EE488 Special Topics in EE
<Deep Learning and AlphaGo>

Sae-Young Chung

Lecture 5

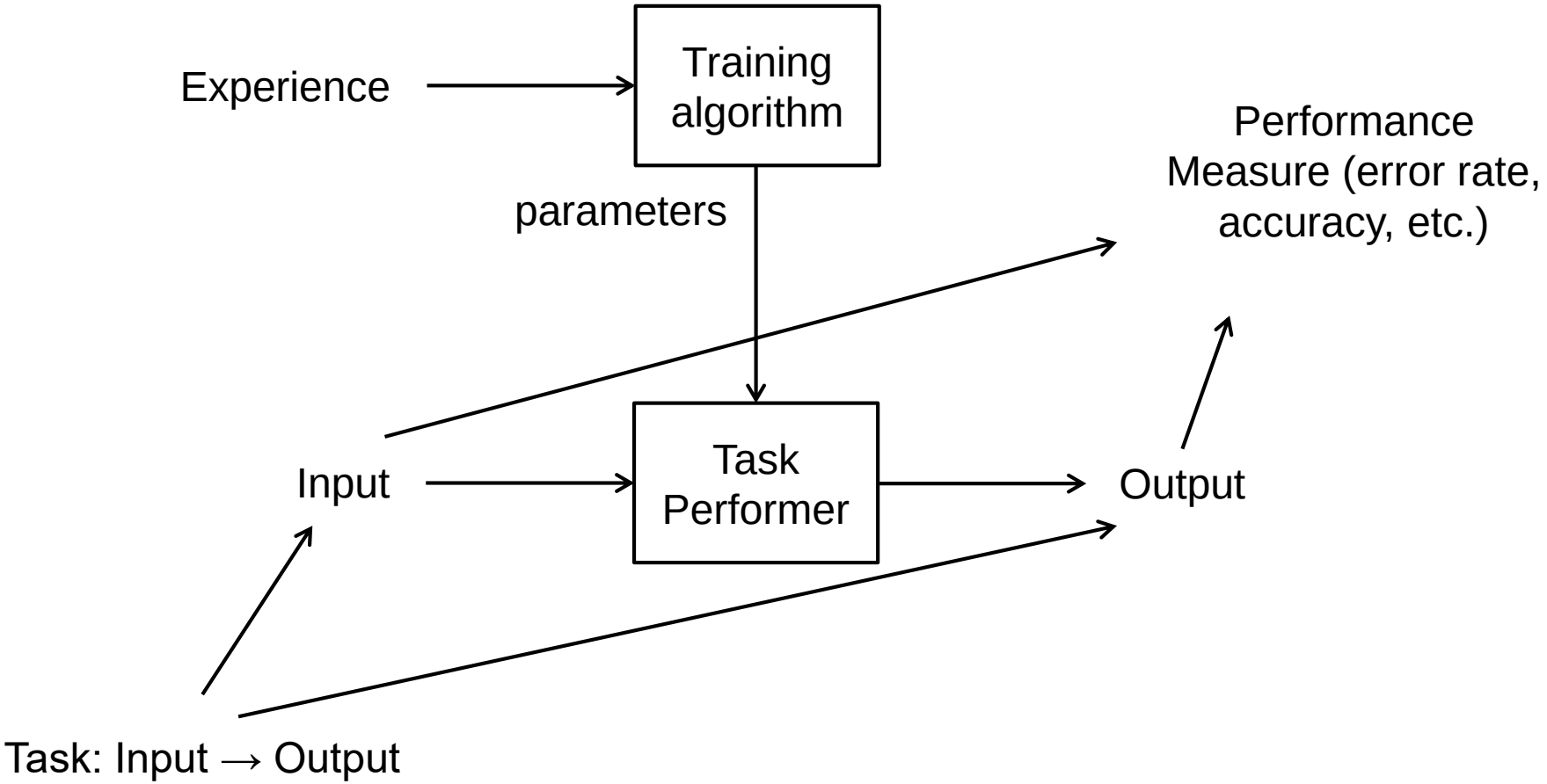
September 18, 2017

KAIST

Chap. 5 Machine Learning

- Machine learning: “A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.” – Tom Mitchell (1997)
- Of course, not limited to software, e.g., hardware-based deep neural network
  - Energy efficiency can improve a lot
  - E.g., 10,000x for IBM’s TrueNorth neuromorphic chip)
- Recommended textbooks for machine learning
  - Christopher Bishop, Pattern Recognition and Machine Learning
  - Kevin Murphy, Machine Learning A Probabilistic Perspective

Machine Learning

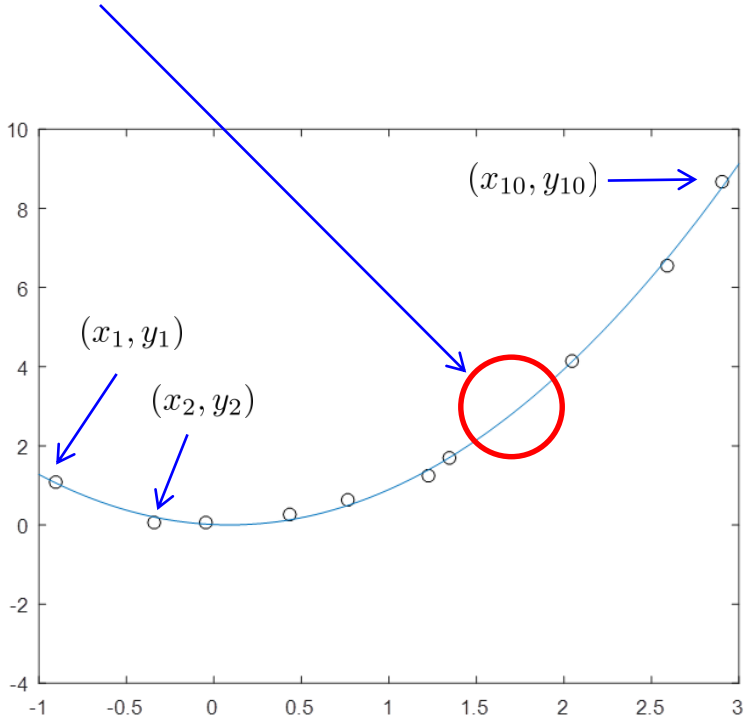


Machine Learning Tasks

- Classification: $\mathbb{R}^n \to \{1, \ldots, k\}$
- Regression: $\mathbb{R}^n \to \mathbb{R}$, $\mathbb{R}^n \to \mathbb{R}^k$
- Denoising: $\mathbb{R}^n \to \mathbb{R}^n$
- Compression & decompression: $\mathbb{R}^n \to \mathbb{R}^k \to \mathbb{R}^n$, $\mathbb{R}^n \to \{1, \ldots, k\} \to \mathbb{R}^n$
- Detection (anomaly detection, ...)
- Estimation (density estimation, ...)
- Transcription, translation, synthesis
- Reinforcement learning
- ...

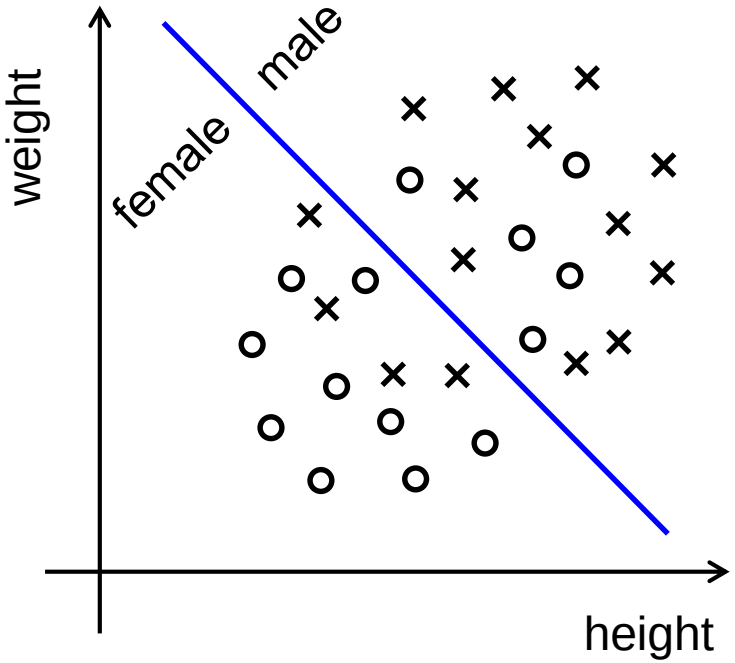
Example) Regression

- Curve fitting using (polynomial) regression
  - Requires a model (polynomial, exponential, etc.)
  - Parameters of the model (coefficients of the polynomial) are found automatically from data (experience) via optimization
  - ***Can predict unseen samples!!! (generalization from experience)***



