Learning From Data
Lecture 7: Model Selection & Regularization

Yang Li yangli@sz.tsinghua.edu.cn

April 12, 2024

Today's Lecture

Practical tools to improve machine learning performance:

- Bias and variance trade off
- Model selection and feature selection
- Regularization
  - Generic techniques
  - Neural network regularization tricks
- Midterm information

Empirical error & Generalization error

Consider a learning task, the **empirical (training) error** of hypothesis h is the expected loss over m training samples

$$\hat{\epsilon}_{0,1}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\} \quad \text{(classification, 0-1 loss)}$$

$$\hat{\epsilon}_{LS}(h) = \frac{1}{m}\sum_{i=1}^{m} ||h(x^{(i)}) - y^{(i)}||_2^2 \quad \text{(regression, least-square loss)}$$

Empirical error & Generalization error

Consider a learning task, the **empirical (training) error** of hypothesis h is the expected loss over m training samples

$$\hat{\epsilon}_{0,1}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\} \quad \text{(classification, 0-1 loss)}$$

$$\hat{\epsilon}_{LS}(h) = \frac{1}{m}\sum_{i=1}^{m} ||h(x^{(i)}) - y^{(i)}||_2^2 \quad \text{(regression, least-square loss)}$$

The **generalization (testing) error** of h is the expected error on examples not necessarily in the training set.

$x, y \sim P_{XY}$.

Empirical error & Generalization error

Consider a learning task, the **empirical (training) error** of hypothesis h is the expected loss over m training samples

$$\hat{\epsilon}_{0,1}(h) = \frac{1}{m} \sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\} \quad \text{(classification, 0-1 loss)}$$

$$\hat{\epsilon}_{LS}(h) = \frac{1}{m} \sum_{i=1}^{m} ||h(x^{(i)}) - y^{(i)}||_2^2 \quad \text{(regression, least-square loss)}$$

The **generalization (testing) error** of h is the expected error on examples not necessarily in the training set.

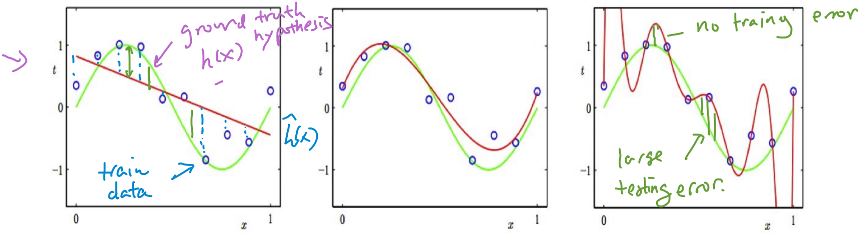
Goal of machine learning

- make training error small (optimization)
- make the gap between empirical and generalization error small

Overfit & Underfit

Underfit Both training error and testing error are large

Overfit Training error is small, testing error is large



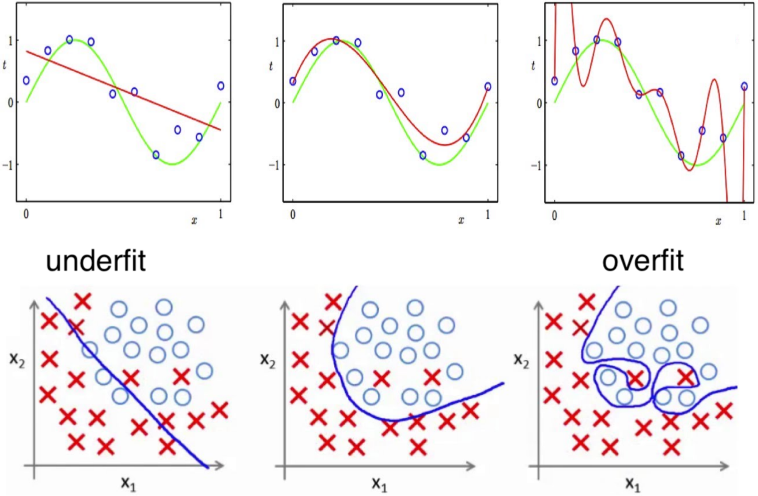
underfit

overfit

Overfit & Underfit

Underfit Both training error and testing error are large

Overfit Training error is small, testing error is large



Model capacity: the ability to fit a wide variety of functions

Model Capacity

Changing a model's **capacity** controls whether it is more likely to overfit or underfit

