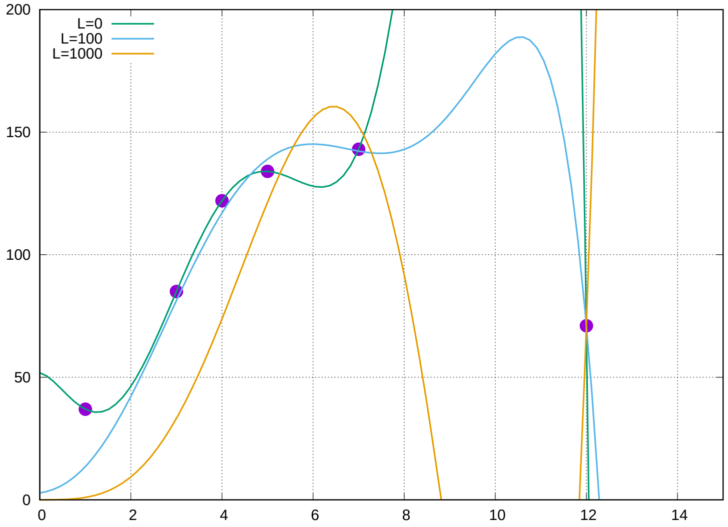


Image source: Deep Learning Book

Norm penalties

- Most of the regularization approaches are based on limiting the capacity of the model
- Objective function becomes $\tilde{J}(\boldsymbol{\theta}; \mathsf{X}, \mathsf{y}) = J(\boldsymbol{\theta}; \mathsf{X}, \mathsf{y}) + \alpha\Omega(\boldsymbol{\theta})$
  - 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

Example: Weight decay



L^2 parameter regularization

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

L^2 parameter regularization

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

L^2 parameter regularization

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

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

L^2 parameter regularization

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

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

L^2 parameter regularization

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

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

- Assuming quadratic nature of curve in the neighborhood of $\mathrm{w}^* = \arg\min_{\mathrm{w}} J(\mathrm{w})$
  - $J(\mathrm{w})$ — 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 $\mathrm{J} \in \mathbb{R}^{n \times m}$ of f defined as $J_{i,j} = \frac{\partial}{\partial x_j} f(\mathrm{x})_i$
- Second derivative is also required sometime
    - For example, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\dfrac{\partial^2}{\partial x_i \partial x_j} f$
    - If second derivative is 0, then there is no curvature
- Hessian matrix $\mathrm{H}(f)(x)_{ij} = \dfrac{\partial^2}{\partial x_i \partial x_j} f(\mathrm{x})$

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, $f : \mathbb{R}^m \to \mathbb{R}^n$
  - Jacobian $\mathrm{J} \in \mathbb{R}^{n \times m}$ of f defined as $J_{i,j} = \frac{\partial}{\partial x_j} f(\mathrm{x})_i$
- Second derivative is also required sometime
  - For example, $f : \mathbb{R}^n \to \mathbb{R}$, $\frac{\partial^2}{\partial x_i \partial x_j} f$
  - If second derivative is 0, then there is no curvature
- Hessian matrix $\mathrm{H}(f)(x)_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\mathrm{x})$
  - 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

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

- 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

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

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

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

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

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

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

Quadratic approximation

- Let $w^* = \arg\min_w J(w)$ be optimum weights for minimal unregularized cost
- If the objective function is quadratic then

$$\hat{J}(\boldsymbol{\theta}) = J(w^*) + \frac{1}{2}(w - w^*)^T H(w - w^*)$$

  - H is the Hessian matrix of J with respect to w at w^*
  - No first order term as w^* is minimum
  - H is positive semidefinite
- Minimum of $\hat{J}$ occurs when $\nabla_w \hat{J}(w) = H(w - w^*) = 0$
- With weight decay we have

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

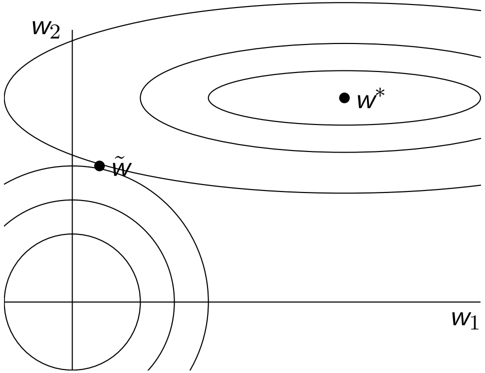
Quadratic approximation (contd)