Example) Classification

- Another example) Binary classification using a linear classifier



Linear Regression

- Regression: a process of estimating the relationships among multiple variables
- We will focus on the relationships between $\mathbf{x} \in \mathbb{R}^n$, $n \geq 1$, containing one or more independent variables and a dependent variable $y \in \mathbb{R}$
- Linear regression: use a linear function to model the relationship
  - $f : \mathbb{R}^n \to \mathbb{R}$, $\hat{y} = f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ (linear)
  - $\mathbf{w} \in \mathbb{R}^n$: parameters of the model
- Training set is used to train $\mathbf{w}$: $(\mathbf{X}^{(\text{train})}, \mathbf{y}^{(\text{train})})$
  - m': number of examples in the training set
  - $\mathbf{X}^{(\text{train})} \in \mathbb{R}^{m' \times n}$: set of input vectors
  - $\mathbf{y}^{(\text{train})} \in \mathbb{R}^{m' \times 1}$: set of outputs
- Test set: $(\mathbf{X}^{(\text{test})}, \mathbf{y}^{(\text{test})})$
  - m: number of examples in the test set
  - $\mathbf{X}^{(\text{test})} \in \mathbb{R}^{m \times n}$: set of input vectors
  - $\mathbf{y}^{(\text{test})} \in \mathbb{R}^{m \times 1}$: set of regression targets

Linear Regression

- $\hat{\mathbf{y}}^{(\text{test})} = \mathbf{X}^{(\text{test})}\mathbf{w}$: Predictions of the model on the test set
- Performance measure on the test set using the mean squared error (MSE)

$$\begin{aligned}\text{MSE}_{\text{test}} &= \frac{1}{m}\left\|\hat{\mathbf{y}}^{(\text{test})} - \mathbf{y}^{(\text{test})}\right\|^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}_i^{(\text{test})} - y_i^{(\text{test})}\right)^2\end{aligned}$$

- Goal: minimize the MSE on the test set
- Problem: test set is not available during training
  - If an ML algorithm has access to the test set during training, it can simply memorize the set and use the knowledge to achieve zero error during test. (cheating!)
- One solution: minimize the MSE on the training set instead, i.e., minimize the following

$$\text{MSE}_{\text{train}} = \frac{1}{m}\left\|\hat{\mathbf{y}}^{(\text{train})} - \mathbf{y}^{(\text{train})}\right\|^2$$

Linear Regression

$$
\begin{aligned}
\nabla_{\mathbf{w}} \mathrm{MSE}_{\mathrm{train}} &= \nabla_{\mathbf{w}} \frac{1}{m'} \left\| \hat{\mathbf{y}}^{(\mathrm{train})} - \mathbf{y}^{(\mathrm{train})} \right\|^2 \\
&= \nabla_{\mathbf{w}} \frac{1}{m'} \left\| \mathbf{X}^{(\mathrm{train})} \mathbf{w} - \mathbf{y}^{(\mathrm{train})} \right\|^2 \\
&= \nabla_{\mathbf{w}} \frac{1}{m'} \left(\mathbf{X}^{(\mathrm{train})} \mathbf{w} - \mathbf{y}^{(\mathrm{train})} \right)^T \left(\mathbf{X}^{(\mathrm{train})} \mathbf{w} - \mathbf{y}^{(\mathrm{train})} \right) \\
&= \nabla_{\mathbf{w}} \frac{1}{m'} \left(\mathbf{w}^T \mathbf{X}^{(\mathrm{train})T} \mathbf{X}^{(\mathrm{train})} \mathbf{w} - 2\mathbf{w}^T \mathbf{X}^{(\mathrm{train})T} \mathbf{y}^{(\mathrm{train})} + \mathbf{y}^{(\mathrm{train})T} \mathbf{y}^{(\mathrm{train})} \right) \\
&= \frac{1}{m'} \left(2\mathbf{X}^{(\mathrm{train})T} \mathbf{X}^{(\mathrm{train})} \mathbf{w} - 2\mathbf{X}^{(\mathrm{train})T} \mathbf{y}^{(\mathrm{train})} \right)
\end{aligned}
$$

$$
\nabla_{\mathbf{w}} \mathrm{MSE}_{\mathrm{train}} = 0 \quad \Rightarrow \quad \mathbf{w}^* = \left(\mathbf{X}^{(\mathrm{train})T} \mathbf{X}^{(\mathrm{train})} \right)^{-1} \mathbf{X}^{(\mathrm{train})T} \mathbf{y}^{(\mathrm{train})}
$$

Polynomial Regression

- Intercept (bias) term: $\hat{y} = \mathbf{w}^T \mathbf{x} + b$
- Polynomial regression

$$\hat{y} = w_0 + w_1 x + \ldots + w_{n-1} x^{n-1} + w_n x^n$$

  - Polynomial regression can be considered as a linear regression with input given as

$$(1, x, \ldots, x^{n-1}, x^n)$$

  - The degree n of the polynomial determines the model capacity
- Model capacity: measures expressive power of the model (e.g., ability to fit various functions)
  - Too low: underfitting problem may occur
  - Too high: overfitting problem may occur
  - For deep learning, model capacity is determined by many factors such as the number of neurons, the number of connections, the number of layers, connection structure, etc.

Recap – Least Square Solution

- Training examples

$$(x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_m, y_m)$$

- Model for polynomial regression

$$y_i = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \ldots + \theta_n x_i^n + z_i, \quad i = 1, \ldots, m$$

- Matrix form: $\mathbf{y} = X\theta + \mathbf{z}$, i.e.,

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{pmatrix}$$

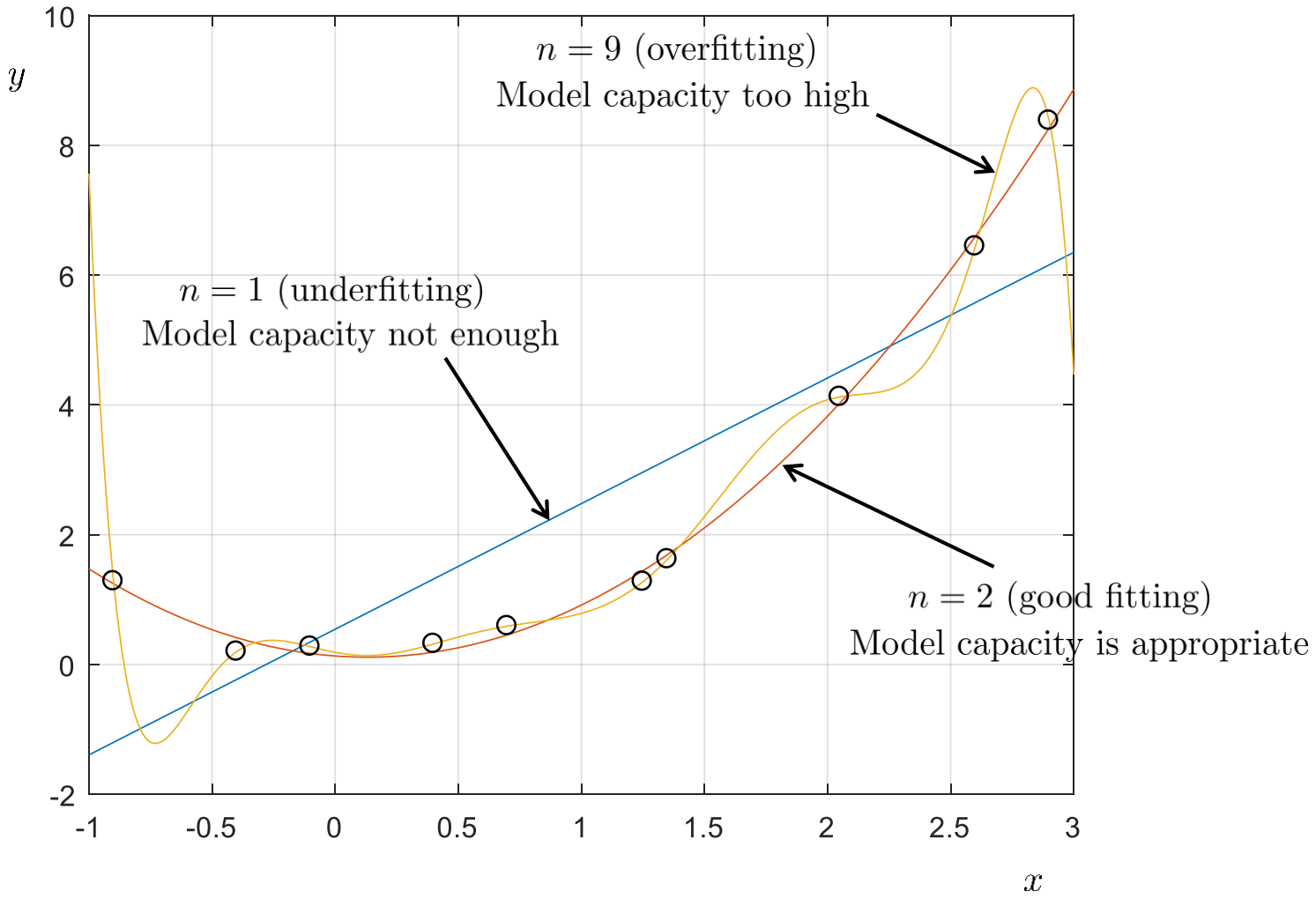
  - X is called a Vandermonde matrix

- Goal: to find the least square solution θ that minimizes $\|\mathbf{z}\|^2$ assuming $n < m$