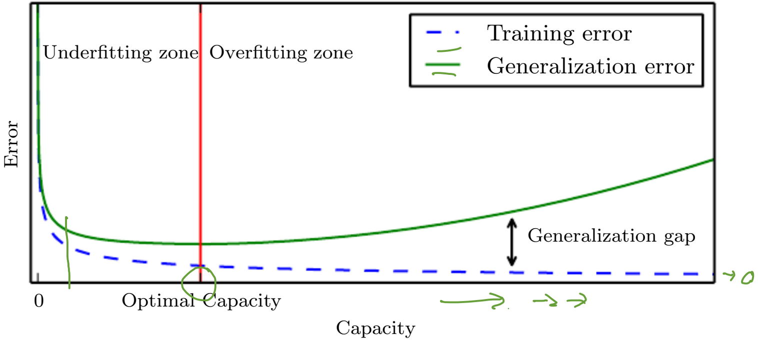
- Choose a model's hypothesis space: e.g. increase # of features (adding parameters) $h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$. $\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$.
- Find the best among a family of hypothesis functions

$$\{ h(x) \mid \theta_0, \theta_1, \theta_2 \in \mathbb{R}^n \}$$

Model Capacity

Changing a model's **capacity** controls whether it is more likely to overfit or underfit

- Choose a model's hypothesis space: e.g. increase # of features (adding parameters)
- Find the best among a family of hypothesis functions



How to formalize this idea?

Bias & Variance

Suppose data is generated by the following model:

true hypothesis

$$y = \underbrace{h(x)}_{} + \underbrace{\epsilon}_{\text{noise}}$$

with $\mathbb{E}[\epsilon] = 0, Var(\epsilon) = \sigma^2$ $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

- $h(x)$: true hypothesis function $\rightarrow$ *fixed value*
- D: training data $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ sampled from P_{XY}
- $\hat{h}(x; D)$: estimated hypothesis function based on D, sometimes written as $\hat{h}(x)$ for short $\rightarrow$ *random variable*

$\min_{\substack{h \in H \\ (x,y) \in D}} \ell(h(x), y)$

Bias & Variance

Bias of a model: The expected estimation error of $\hat{h}$ over all choices of training data D sampled from P_{XY},

true hypothesis function (label)

$$Bias(\hat{h}) = \mathbb{E}_D[\hat{h}(x) - h(x)] = \mathbb{E}_D[\hat{h}(x)] - h(x)$$

$\mathbb{E}_D[\hat{h}(x)] - \mathbb{E}_D(h(x))$

When we make wrong assumptions about the model, $\hat{h}$ will have large bias (underfit)

Variance of a model: How much $\hat{h}$ move around its mean

$Var(X) = \mathbb{E}[(X - \bar{X})^2]$

$= \mathbb{E}[X^2] - \mathbb{E}[X]^2$

$$Var(\hat{h}) = \mathbb{E}_D[(\hat{h}(x) - \mathbb{E}_D(\hat{h}(x))^2]$$
$$= \mathbb{E}_D[\hat{h}(x)^2] - \mathbb{E}_D[\hat{h}(x)]^2$$

average hypothesis function.

When the model overfits "spurious" patterns, it has large variance (overfit).

Bias - Variance Tradeoff

$\underset{x\sim P}{\mathbb{E}}\left(Var(\hat{h}(x)) + Bias(\hat{h}(x))^2 + \sigma^2\right)$

$= \underset{x}{\mathbb{E}}[Var(\hat{h}(x)) + Bias(\hat{h}(x))^2] + \underbrace{\underset{x}{\mathbb{E}}(\sigma^2)}_{\sigma^2}$

MSE Decomposition

(test)

We can decompose the expected error of MSE on a new sample (x,y):

$$MSE = \mathbb{E}_{D,\epsilon}[(\hat{h}(x) - y)^2] = Bias(\hat{h})^2 + Var(\hat{h}) + \sigma^2,$$

- σ^2 represents irreducible error
- in practice, increasing capacity tends to increase variance and decrease bias.