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

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

  - 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: Geometrical interpretation



Linear regression

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

Linear regression

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

Linear regression

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

L^1 regularization

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

L^1 regularization

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

L^1 regularization

- Quadratic approximation of L^1 regularization objective function becomes $\hat{J}(\mathrm{w}; \mathrm{X}, \mathrm{y}) = J((\mathrm{w}^*; \mathrm{X}, \mathrm{y}) + \sum_i \left[\frac{1}{2} H_{i,i}(\mathrm{w}_i - \mathrm{w}_i^*)^2 + \alpha |w_i|\right]$
- So, analytical solution in each dimension will be $w_i = \mathrm{sign}(w_i^*) \max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}$
- 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

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

- 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

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

- If there exists a solution then

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

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

Constraint optimization (contd.)

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

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

- 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

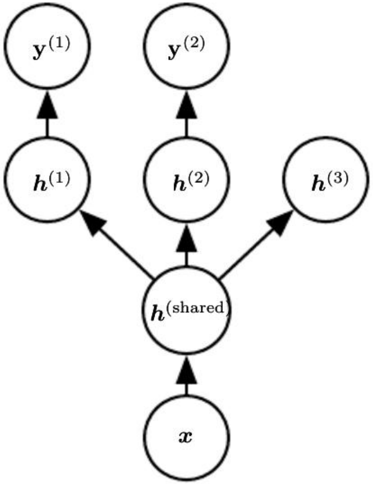


Image source: Deep Learning Book

Early stopping

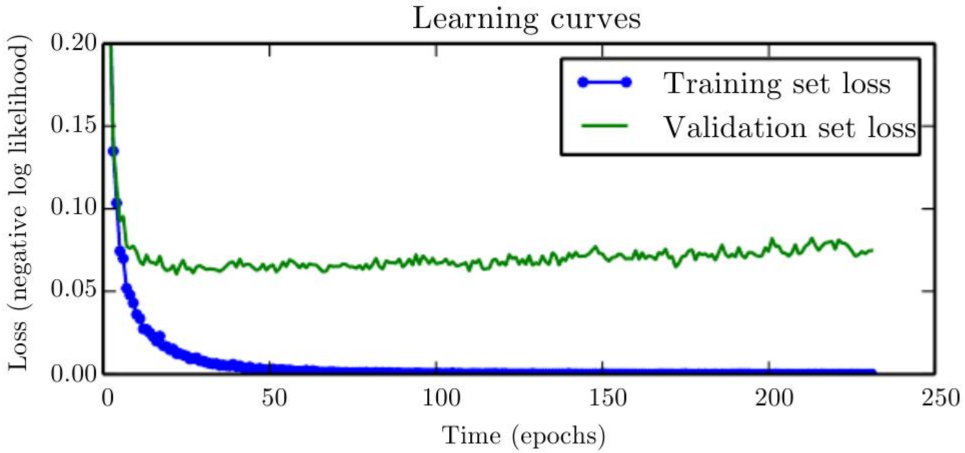


Image source: Deep Learning Book

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, $\hat{J}(\boldsymbol{\theta}) = J(\mathrm{w}^*) + \frac{1}{2}(\mathrm{w} - \mathrm{w}^*)\mathrm{H}(\mathrm{w} - \mathrm{w}^*)$ and $\nabla_{\mathrm{w}}\hat{J}(\mathrm{w}) = \mathrm{H}(\mathrm{w} - \mathrm{w}^*)$
- Assume $\mathrm{w}^{(0)} = 0$

Early stopping as regularizer

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

Early stopping as regularizer

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

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

Early stopping as regularizer

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

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

Early stopping as regularizer

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

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

Early stopping as regularizer

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

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

Early stopping as regularizer

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

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

Early stopping as regularizer

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

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

Early stopping as regularizer (contd)

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

$$Q^T\tilde{w} \quad = \quad (\mathbf{\Lambda} + \alpha \mathbf{I})^{-1}\mathbf{\Lambda} Q^T w^*$$

Early stopping as regularizer (contd)

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

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

Early stopping as regularizer (contd)

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

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

- Therefore we have, $(\mathbf{I} - \epsilon\boldsymbol{\Lambda})^\tau = (\boldsymbol{\Lambda} + \alpha\mathbf{I})^{-1}\alpha$
- 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 $\frac{1}{k}\sum_i \epsilon_i$
- Expected mean square error