- Answer

$$\theta^* = (X^T X)^{-1} X^T \mathbf{y}$$

Underfitting & Overfitting



Empirical Risk & Generalization Error

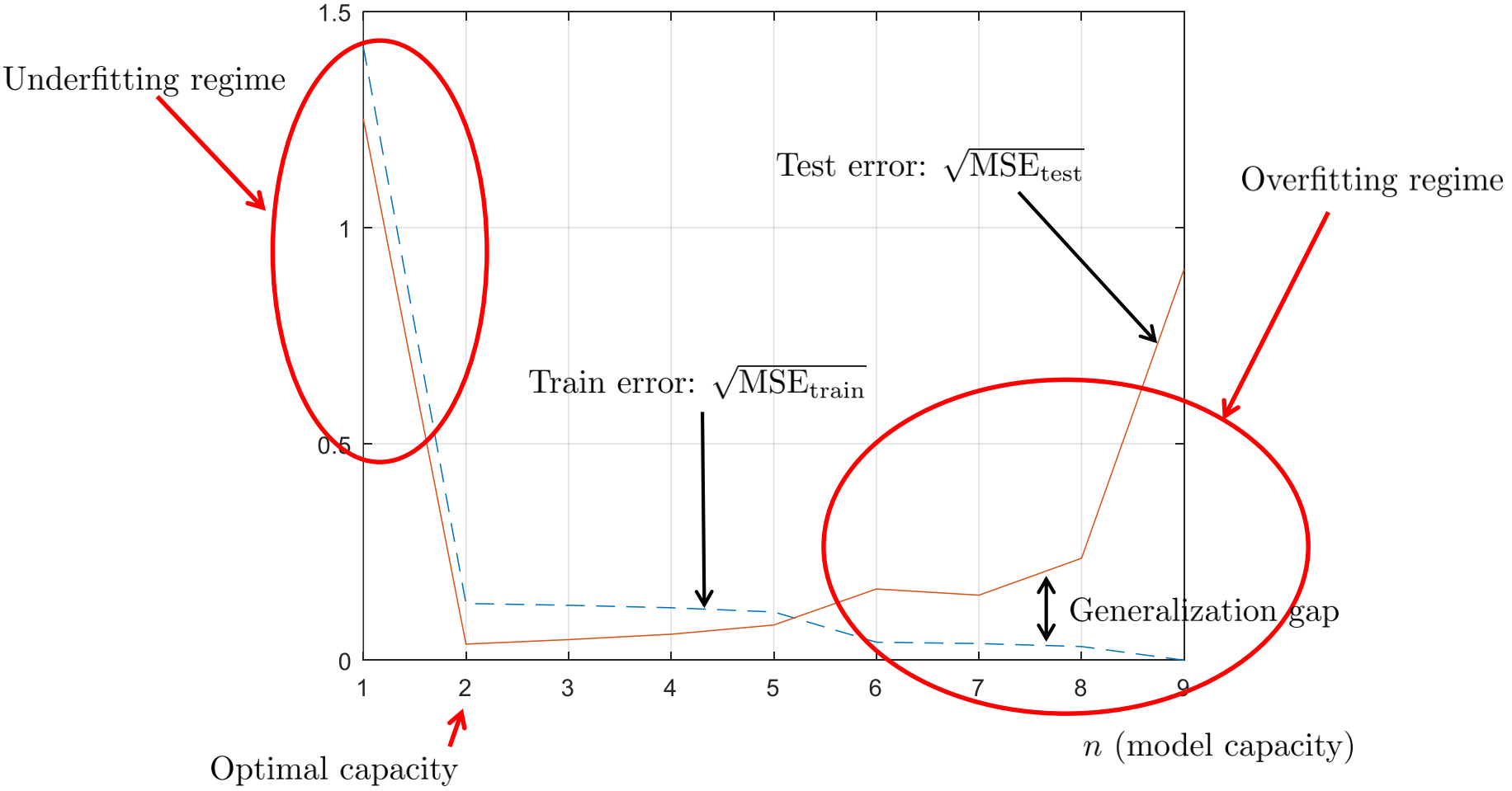
- $p(x, y)$: probability distribution that generates data for both training and test sets
- Oracle: one that knows the true probability $p(x, y)$ generating the data
- Bayes risk (or Bayes error): best test error, i.e., the best error incurred by an oracle using the true probability
  - This may not be zero due to noise, corruption, etc.
- Empirical risk (or training error)

$$\text{e.g., } \text{MSE}_{\text{train}} = \frac{1}{m'} \left\| \hat{\mathbf{y}}^{(\text{train})} - \mathbf{y}^{(\text{train})} \right\|^2$$

- Generalization error (or test error)

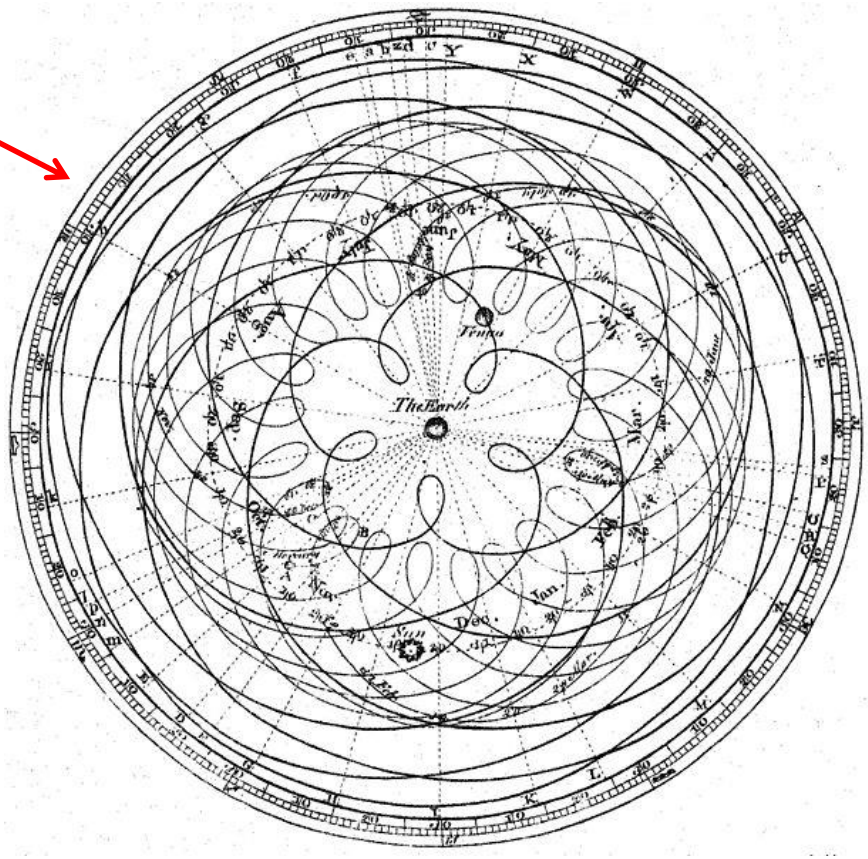
$$\text{e.g., } \text{MSE}_{\text{test}} = \frac{1}{m} \left\| \hat{\mathbf{y}}^{(\text{test})} - \mathbf{y}^{(\text{test})} \right\|^2$$

Underfitting & Overfitting



Occam's Razor

Overfitting



Geocentrism: Encyclopedia Britannica (1777)

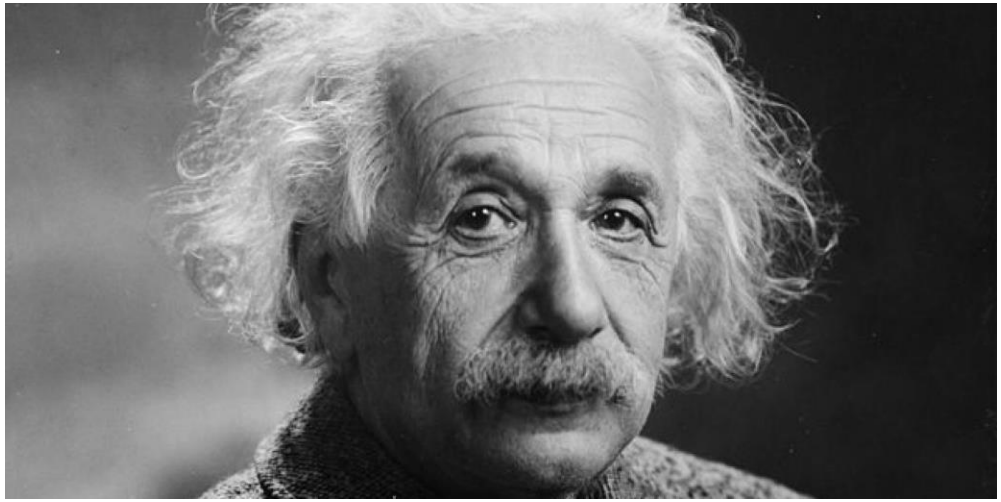
Occam's razor: A principle stating the simplest explanation should be chosen among the ones that are consistent with observations