proof. $Bias(\hat{h}(x)) = \underset{D}{\mathbb{E}}[\hat{h}(x) - h(x)] = \underset{D}{\mathbb{E}}[\hat{h}(x)] - h(x)$ ✓

$\varepsilon \sim N(0, \sigma^2)$

$y = h(x) + \varepsilon$

$Var(\hat{h}(x)) = \underset{D}{\mathbb{E}}[\hat{h}(x)^2] - \underset{D}{\mathbb{E}}[\hat{h}(x)]^2$ (*)

$MSE = \underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)^2 + y^2 - 2\hat{h}(x)y] = \underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)^2] + \underset{D,\varepsilon}{\mathbb{E}}[y^2] - 2\underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)y]$

$\underset{D,\varepsilon}{\mathbb{E}}[y^2] = \underset{D,\varepsilon}{\mathbb{E}}[(h(x)+\varepsilon)^2] = \underset{D,\varepsilon}{\mathbb{E}}[h(x)^2] + \underset{D,\varepsilon}{\mathbb{E}}[\varepsilon^2] + 2\underset{D,\varepsilon}{\mathbb{E}}[h(x)\varepsilon]$

$= h(x)^2 + \sigma^2 + 2\underset{D,\varepsilon}{\mathbb{E}}[h(x)]\cancel{\underset{D,\varepsilon}{\mathbb{E}}[\varepsilon]} = h(x)^2 + \sigma^2$

0 $h(x)$ and ε are independent

$\underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)y] = \underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)h(x)] + \underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)\varepsilon] = h(x)\underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)] + \underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)]\cancel{\underset{\varepsilon}{\mathbb{E}}[\varepsilon]}^{\,0} = h(x)\underset{D}{\mathbb{E}}[\hat{h}(x)]$

$\underset{D,\varepsilon}{\mathbb{E}}[\hat{h}(x)^2] = Var(\hat{h}(x)) + \underset{D}{\mathbb{E}}(\hat{h}(x))^2$. by (*).

Therefore, $MSE = Var(\hat{h}(x)) + \mathbb{E}(\hat{h}(x))^2 + h(x)^2 + \sigma^2 - 2h(x)\underset{D}{\mathbb{E}}(\hat{h}(x))$

Bias - Variance Tradeoff

$= Var(\hat{h}(x)) + (\underbrace{\mathbb{E}_D[\hat{h}(x)] - h(x)}) ^2 + \sigma^2$

$= Var(\hat{h}(x)) + Bias(\hat{h}(x))^2 + \sigma^2$

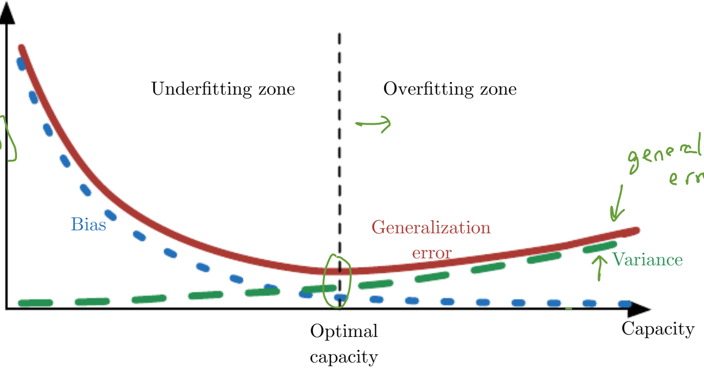
MSE Decomposition

We can decompose the expected error of MSE on a new sample (x,y):

$$MSE = \mathbb{E}_{D,\epsilon}[(\hat{h}(x) - y)^2] = Bias(\hat{h})^2 + Var(\hat{h}) + \sigma^2,$$

- σ^2 represents irreducible error
- in practice, increasing capacity tends to increase variance and decrease bias.

$\mathbb{E}_D[\hat{h}(x) - h(x)] =$



generalization error.

Cross validation

Model selection

Model Selection

For a given task, how do we select which model to use?

- Different learning models
  - e.g. SVM vs. logistic regression for binary classification
- Same learning models with different **hyperparameters**
  - e.g. # of clusters in k-means clustering

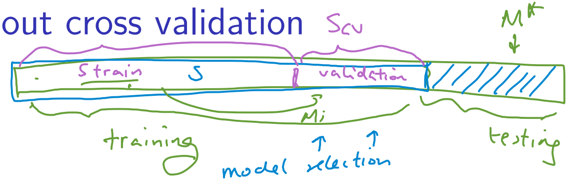
Model Selection

For a given task, how do we select which model to use?