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

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 $\frac{1}{k}\sum_i \epsilon_i$
- Expected mean square error

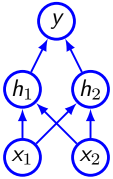
$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i \left(\epsilon_i^2 + \sum_{i \neq j} \epsilon_i \epsilon_j\right)\right] = \frac{v}{k} + \frac{k-1}{k}c$$

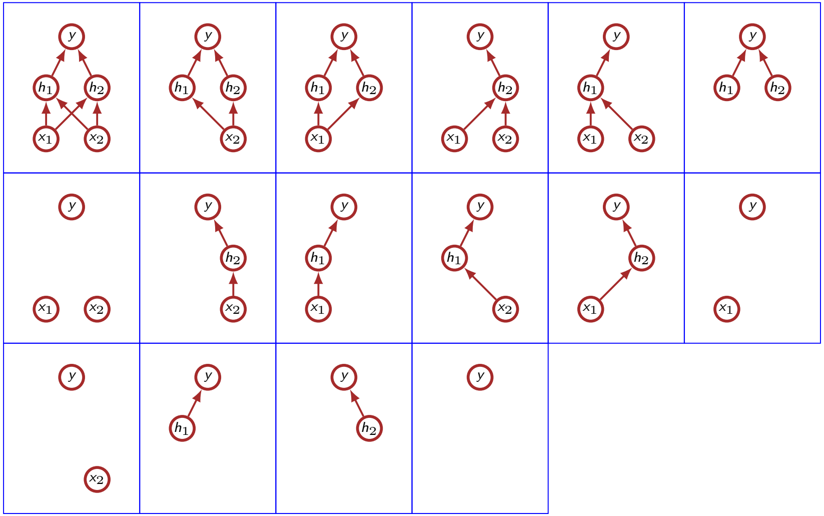
- If ϵ_i and ϵ_j are uncorrelated, ie. $c = 0$, then expected mse will be $\frac{v}{k}$ - Significant reduction in error
- If ϵ_i and ϵ_j are correlated, ie. $c = v$, then expected mse will be v - No change in error

Dropout

- It can be treated as a method of making bagging practical for ensembles of many large neural networks
  - Bagging is impractical with large number of models
  - Dropout is capable of handling exponentially many networks
- It trains the ensemble consiting of all subnetworks that can be formed by removing non-output units for the base network
- Removal of a node can be realized by multiplying it with 0, hence, binary mask is used
- Typically, dropout probability for input layer is low (~ 0.2). Hidden layer can have high probability (~ 0.5)
- Dropout is not used after training when making a prediction with the fit network.
- If a unit is retained with probability p during training, the outgoing weights of that unit are multiplied by p at test time

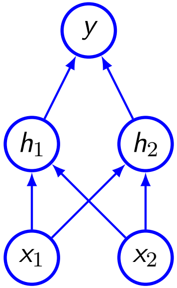
Dropout: sub-networks

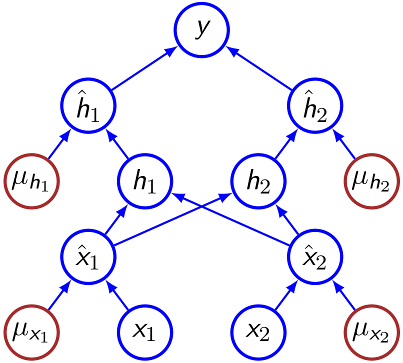




Dropout

- μ_u denotes the binary mask for node u





Adversarial training

- It is expected that outcome of an example to be constant in the close vicinity of the training data
- Small change in input can lead to misclassification because linearity with high coefficient

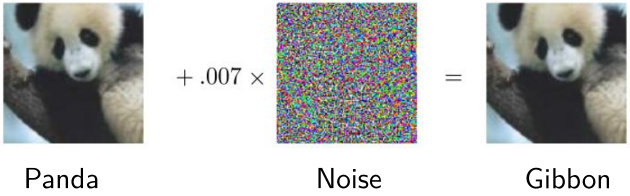


Image source: Deep Learning Book

Summary

- Goal of regularization techniques is to reduce generalization error. Large data sets help in generalization
- Increasing the number of units in hidden layer increases the model capacity. Increasing the depth helps in reducing the number of units in intermediate layers.
- Common approaches for regularization
  - Penalty based
  - Ensemble method
  - Introducing stochasticity to inputs and weights