KAIST

"Make everything as simple as possible, but not simpler." – A. Einstein

Don't overfit

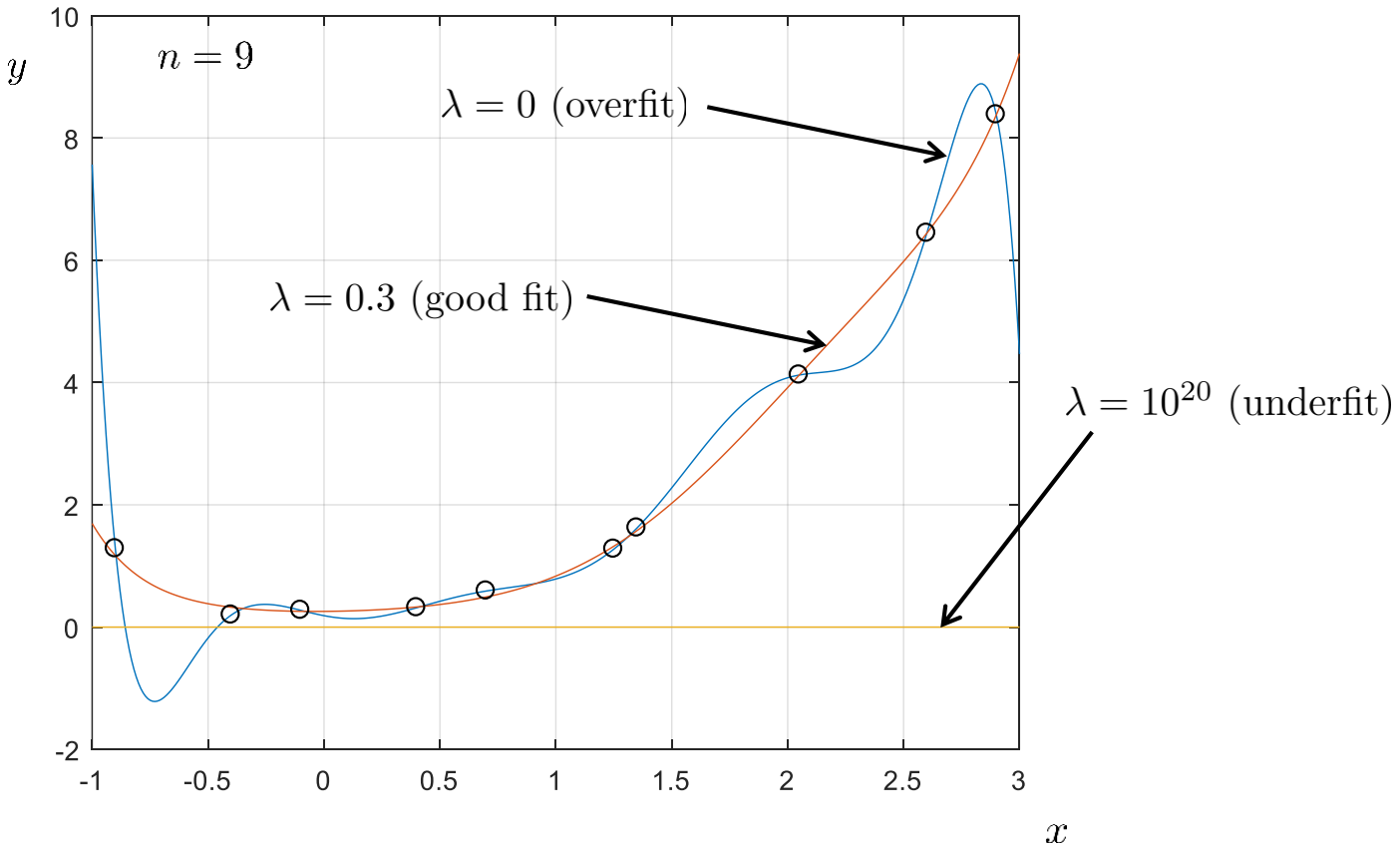
Don't underfit



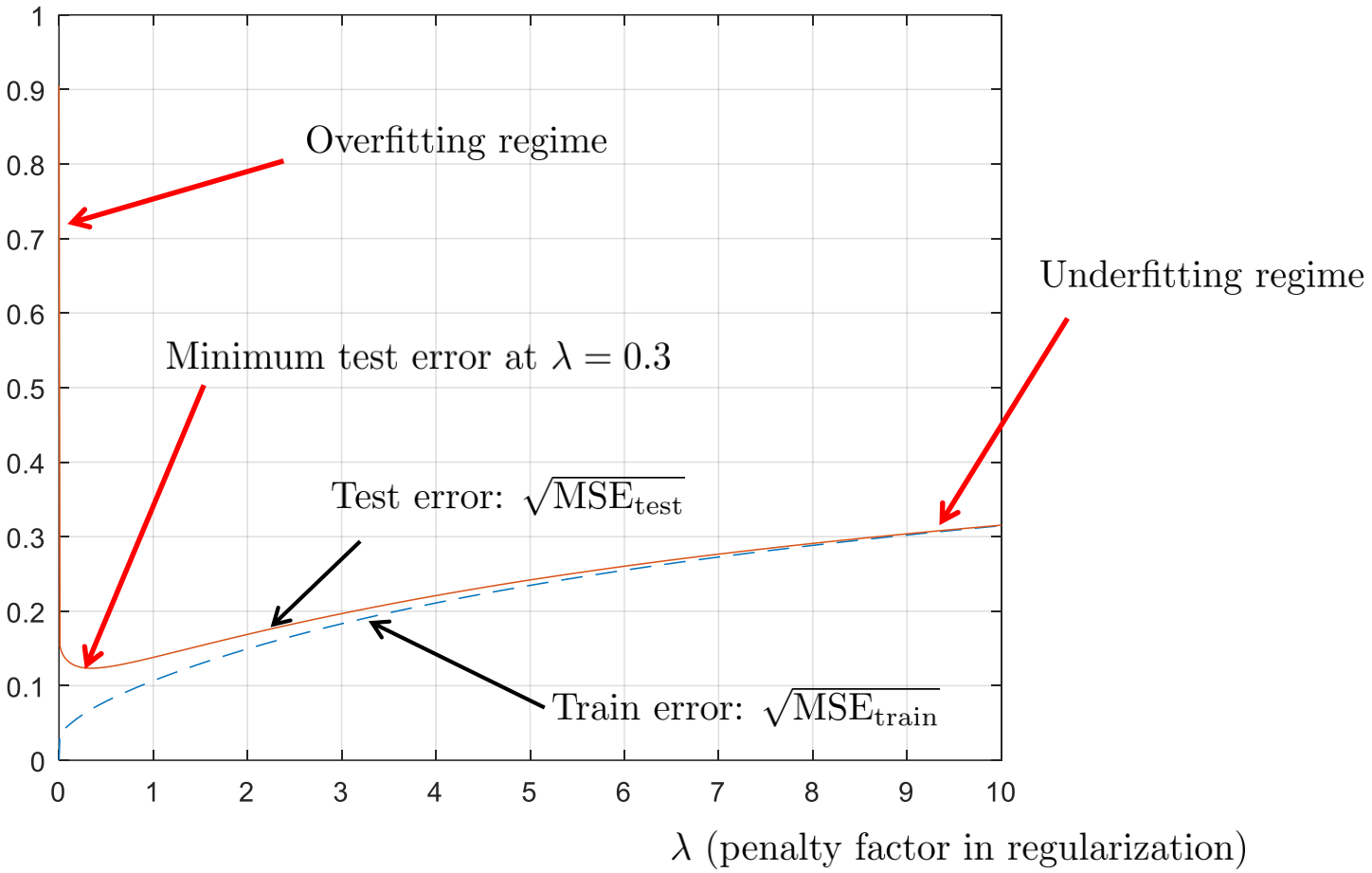
Regularization

- To reduce the generalization error, we can penalize higher model complexity.

e.g., find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = \text{MSE}_{\text{train}} + \lambda \mathbf{w}^T \mathbf{w}$

