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

Norm penalties

- **Most of the regularization approaches are based on limiting the capacity of the model**
- **Objective function becomes** $\tilde{J}(\boldsymbol{\theta}; X, y) = J(\boldsymbol{\theta}; X, y) + \alpha\Omega(\boldsymbol{\theta})$
  - 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 $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{w}\|_2^2$**
  - **Also known as ridge regression or Tikhonov regression**
- **Objective function $\tilde{J}(\mathbf{w}; \mathsf{X}, \mathsf{y}) = \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + J(\mathbf{w}; \mathsf{X}, \mathsf{y})$**

L^2 parameter regularization

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

L^2 parameter regularization

- **Weights are closer to origin as** $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{w}\|_2^2$
  - **Also known as ridge regression or Tikhonov regression**

- **Objective function** $\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + J(\mathbf{w}; \mathbf{X}, \mathbf{y})$

- **Gradient** $\nabla_\mathbf{w}\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha\mathbf{w} + \nabla_\mathbf{w}J(\mathbf{w}; \mathbf{X}, \mathbf{y})$

- **New weights**

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

L^2 parameter regularization

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

$$w = w - \epsilon(\alpha w + \nabla_w J(w; X, y)) = w(1 - \epsilon\alpha) - \epsilon\nabla_w J(w; X, y)$$

L^2 parameter regularization

- **Weights are closer to origin as $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|w\|_2^2$**
  - **Also known as ridge regression or Tikhonov regression**

- **Objective function $\tilde{J}(w; X, y) = \frac{\alpha}{2}w^\top w + J(w; X, y)$**

- **Gradient $\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y)$**

- **New weights**

  $$w \;=\; w - \epsilon(\alpha w + \nabla_w J(w; X, y)) \quad=\quad w(1 - \epsilon\alpha) - \epsilon\nabla_w J(w; X, y)$$

- **Assuming quadratic nature of curve in the neighborhood of**

  $$w^* = \arg\min_w J(w)$$

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

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** $J \in \mathbb{R}^{n \times m}$ **of** f **defined as** $J_{i,j} = \frac{\partial}{\partial x_j} f(x)_i$
- **Second derivative is also required sometime**

  - **For example,** $f : \mathbb{R}^n \to \mathbb{R}$, $\dfrac{\partial^2}{\partial x_i \partial x_j} f$
  - **If second derivative is 0, then there is no curvature**

- **Hessian matrix** $H(f)(x)_{ij} = \dfrac{\partial^2}{\partial x_i \partial x_j} f(x)$

  - **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**

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

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

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**

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

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

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

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 $\alpha \to \infty$**
  - H **is symmetric, therefore** $H = Q\Lambda Q^T$. **Now we have**

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

- **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
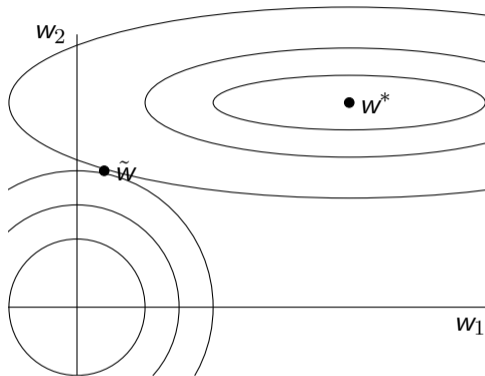
Linear regression

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

L^1 regularization

- Formally it is defined as $\Omega(\boldsymbol{\theta}) = \|w\|_1 = \sum_i |w_i|$

- Regularized objective function will be $\tilde{J}(w; X, y) = \alpha\|w\|_1 + J(w; X, y)$

L^1 regularization

- **Formally it is defined as** $\Omega(\boldsymbol{\theta}) = \|\mathbf{w}\|_1 = \sum_i |w_i|$

- **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 $\hat{J}(w; X, y) = J((w^*; X, y)) + \sum_i \left[\frac{1}{2} H_{i,i}(w_i - w_i^*)^2 + \alpha|w_i| \right]$

- So, analytical solution in each dimension will be $w_i = \text{sign}(w_i^*) \max\left\{ |w_i^*| - \frac{\alpha}{H_{i,i}}, 0 \right\}$

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

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

- **If there exists a solution then**
$$\min_x \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha} \geq 0} L(x, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \min_x f(x)$$

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

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

Learning curves

Image source: Deep Learning Book
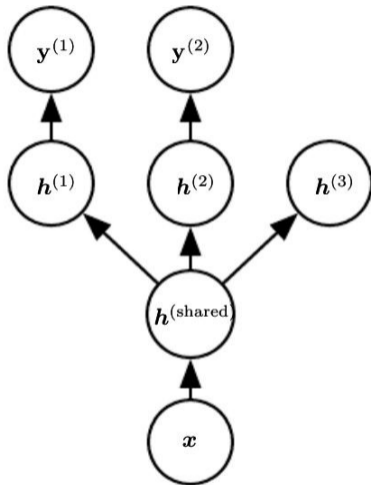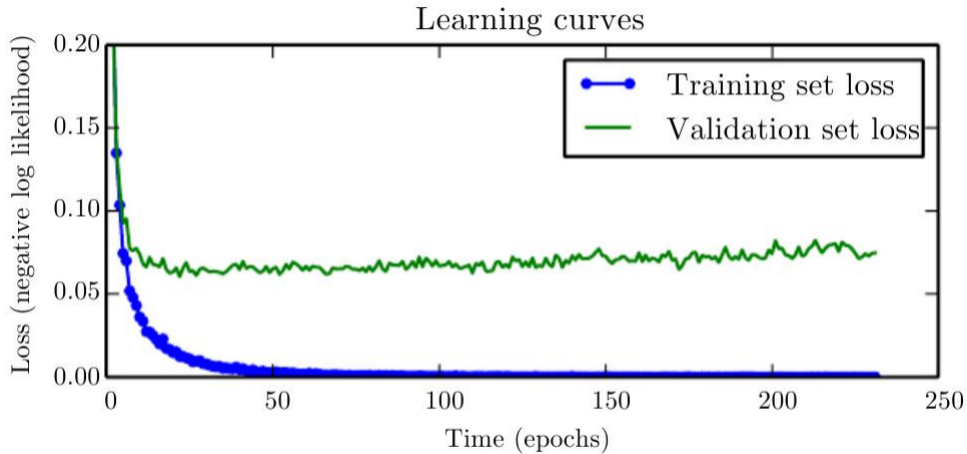
Early stopping approach