- Different learning models
  - e.g. SVM vs. logistic regression for binary classification
- Same learning models with different **hyperparameters**
  - e.g. # of clusters in k-means clustering

Cross validation is a class of methods for selecting models using a *validation set*.

Hold-out cross validation



Given training set S and candidate models $M_1, \ldots, M_n$: $\to M^*$

1. Randomly split S into S_{train} and S_{cv} (e.g. 70% S_{train})
2. Training each M_i on S_{train},
3. Select the model with smallest empirical error on S_{cv}

Hold-out cross validation

Given training set S and candidate models $M_1,\ldots,M_n$:

1. Randomly split S into S_{train} and S_{cv} (e.g. 70% S_{train})
2. Training each M_i on S_{train},
3. Select the model with smallest empirical error on S_{cv}

Disavantages of hold-out cross validation

- "wastes" about 30% data S_{cv}
- chances of an unfortunate split

K-Fold Cross Validation

Goal: ensure each sample is equally likely to be selected for validation.

1. Randomly split S into k disjoint subsets $S_1, \ldots, S_k$ of m/k training examples (e.g. $k = 5$)

K-Fold Cross Validation

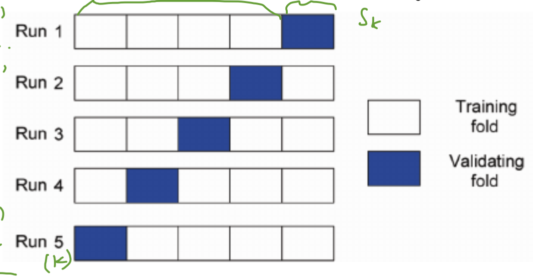
$M_1, \ldots, M_n$

Goal: ensure each sample is equally likely to be selected for validation.

1. Randomly split S into k disjoint subsets $S_1, \ldots, S_k$ of m/k training examples (e.g. $k = 5$)
2. For $j = 1 \ldots k$:

   Train each model on $S \setminus S_j$, then validate on S_j,

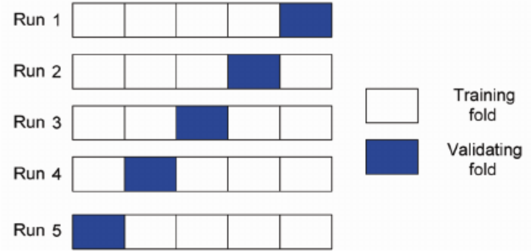
$Acc_1^{(1)}$	$Acc_2^{(1)}$	$\cdots$	$Acc_n^{(1)}$
$Acc_1^{(2)}$	$\cdots$		$Acc_n^{(2)}$
$\vdots$			
$Acc_1^{(k)}$	$\cdots$		$Acc_n^{(k)}$
$\overline{Acc}(M_1)$	$\overline{Acc}(M_2)$		



K-Fold Cross Validation

Goal: ensure each sample is equally likely to be selected for validation.

1. Randomly split S into k disjoint subsets $S_1, \ldots, S_k$ of m/k training examples (e.g. $k = 5$)
2. For $j = 1 \ldots k$:
   Train each model on $S \setminus S_j$, then validate on S_j,



3. Select the model with the smallest **average** empirical error among all k trails.

Leave-One-Out Cross Validation

$$\underbrace{x_1}_{S_1}, \quad \dots \quad \dots, \underbrace{\boxed{x_m}}_{S_{k=m}}$$

A special case of k-fold cross validation, when $\underline{k} = m$.

1. For each training example x_i
   Train each model on $\underline{S \backslash \{x_i\}}$, then evaluate on x_i,
2. Select the model with the smallest average empirical error among all m trails.

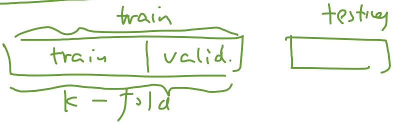
Often used when training data is scarce.