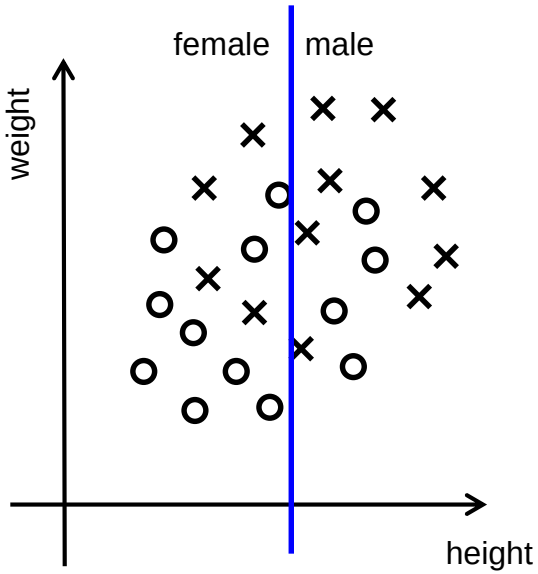


Regularization

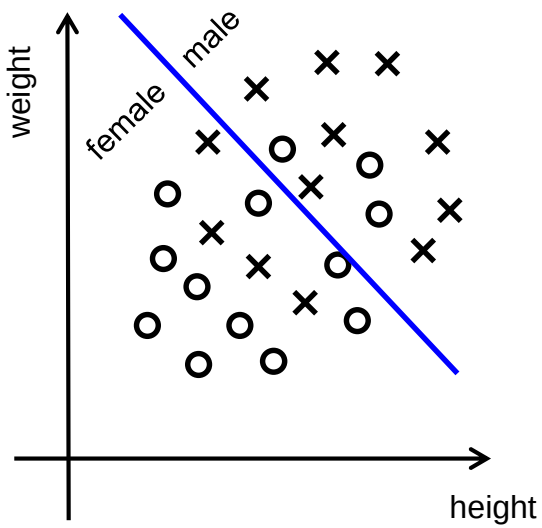


Overfitting & Underfitting in Classification

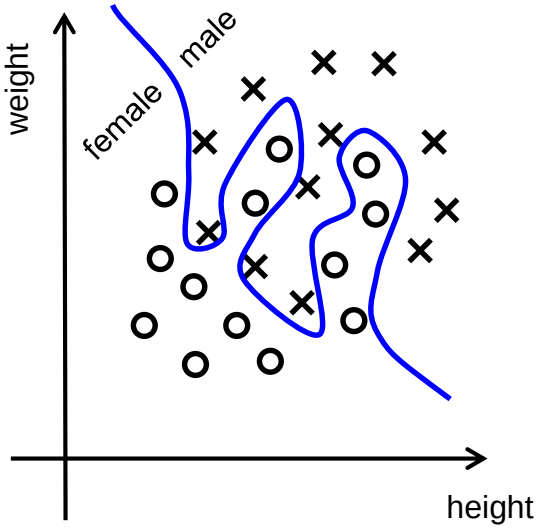
Underfit



Good fit



Overfit

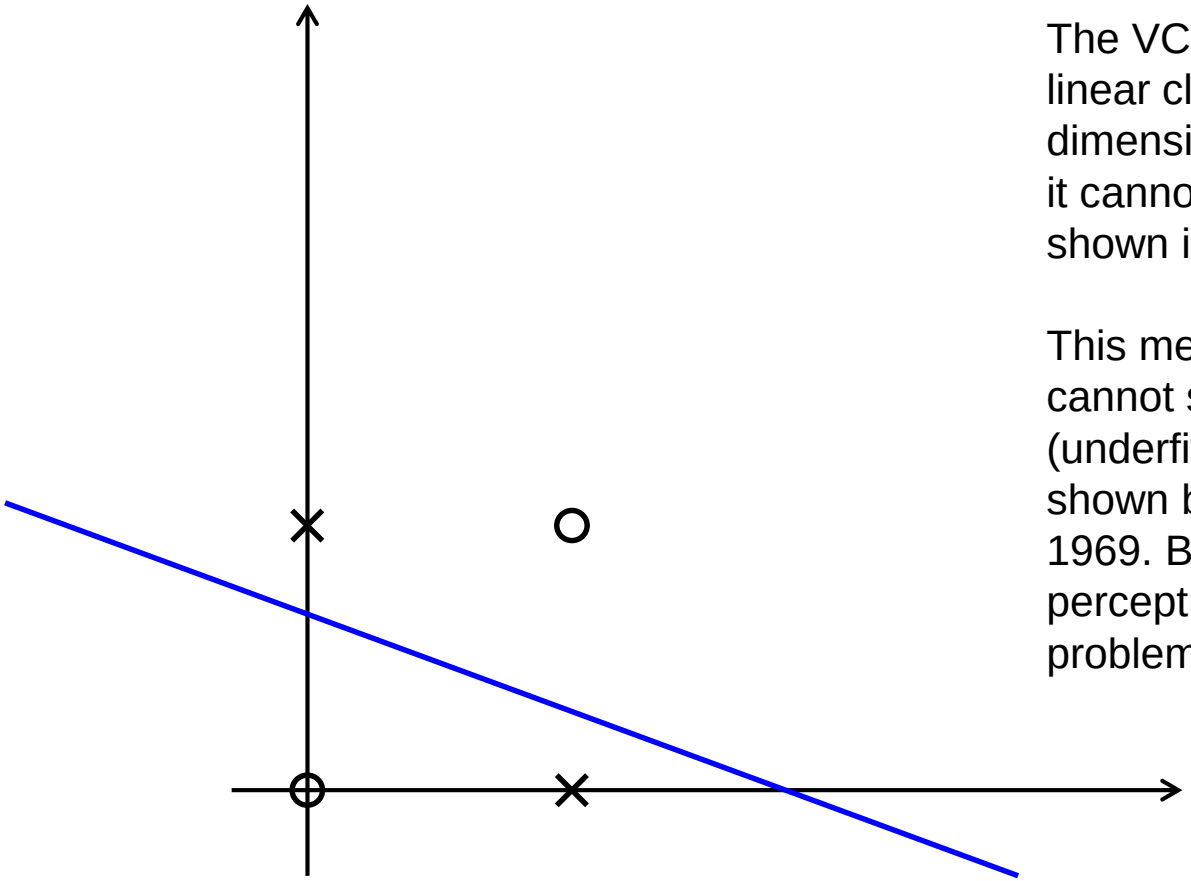


KAIST

VC Dimension

- Assume $\{x_1, \ldots, x_m\}$ are m distinct points in $\mathbb{R}^n$
- $f_\theta(\cdot)$ is said to shatter m points $\{x_1, \ldots, x_m\}$ if there exists θ such that $f_\theta(\cdot)$ can correctly classify them for *all assignments* of labels.
- VC dimension (Vapnik-Chervonenkis dimension): maximum m such that there exists $(x_1, \ldots, x_m)$ so that $f_\theta(\cdot)$ can shatter them.
- VC dimension is a measure of model capacity
- For deep learning with many neurons and layers, VC dimension is practically impossible to compute exactly

XOR Problem



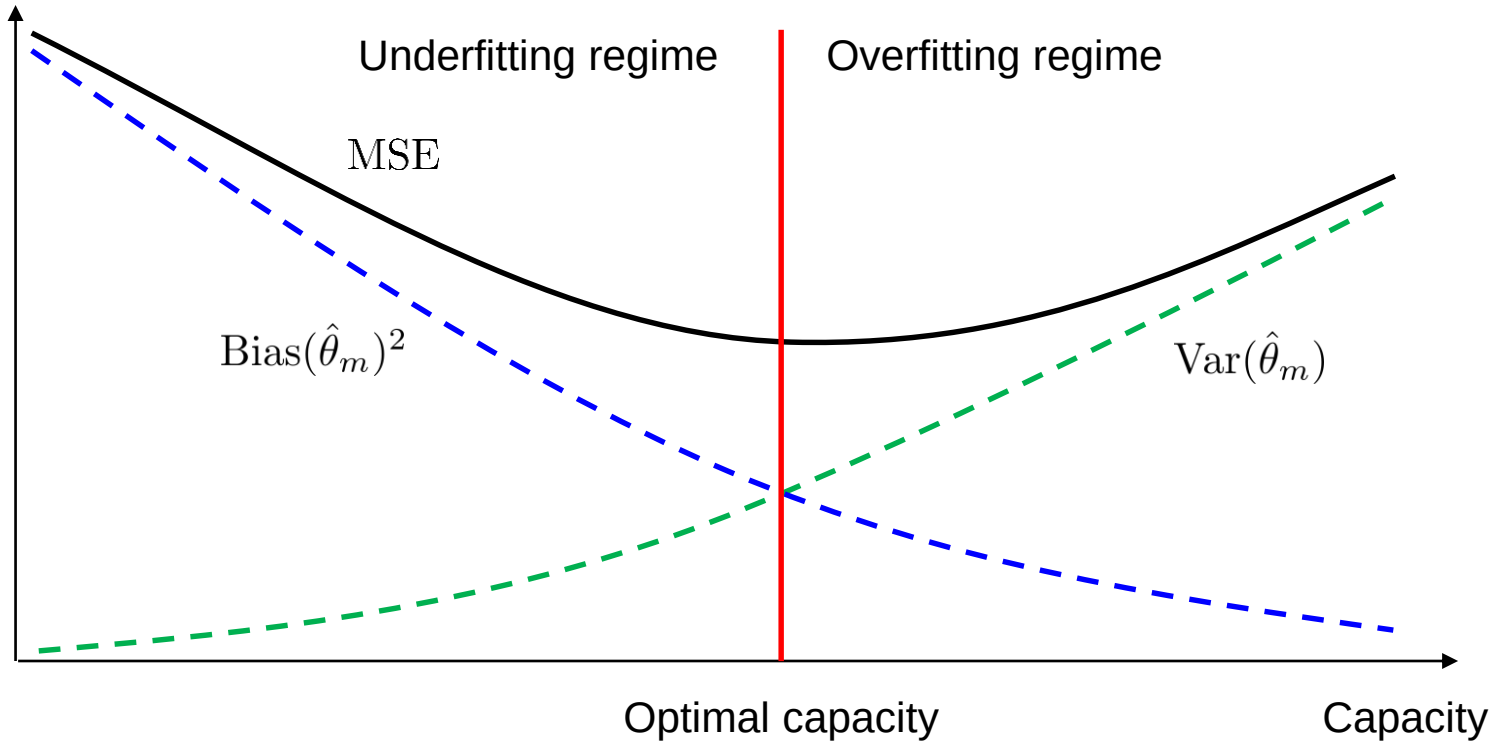
The VC dimension of a binary linear classifier in a two-dimensional plane is only 3 and it cannot shatter 4 points as shown in the figure.

This means the perceptron cannot solve the XOR problem (underfitting), which was first shown by Marvin Minsky in 1969. But, multi-layer perceptron can solve the XOR problem.