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

Early stopping as regularizer

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

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

Early stopping as regularizer

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

$$w^{(\tau)} = w^{(\tau-1)} - \epsilon\nabla_w \hat{J}(w^{(\tau-1)})$$
$$w^{(\tau)} = w^{(\tau-1)} - \epsilon H(w^{(\tau-1)} - w^*)$$

Early stopping as regularizer

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

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

Early stopping as regularizer

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

$$w^{(\tau)} = w^{(\tau-1)} - \epsilon\nabla_w\hat{J}(w^{(\tau-1)})$$
$$w^{(\tau)} = w^{(\tau-1)} - \epsilon H(w^{(\tau-1)} - w^*)$$
$$w^{(\tau)} - w^* = (I - \epsilon H)(w^{(\tau-1)} - w^*)$$
$$w^{(\tau)} - w^* = (I - \epsilon Q\Lambda Q^T)(w^{(\tau-1)} - w^*)$$

Early stopping as regularizer

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

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

$$w^{(\tau)} = w^{(\tau-1)} - \epsilon H(w^{(\tau-1)} - w^*)$$

$$w^{(\tau)} - w^* = (I - \epsilon H)(w^{(\tau-1)} - w^*)$$

$$w^{(\tau)} - w^* = (I - \epsilon Q\Lambda Q^T)(w^{(\tau-1)} - w^*)$$

$$Q^T(w^{(\tau)} - w^*) = (I - \epsilon\Lambda)Q^T(w^{(\tau-1)} - w^*)$$

Early stopping as regularizer

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

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

Early stopping as regularizer (contd)

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

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

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

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

- **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
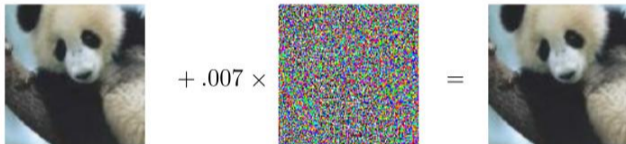
Dropout: sub-networks
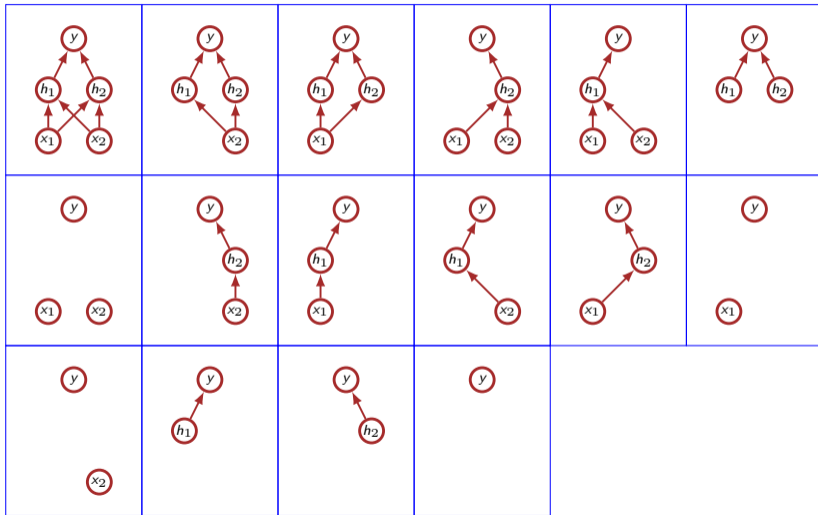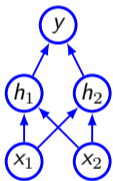
Dropout

- μ_u denotes the binary mask for node u
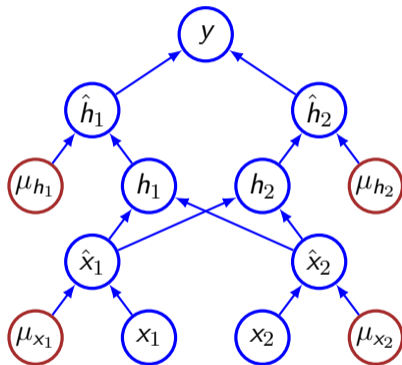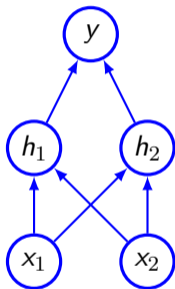
Adversarial training

- **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**

$$+ .007 \times \qquad =$$

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**