Other Cross Validation Methods

- Random subsampling
- Bootstrapping: sample with replacement from training examples (used for small training set)
- Information criteria based methods: e.g. Bayesian information criterion (BIC), Akaike information criterion (AIC)

Other Cross Validation Methods

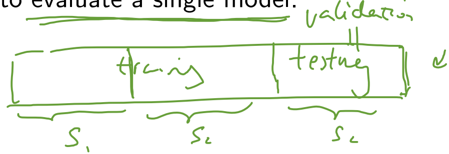
data leakage: (model selection)



- Random subsampling
- Bootstrapping: sample with replacement from training examples (used for small training set)
- Information criteria based methods: e.g. Bayesian information criterion (BIC), Akaike information criterion (AIC)

Cross validation can also be used to evaluate a single model.

$K=3$.



report $\underset{i=1..k}{\mathrm{Avg}}(Acc_i)$ no picking involved!

Regularization

Regularization

Regularization is any modification we make to a learning algorithm to reduce its generalization error, but not the training error

Regularization

Regularization is any modification we make to a learning algorithm to reduce its generalization error, but not the training error

Common regularization techniques:

- Penalize parameter size
  e.g. linear regression with **weight decay**:

$$J(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) + \lambda||\theta||_2^2$$

Regularization

Regularization is any modification we make to a learning algorithm to reduce its generalization error, but not the training error

Common regularization techniques:

- Penalize parameter size
  e.g. linear regression with **weight decay**:

$$J(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) + \lambda||\theta||_2^2$$

- Use prior probability: max-a-posteriori estimation

Parameter Norm Penalty

Adding a regularization term to the loss (error) function:

$$\tilde{J}(X, Y; \theta) = \underbrace{J(X, Y; \theta)}_{\text{data-dependent loss}} + \lambda \underbrace{\Omega(\theta)}_{\text{regularizer}}$$

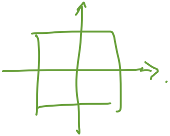
where

$$\Omega(\theta) = \frac{1}{2} \sum_{j=1}^{n} |\theta_j|^{\boxed{q}} = \frac{1}{2} ||\theta||_q^q$$

Parameter Norm Penalty

Adding a regularization term to the loss (error) function:

$$\tilde{J}(X,Y;\theta) = \underbrace{J(X,Y;\theta)}_{\text{data-dependent loss}} + \lambda \underbrace{\Omega(\theta)}_{\text{regularizer}}$$



where

$$\Omega(\theta) = \frac{1}{2}\sum_{j=1}^{n} |\theta_j|^q = \frac{1}{2}||\theta||_q^q$$

$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

levelset $\Omega(\theta)=1$

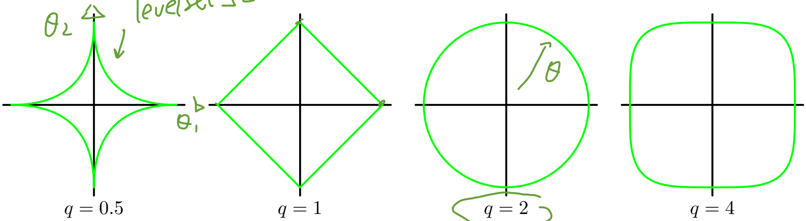


Figure: Contours of the regularizer ($||\theta||^q = 1$) for different q

L2 parameter penalty

When $q = 2$, it's also known as **Tokhonov regularization** or **ridge regression**

$$\tilde{J}(X, Y; \theta) = J(X, Y; \theta) + \frac{\lambda}{2}\theta^T\theta$$

L2 parameter penalty

When $q = 2$, it's also known as **Tokhonov regularization** or **ridge regression**

$$\tilde{J}(X, Y; \theta) = J(X, Y; \theta) + \frac{\lambda}{2}\theta^T\theta$$

Gradient descent update:

weight decay

$$\nabla_\theta \left(J(X,Y;\theta) + \frac{\lambda}{2}\theta^T\theta \right)$$

$$\nabla_\theta J(X,Y;\theta) + \frac{\lambda}{2}\underbrace{\nabla_\theta \theta^T\theta}_{2\theta}$$

$$\begin{aligned} \theta &\leftarrow \theta - \alpha\nabla_\theta\tilde{J}(X, Y; \theta) \\ &= \theta - \alpha(\nabla_\theta J(X, Y; \theta) + \lambda\theta) \\ &= \underbrace{(1 - \alpha\lambda)}_{<1}\theta - \alpha\nabla_\theta J(X, Y; \theta) \end{aligned}$$