Estimation

- Point estimation: estimation of a quantity of interest, say θ
  - $\{x^{(1)}, \ldots, x^{(m)}\}$: m i.i.d. data points generated by p_θ
  - $\hat{\theta}_m = g(x^{(1)}, \ldots, x^{(m)})$
  - Cf) function estimation: estimation of a function $f(x)$ based on samples of (x, y)
- Bias of an estimator: $\text{bias}(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta$
- Variance of an estimator: $\text{Var}(\hat{\theta}_m)$
- Mean squared error (MSE)

$$\begin{aligned}
\text{MSE} &= \mathbb{E}[(\hat{\theta}_m - \theta)^2] \\
&= \mathbb{E}[(\hat{\theta}_m - \mathbb{E}(\hat{\theta}_m) + \mathbb{E}(\hat{\theta}_m) - \theta)^2] \\
&= \text{Var}(\hat{\theta}_m) + 2\mathbb{E}[\hat{\theta}_m - \mathbb{E}(\hat{\theta}_m)]\mathbb{E}[\mathbb{E}(\hat{\theta}_m) - \theta] + (\mathbb{E}(\hat{\theta}_m) - \theta)^2 \\
&= \text{Var}(\hat{\theta}_m) + 2\{\mathbb{E}(\hat{\theta}_m) - \mathbb{E}(\hat{\theta}_m)\} \cdot \{\mathbb{E}(\hat{\theta}_m) - \theta\} + \text{Bias}(\hat{\theta}_m)^2 \\
&= \text{Var}(\hat{\theta}_m) + \text{Bias}(\hat{\theta}_m)^2
\end{aligned}$$


Underfitting regime
Overfitting regime
MSE
$\text{Bias}(\hat{\theta}_m)^2$
$\text{Var}(\hat{\theta}_m)$
Optimal capacity
Capacity

ML Estimation

- What is a good point estimator for $\boldsymbol{\theta}$?
- The maximum likelihood (ML) estimator finds $\boldsymbol{\theta}$ that minimizes the KL divergence between $\hat{p}_{\text{data}}$ and p_{model}
  - $\mathbf{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$: a set of m examples drawn i.i.d. from the true but unknown distribution $p_{\text{data}}(\mathbf{x})$
  - $\hat{p}_{\text{data}}$: empirical distribution defined by the training data
  - $p_{\text{model}}(\mathbf{X}; \boldsymbol{\theta})$: distribution of $\mathbf{X}$ parameterized by $\boldsymbol{\theta}$

ML Estimation (MLE)

- Maximum likelihood estimation

$$
\begin{aligned}
\boldsymbol{\theta}_{\mathrm{ML}} &= \mathrm{argmax}_{\boldsymbol{\theta}}\, p_{\mathrm{model}}(\mathbf{X};\boldsymbol{\theta}) \\
&= \mathrm{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\mathrm{model}}(\mathbf{x}^{(i)};\boldsymbol{\theta}) \\
&= \mathrm{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\mathrm{model}}(\mathbf{x}^{(i)};\boldsymbol{\theta}) \\
&= \mathrm{argmax}_{\boldsymbol{\theta}}\, \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(\mathbf{x};\boldsymbol{\theta}) \\
&= \mathrm{argmin}_{\boldsymbol{\theta}}\, -\mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(\mathbf{x};\boldsymbol{\theta}) \quad \text{(minimization of cross entropy or NLL)} \\
&= \mathrm{argmin}_{\boldsymbol{\theta}}\, D_{\mathrm{KL}}(\hat{p}_{\mathrm{data}} \| p_{\mathrm{model}})
\end{aligned}
$$

- KL divergence

$$
D_{\mathrm{KL}}(\hat{p}_{\mathrm{data}} \| p_{\mathrm{model}}) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathrm{data}}}[\log \hat{p}_{\mathrm{data}}(\mathbf{x}) - \log p_{\mathrm{model}}(\mathbf{x}|\boldsymbol{\theta})]
$$

Conditional Version

- Conditional version

$$
\begin{aligned}
\boldsymbol{\theta}_{\mathrm{ML}} &= \mathrm{argmax}_{\boldsymbol{\theta}}\, p_{\mathrm{model}}(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta}) \\
&= \mathrm{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\mathrm{model}}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta}) \\
&= \mathrm{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\mathrm{model}}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta}) \\
&= \mathrm{argmin}_{\boldsymbol{\theta}} -\mathbb{E}_{(\mathbf{y}|\mathbf{x})\sim\hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})
\end{aligned}
$$

MLE & Linear Regression

- MLE assuming $p_{\text{model}}(y|\mathbf{x};\mathbf{w}) = \mathcal{N}(y;\hat{y}(\mathbf{x};\mathbf{w}),\sigma^2)$ and $\hat{y}(\mathbf{x};\mathbf{w}) = \mathbf{w}^T\mathbf{x}$

$$\begin{aligned}\mathbf{w}_{\text{ML}} &= \text{argmax}_{\mathbf{w}} \sum_{i=1}^{m} \log p_{\text{model}}(y^{(i)}|\mathbf{x}^{(i)};\mathbf{w}) \\ &= \text{argmax}_{\mathbf{w}} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2 \\ &= \text{argmin}_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2\end{aligned}$$

- Closed-form solution given by

$$\mathbf{w}_{\text{ML}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- Gives the same answer as linear regression minimizing the mean squared error

$$\mathbf{w}^* = \left(\mathbf{X}^{(\text{train})T}\mathbf{X}^{(\text{train})}\right)^{-1}\mathbf{X}^{(\text{train})T}\mathbf{y}^{(\text{train})}$$

Bayes' Rule

- Frequentist statstics: inference based on frequency of the data
  - Example) Point estimation of $\boldsymbol{\theta}$ based on training data
- Bayesian statistics: inference using Bayes' rule
- Bayes' rule

$$p(\boldsymbol{\theta}|\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}) = \frac{p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)})}$$

- $p(\boldsymbol{\theta})$: the prior probability (the prior)
- $p(\boldsymbol{\theta}|\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)})$: the posterior probability
- $p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}|\boldsymbol{\theta})$: the likelihood
  - was called $p_{\text{model}}(\mathbf{X}; \boldsymbol{\theta})$ in explaining MLE

MAP Estimation

- Maximum a posteriori (MAP) estimation

$$\begin{aligned}\boldsymbol{\theta}_{\text{MAP}} &= \text{argmax}_{\boldsymbol{\theta}}\, p(\boldsymbol{\theta}|\mathbf{X}) \\ &= \text{argmax}_{\boldsymbol{\theta}}\, \frac{p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{X})} \\ &= \text{argmax}_{\boldsymbol{\theta}}\, p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \\ &= \text{argmax}_{\boldsymbol{\theta}}\, \log p(\mathbf{X}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\end{aligned}$$

- MAP becomes ML if $\boldsymbol{\theta}$ is equally likely (uniform prior), i.e., $p(\boldsymbol{\theta})$ is constant
- If $p(\boldsymbol{\theta})$ is not available, we can still do MLE

Conditional Version

- Conditional version of Bayes' rule

$$p(\boldsymbol{\theta}|\mathbf{X},\mathbf{Y}) = \frac{p(\mathbf{X},\mathbf{Y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{X},\mathbf{Y})} = \frac{p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta})p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{X},\mathbf{Y})}$$

- Conditional version of MAP assuming $\mathbf{X}$ and $\boldsymbol{\theta}$ are independent, i.e., $p(\mathbf{X}|\boldsymbol{\theta}) = p(\mathbf{X})$

$$\begin{aligned}\boldsymbol{\theta}_{\text{MAP}} &= \text{argmax}_{\boldsymbol{\theta}}\, p(\boldsymbol{\theta}|\mathbf{X},\mathbf{Y}) \\ &= \text{argmax}_{\boldsymbol{\theta}}\, \frac{p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta})p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{X},\mathbf{Y})} \\ &= \text{argmax}_{\boldsymbol{\theta}}\, p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta})p(\boldsymbol{\theta})\end{aligned}$$

MAP & Linear Regression with Regularization

- MAP assuming $p(y|\mathbf{x},\mathbf{w}) = \mathcal{N}(y;\hat{y}(\mathbf{x};\mathbf{w}),1)$, $\hat{y}(\mathbf{x};\mathbf{w}) = \mathbf{w}^T\mathbf{x}$ and $p(\mathbf{x}|\mathbf{w}) = p(\mathbf{x})$
- $p(\mathbf{y}|\mathbf{X},\mathbf{w}) = \mathcal{N}(\mathbf{y};\hat{\mathbf{y}}(\mathbf{X};\mathbf{w}),I)$, $\hat{\mathbf{y}}(\mathbf{X};\mathbf{w}) = \mathbf{X}\mathbf{w}$: with m examples

$$\begin{aligned}\mathbf{w}_{\text{MAP}} &= \text{argmax}_{\mathbf{w}}\, p(\mathbf{w}|\mathbf{X},\mathbf{y}) \\ &= \text{argmax}_{\mathbf{w}}\, \log p(\mathbf{y}|\mathbf{X},\mathbf{w}) + \log p(\mathbf{w}) \\ &= \text{argmax}_{\mathbf{w}}\, -\frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \log p(\mathbf{w})\end{aligned}$$

- Assume $p(\mathbf{w}) = \mathcal{N}(\mathbf{w};\boldsymbol{\mu}_0,\boldsymbol{\Lambda}_0)$, then

$$\begin{aligned}\mathbf{w}_{\text{MAP}} &= \text{argmax}_{\mathbf{w}}\, -\frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) - \frac{1}{2}(\mathbf{w}-\boldsymbol{\mu}_0)^T\boldsymbol{\Lambda}_0^{-1}(\mathbf{w}-\boldsymbol{\mu}_0) \\ &= \text{argmin}_{\mathbf{w}}(\mathbf{w}-\boldsymbol{\mu}_m)^T\boldsymbol{\Lambda}_m^{-1}(\mathbf{w}-\boldsymbol{\mu}_m) \\ &= \boldsymbol{\mu}_m\end{aligned}$$

- $\boldsymbol{\Lambda}_m = (\mathbf{X}^T\mathbf{X} + \boldsymbol{\Lambda}_0^{-1})^{-1}$, $\boldsymbol{\mu}_m = \boldsymbol{\Lambda}_m(\mathbf{X}^T\mathbf{y} + \boldsymbol{\Lambda}_0^{-1}\boldsymbol{\mu}_0)$

