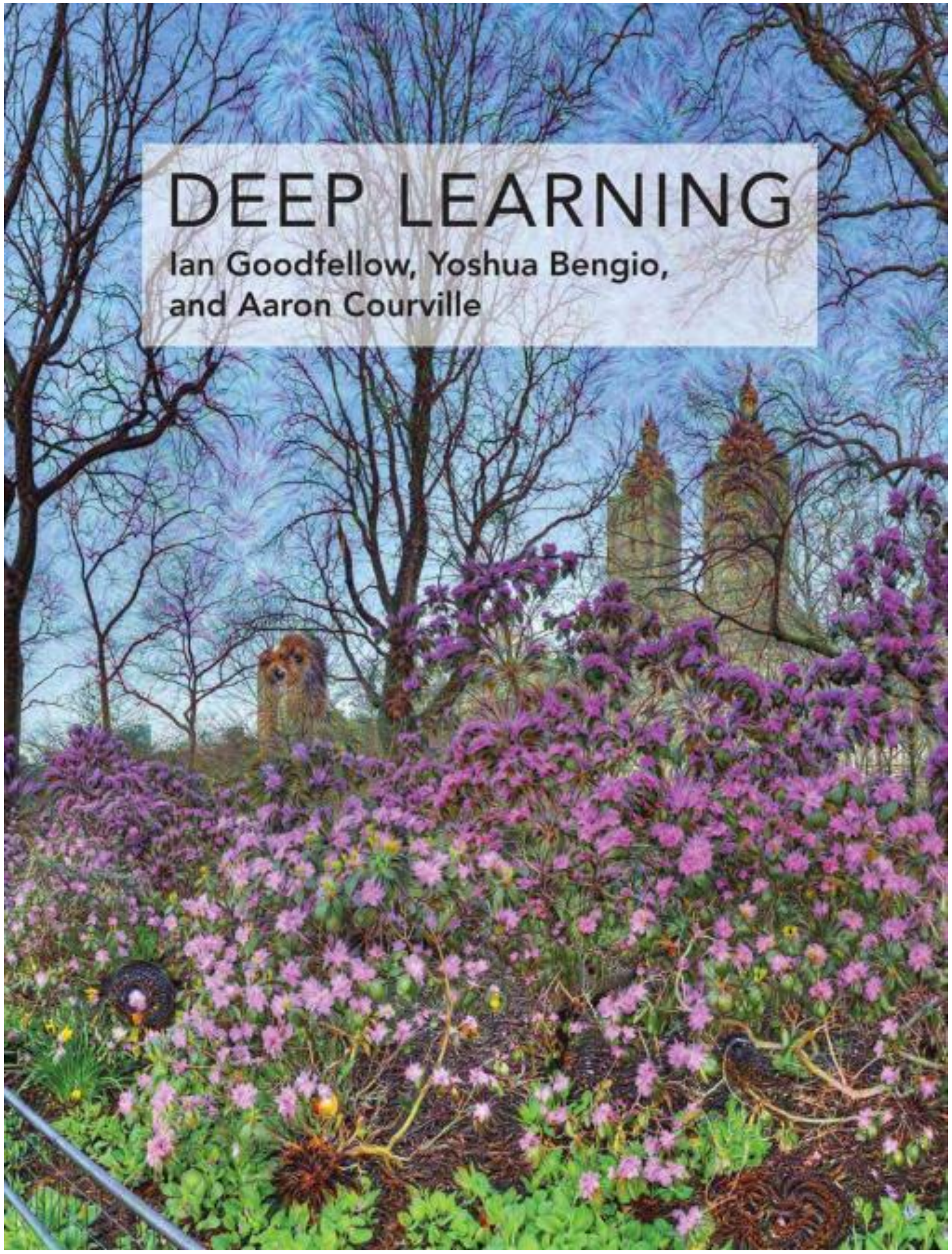
Fundamentals of Media Processing (Machine Learning Part)