L2 penalty multiplicatively shrinks parameter θ by a constant

L2 parameter penalty

When $q = 2$, it's also known as **Tokhonov regularization** or **ridge regression**

$$\tilde{J}(X, Y; \theta) = J(X, Y; \theta) + \frac{\lambda}{2}\theta^T\theta$$

Gradient descent update:

$$\begin{aligned}
\theta \leftarrow & \ \theta - \alpha \nabla_\theta \tilde{J}(X, Y; \theta) \\
& = \theta - \alpha(\nabla_\theta J(X, Y; \theta) + \lambda\theta) \\
& = (1 - \alpha\lambda)\theta - \alpha \nabla_\theta J(X, Y; \theta)
\end{aligned}$$

L2 penalty multiplicatively shrinks parameter θ by a constant

Example: regularized least square

When $J(X, Y; \theta) = \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$ (ordinary least squares),
$\tilde{\theta}_{OLS} = (X^T X + \lambda I)^{-1}(X^T Y)$

L1 parameter penalty

When $q = 1$, $\Omega(\theta) = \frac{1}{2}\sum_{j=1}^{n} |\theta_j|$ is also known as **LASSO regression**.

- If λ is sufficiently large, some coefficients θ_j are driven to 0.
- It will lead to a *sparse* model

L1 parameter penalty

When $q = 1$, $\Omega(\theta) = \frac{1}{2}\sum_{j=1}^{n}|\theta_j|$ is also known as **LASSO regression**.

- If λ is sufficiently large, some coefficients θ_j are driven to 0.
- It will lead to a *sparse* model

Proposition 1

θ^*, λ. (1).

Solving $\min_\theta \tilde{J}(X, Y; \theta) = J(X, Y; \theta) + \frac{\lambda}{2}\sum_{j=1}^{n}|\theta_j|^q$ *is equivalent to*

$\Updownarrow$

$$\begin{cases} \min_\theta J(X, Y; \theta) \\ s.t. \sum_{j=1}^{n}|\theta_j|^q \leq \eta \end{cases}$$

(2) θ, η

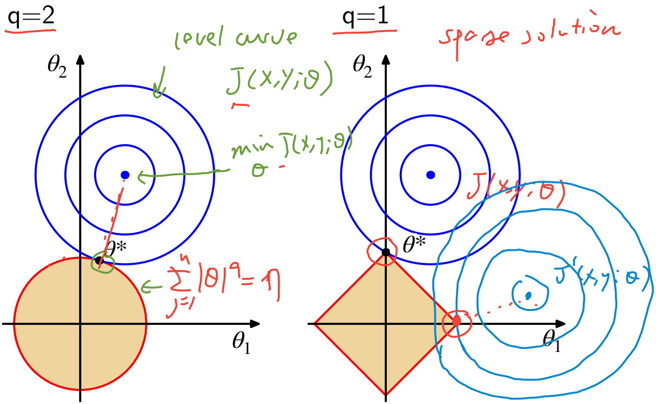
for some constant $\eta > 0$ $(\star)$*. Furthermore,* $\boxed{\eta} = \sum_{j=1}^{n}|\theta_j^*(\lambda)|^q$ *where* $\theta^*(\lambda) = \text{argmin}_\theta \tilde{J}(X, Y; \theta, \lambda)$

- $(\star)$ assumes constraints are satisfiable (e.g. with slater's condition)
- Choosing λ is equivalent to choosing η and vice versa
- Smaller $\lambda \to$ larger constraint region

larger η

L1 vs L2 parameter penalty

Figure: Contour plot of unregularized error $J(X, Y; \theta)$ and the constraint region $\sum_{j=1}^{n} |\theta|^q \leq \eta$



The lasso (l1 regularizer) gives a sparse solution with $\theta_1^* = 0$.