MAP & Linear Regression with Regularization

- Further, if we assume $\boldsymbol{\mu}_0 = 0$ and $\boldsymbol{\Lambda}_0 = \frac{1}{\alpha} I$, then

$$\begin{aligned} \mathbf{w}_{\text{MAP}} &= (\mathbf{X}^T\mathbf{X} + \boldsymbol{\Lambda}_0^{-1})^{-1}(\mathbf{X}^T\mathbf{y} + \boldsymbol{\Lambda}_0^{-1}\boldsymbol{\mu}_0) \\ &= (\mathbf{X}^T\mathbf{X} + \alpha I)^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

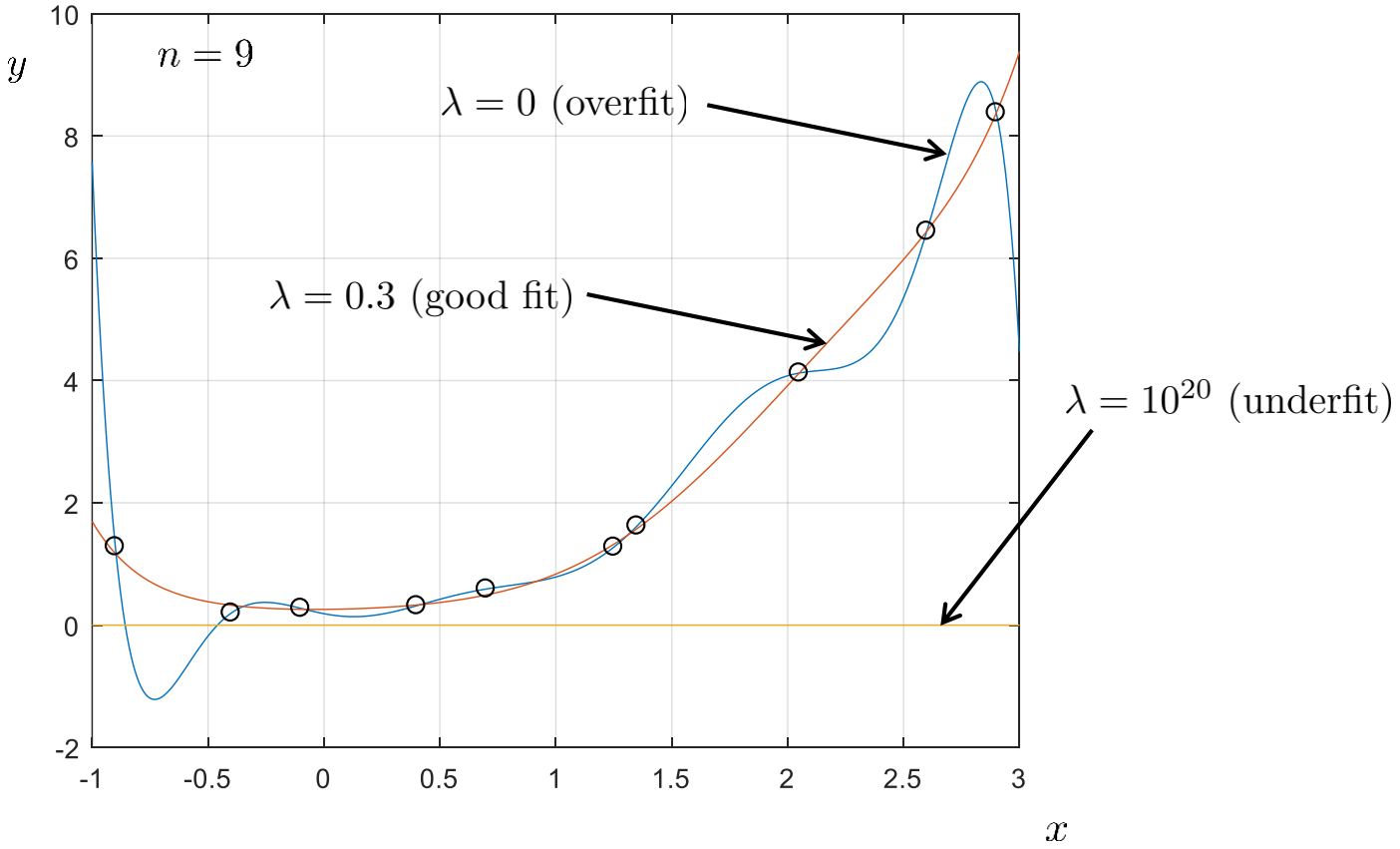
- Gives the same answer as regularized linear regression minimizing the mean squared error plus penalty, i.e., $J(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \alpha \mathbf{w}^T\mathbf{w}$
- As $\alpha \to 0$, MAP becomes ML (although $\boldsymbol{\Lambda}_0$ is not defined if $\alpha = 0$)
- Bayesian linear regression

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \propto \exp\left(-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_m)^T \boldsymbol{\Lambda}_m^{-1}(\mathbf{w} - \boldsymbol{\mu}_m)\right)$$

Recap – Regularization

- To reduce the generalization error, we can penalize higher model complexity.

e.g., find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = \text{MSE}_{\text{train}} + \lambda \mathbf{w}^T \mathbf{w}$



Consistency

- Consistent estimator:

$$\lim_{m\to\infty} \Pr(|\hat{\theta}_m - \theta| > \epsilon) = 0, \ \forall \epsilon > 0 \ \text{(weak consistency)}$$

$$\Pr\left(\lim_{m\to\infty} \hat{\theta}_m = \theta\right) = 1 \ \text{(strong consistency)}$$

- Bias of an estimator: $\text{bias}(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta$
- Unbiased estimator: $\text{bias}(\hat{\theta}_m) = 0$
- Asymptotically unbiased estimator: $\lim_{m\to\infty} \text{bias}(\hat{\theta}_m) = 0$
- Consistency implies asymptotic unbiasedness, but the converse does not hold in general.

Consistency

- MLE is consistent if
  - $p_{\text{model}}(\cdot;\boldsymbol{\theta})$ includes p_{data}
  - $\boldsymbol{\theta}_{\text{ML}}$ is unique
- Reason) MLE minimizes the KL divergence given as

$$D_{\text{KL}}(\hat{p}_{\text{data}}\|p_{\text{model}}) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{data}}}[\log \hat{p}_{\text{data}}(\mathbf{x}) - \log p_{\text{model}}(\mathbf{x}|\boldsymbol{\theta})]$$

  and the KL divergence will approach 0 as $m \to \infty$ because $p_{\text{model}}(\cdot;\boldsymbol{\theta})$ includes p_{data}. Furthermore, $\hat{p}_{\text{data}} \to p_{\text{data}}$ as $m \to \infty$. Therefore, $p_{\text{model}}(\cdot;\boldsymbol{\theta}_{\text{ML}}) \to p_{\text{data}}$ as $m \to \infty$. Finally, MLE will be consitent if $\boldsymbol{\theta}_{\text{ML}}$ is unique.

- MAP may not be consistent due to regularization

Logistic Regression

- Popular models for $p_{\text{model}}(y|\mathbf{x};\boldsymbol{\theta})$
  - Continuous case: $p(y|\mathbf{x};\boldsymbol{\theta}) = \mathcal{N}(y;\boldsymbol{\theta}^T\mathbf{x},\sigma^2)$ (correspond to linear regression)
  - Binary case: $p(y=1|\mathbf{x};\boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T\mathbf{x})$ (correspond to logistic regression)
- Logistic regression: regression using logistic (sigmoid) function

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1} = \frac{e^{z_1}}{e^{z_0}+e^{z_1}}$$

  where $z = z_1 - z_0$

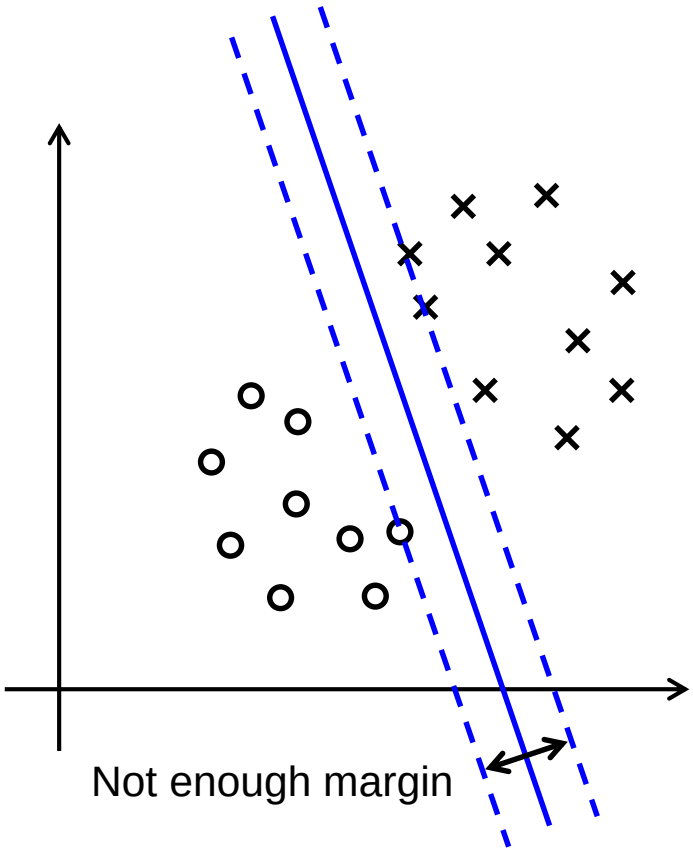
- Logistic regression can be used as a binary classification with soft output (output value is between 0 and 1 just like a probability)
- Can be generalized to k-ary classification using the softmax function