Bayesian Statistics

Maximum likelihood estimation: θ is an unknown constant

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$$

Bayesian view: θ is a random variable

prior distribution

$$\theta \sim p(\theta)$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

Given training set $S = \{x^{(i)}, y^{(i)}\}$, posterior distribution of θ

$$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)}$$

Fully Bayesian statistics

parameter

data

$$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)} = \frac{\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)p(\theta)}{\int_\theta (\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)p(\theta))d\theta}$$

To predict the label for new sample x, compute the posterior distribution over training set S:

MLE

$p(y|x;\theta)$

$$p(y|x,S) = \int_\theta p(y|x,\theta)p(\theta|S)d\theta$$

The label is

$$\mathbb{E}[y|x,S] = \int_y y\ p(y|x,S)dy$$

Fully bayesian estimate of θ is difficult to compute, has no close-form solution.

Bayesian Statistics

Posterior distribution on class label y using $p(\theta|S)$

$$p(y|x, S) = \int_{\theta} p(y|x, \theta) p(\theta|S) d\theta$$

Bayesian Statistics

Posterior distribution on class label y using $p(\theta|S)$

$$p(y|x,S) = \int_{\theta} p(y|x,\theta)p(\theta|S)d\theta$$

We can approximate $p(y|x,\theta)$ as follows:

MAP approximation

The **MAP (maximum a posteriori) estimate** of θ is

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)p(\theta)$$

Bayesian Statistics

Posterior distribution on class label y using $p(\theta|S)$

$$p(y|x, S) = \int_\theta p(y|x, \theta)p(\theta|S)d\theta$$

We can approximate $p(y|x, \theta)$ as follows:

MAP approximation

The **MAP (maximum a posteriori) estimate** of θ is

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta)\boxed{p(\theta)}$$

$p(y^{(i)}|x^{(i)}, \theta)$ is not the same as $p(y^{(i)}|x^{(i)}; \theta)$

MAP estimation and regularized least square

Recall ordinary least square is equivalent to maximum likelihood estimation when $p(y^{(i)}|x^{(i)}) \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$:

$$\begin{aligned}\theta_{MLE} &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} p(y^i|x^i;\theta) \\ &= (X^T X)^{-1} X^T Y = \theta_{OLS}\end{aligned}$$

MAP estimation and regularized least square

Recall ordinary least square is equivalent to maximum likelihood estimation when $p(y^{(i)}|x^{(i)}) \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$:

$$\begin{aligned}\theta_{MLE} &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} p(y^j|x^i;\theta) \\ &= (X^TX)^{-1}X^TY = \theta_{OLS}\end{aligned}$$

The MAP estimation when $\theta \sim N(0, \tau^2 I)$ is

$$\begin{bmatrix}\tau^2 & & \\ & \tau^2 & \\ & & \ddots \ \tau^2\end{bmatrix}$$

$$\begin{aligned}\theta_{MAP} &= \underset{\theta}{\operatorname{argmax}} \left(\prod_{i=1}^{m} p(y^j|x^i;\theta)\right) p(\theta) \\ &= \underset{\theta}{\operatorname{argmin}} \left(\frac{\sigma^2}{\tau^2}\theta^T\theta + (Y - X\theta)^T(Y - X\theta)\right) \\ &= (X^TX + \underbrace{\frac{\sigma^2}{\tau}}_{\lambda} I)^{-1}X^TY = \tilde{\theta}_{OLS} \text{ when } \boxed{\lambda} = \frac{\sigma^2}{\tau}\end{aligned}$$

variance

$P(y|x)$

prior variance

Discussion on MAP Estimation

General remarks on MAP:

- When θ is uniform, $\theta_{MAP} = \theta_{MLE}$
- A common choice for $p(\theta)$ is $\theta \sim \mathcal{N}(0, \tau^2 I)$, and θ_{MAP} corresponds to weight decay (L2-regularization)
- When θ is an isotropic Laplace distribution, θ_{MAP} corresponds to LASSO (L1-regularization).
- θ_{MAP} often have smaller norm than θ_{MLE}

Regularization for neural networks

Common regularization techniques:

- Data augmentation
- Parameter sharing
- Drop out

Data augmentation

Create fake data and add it to the training set. (Useful in certain tasks such as object classification.)