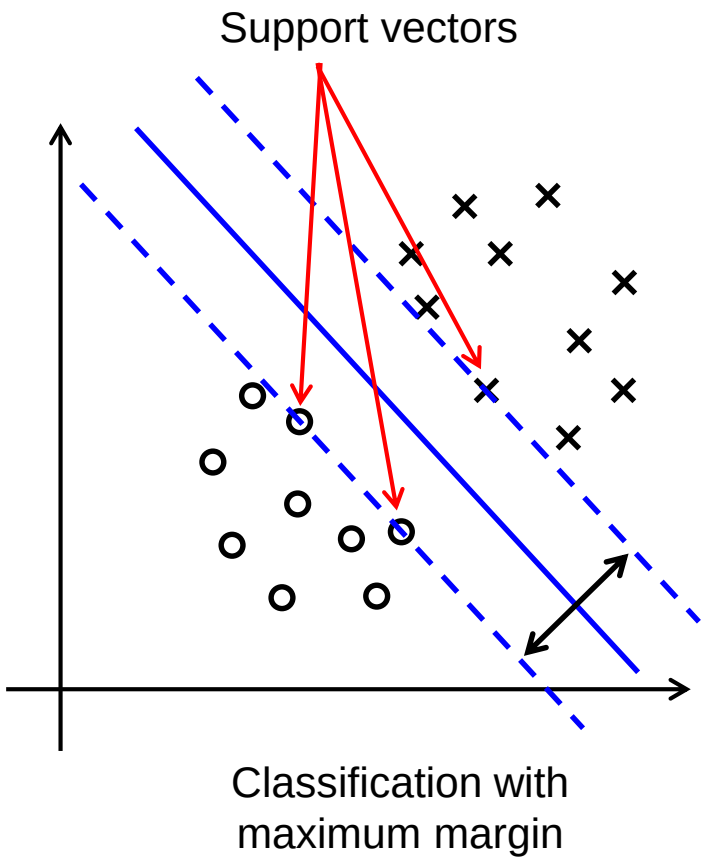
$$\frac{\exp(z_i)}{\sum_{j=1}^{k}\exp(z_j)}$$

- To be discussed in more detail later

Support Vector Machine

- Support vector machine: maximum margin classifier





Kernel Trick

- In SVM, $\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}^{(i)T}$
- Kernel trick
  - Instead of $b + \mathbf{w}^T\mathbf{x} = b + \sum_{i=1}^{m} \alpha_i \mathbf{x}^T\mathbf{x}^{(i)}$, use $b + \sum_{i=1}^{m} \alpha_i k(\mathbf{x}, \mathbf{x}^{(i)})$
  - $k(\mathbf{x}, \mathbf{x}^{(i)}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}^{(i)})$: kernel
  - Example) Gaussian kernel (or radial basis function): $k(\mathbf{u}, \mathbf{v}) = \mathcal{N}(\mathbf{u} - \mathbf{v}; 0, \sigma^2 I)$

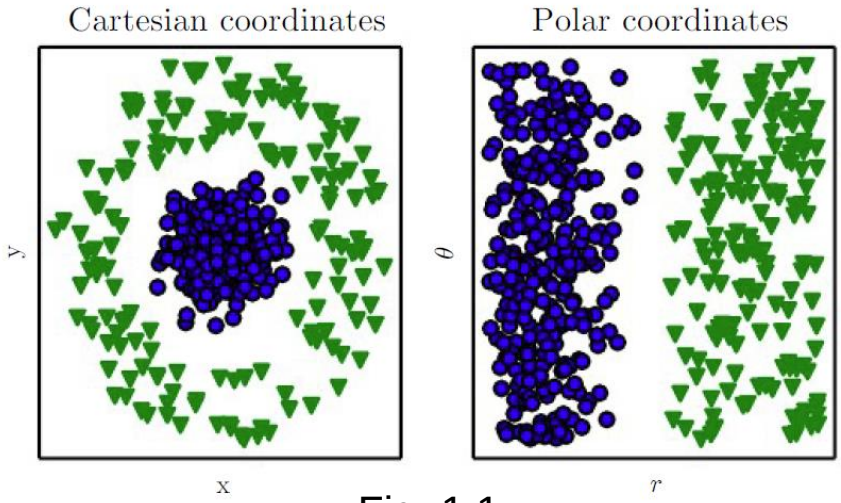


Fig. 1.1

Unsupervised Learning

- Supervised learning examples
  - Regression
  - Classification
  - k-nearest neighbors (non-parametric learning)
  - Decision tree (non-parametric learning)
  - ...
- Unsupervised learning examples
  - Principal component analysis (PCA)
  - k-means clustering
  - ...

Principal Component Analysis

- Data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$
- $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{W}^T$: SVD
- $\mathbf{X}^T\mathbf{X} = (\mathbf{U}\mathbf{\Sigma}\mathbf{W}^T)^T(\mathbf{U}\mathbf{\Sigma}\mathbf{W}^T) = \mathbf{W}\mathbf{\Sigma}^2\mathbf{W}^T$
- Define $\mathbf{Z} = \mathbf{X}\mathbf{W}$, then

$$\mathbf{Z}^T\mathbf{Z} = \mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W} = \mathbf{W}^T\mathbf{W}\mathbf{\Sigma}^2\mathbf{W}^T\mathbf{W} = \mathbf{\Sigma}^2$$

- If $\mathbf{X}$ has zero mean, then so does $\mathbf{Z}$. Then, the unbiased estimation of the covariance matrix of $\mathbf{z}$ from the samples $\mathbf{Z}$ is given by

$$\frac{1}{m-1}\mathbf{Z}^T\mathbf{Z} = \frac{1}{m-1}\mathbf{\Sigma}^2$$

Principal Component Analysis

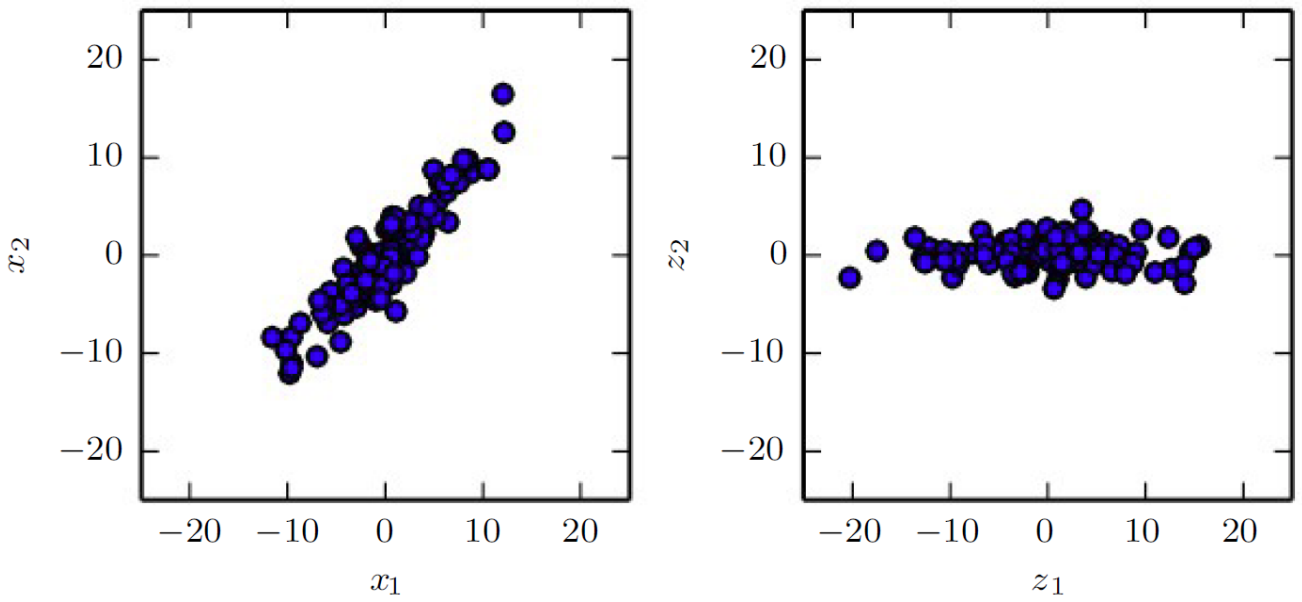


Fig. 5.8

Hyperparameters

- Hyperparameters: parameters that are not learned by the learning algorithm but instead specified outside the learning algorithm
  - e.g., the maximum degree for polynomial regression, λ in regularization
  - Possible to learn hyperparameters also by running another learning algorithm outside (nested learning algorithms)
- Validation set: a data set separate from the training set to keep track of progress of learning and to modify hyperparameters
  - e.g., split data set into 80%:20% for training and validation
  - problem: only 20% of data is used for validation
- Cross validation
  - k-fold cross validation

Assignments

- Reading assignment: Chapter 6 of DL book
- Homework #1
  - Due: 9/27 (Wed) 1pm
    - No late submissions allowed for homeworks since solutions will be uploaded immediately after the deadline
  - Will be uploaded soon (by Wednesday at the latest)
  - For each problem, we will give you choices between
    - Analysis
    - Programming
  - Separate normalization for analysis and programming