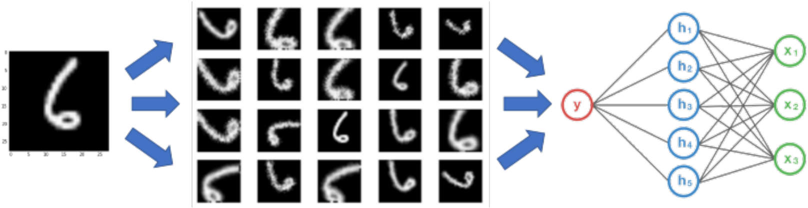


Figure: Generate fake digits via geometric transformation, e.g. scale, rotation etc

Figure: Generate images of different styles using GAN

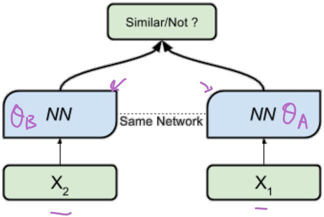
Shorten et. al. A survey on Image Data Augmentation for Deep Learning, 2019

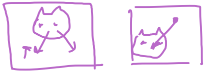
Parameter Sharing

Force sets of parameters to be equal based on prior knowledge. $y_1 \overset{?}{=} y_2$

Siamese Network

- Given input X, learns a discriminative feature $f(X)$
- For every pair of samples (X_1, X_2) in the same class, minimize their distance in feature space $||f(X_1) - f(X_2)||^2$





Convolutional Neural Network (CNN)

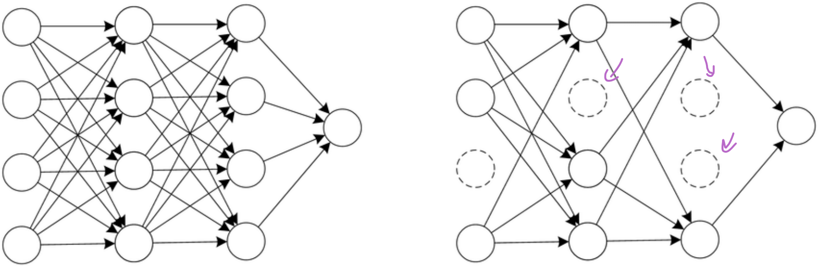
- Image features should be invariant to translation
- CNN shares parameters across multiple image locations.

Soft parameter sharing: add a norm penalty between sets of parameters:

$$\Omega(\theta^A, \theta^B) = ||\theta^A - \theta^B||_2^2$$

Drop Out

- Randomly remove a non-output unit from network by multiplying its output by zero (with probability p)
- In each mini-batch, randomly sample binary masks to apply to all inputs and hidden units
- Dropout trains an ensemble of different sub-networks to prevent the "co-adaptation" of neurons



(a) Standard Neural Network (b) Network after Dropout

Midterm Information

- Time: Next Friday, April 19, 10:00am (Arrive at 9:50am)
- Location: A307
- What to bring: One double-sided A4 notesheet
- Covers everything up to today (neural networks and model selection will only be short questions.)
- Midterm review session @ C401, April 14th 9:50am
- No late submission for WA2

13th

My office hour will be moved to tomorrow afternoon 2pm

Next lecture: learning theory

How to quantify generalization error?

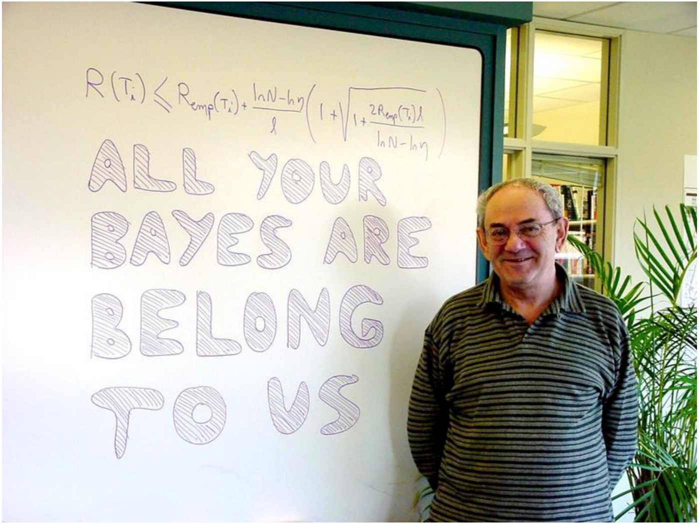


Figure: Prof. Vladimir Vapnik in front of his famous theorem