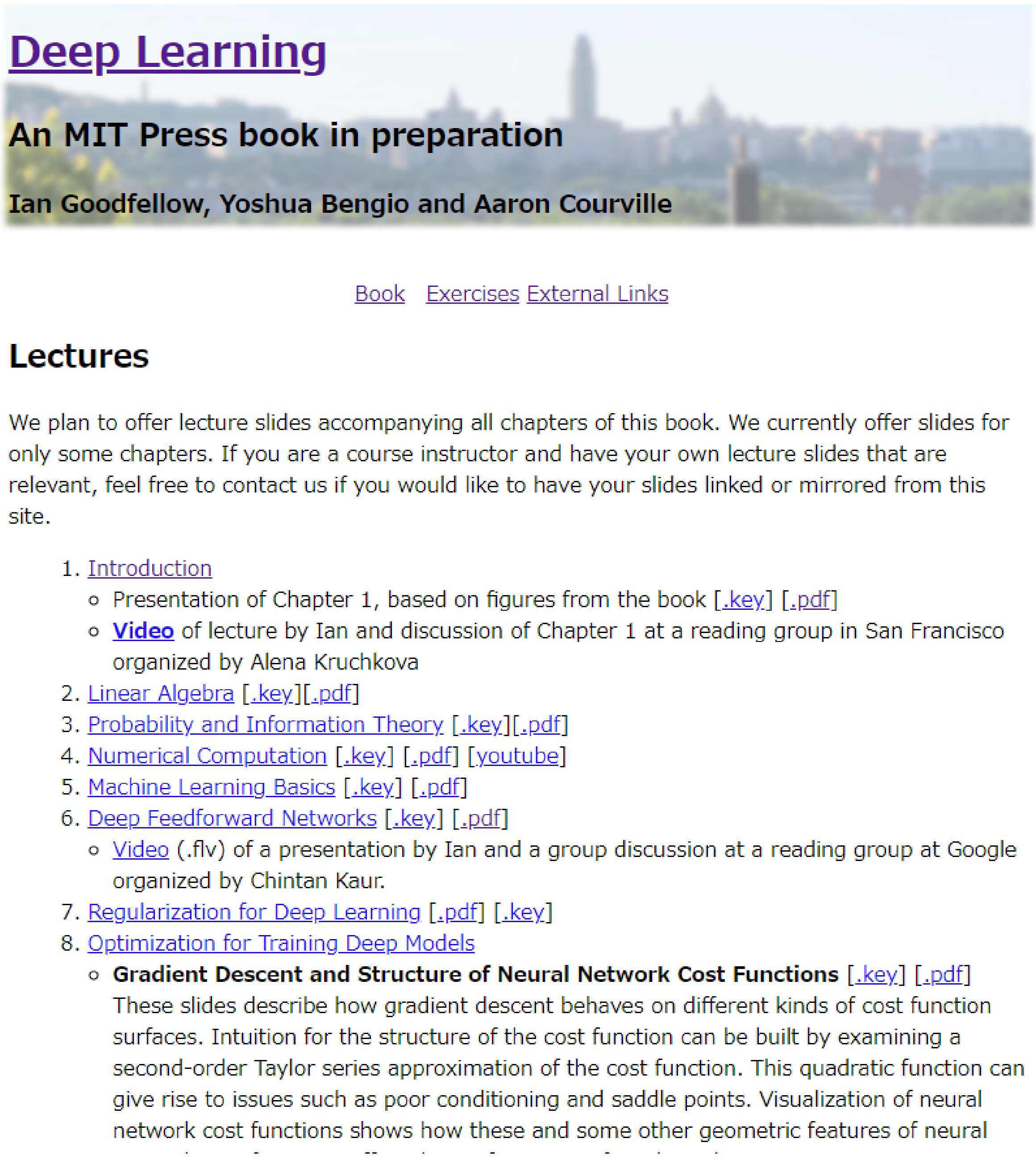
Lecturer:
佐藤　真一（Prof. SATO　Shinichi）
池畑　諭（Prof. IKEHATA　Satoshi）10/27, 11/10, 11/17, 11/21, 12/1, 12/8
山岸　順一（Prof. Junichi Yamagishi）
児玉　和也（Prof. KODAMA　Kazuya）
孟　洋（Prof. MO　Hiroshi）



Chapter 1-9 (out of 20)

An introduction to a broad range of topics in deep learning, covering mathematical and conceptual background, deep learning techniques used in industry, and research perspectives.

- Due to my background, I will mainly talk about "image"
- I will introduce some applications beyond this book



Deep Learning

An MIT Press book in preparation

Ian Goodfellow, Yoshua Bengio and Aaron Courville

Book Exercises External Links

Lectures

We plan to offer lecture slides accompanying all chapters of this book. We currently offer slides for only some chapters. If you are a course instructor and have your own lecture slides that are relevant, feel free to contact us if you would like to have your slides linked or mirrored from this site.

1. Introduction
   - Presentation of Chapter 1, based on figures from the book [.key] [.pdf]
   - **Video** of lecture by Ian and discussion of Chapter 1 at a reading group in San Francisco organized by Alena Kruchkova
2. Linear Algebra [.key][.pdf]
3. Probability and Information Theory [.key][.pdf]
4. Numerical Computation [.key] [.pdf] [youtube]
5. Machine Learning Basics [.key] [.pdf]
6. Deep Feedforward Networks [.key] [.pdf]
   - Video (.flv) of a presentation by Ian and a group discussion at a reading group at Google organized by Chintan Kaur.
7. Regularization for Deep Learning [.pdf] [.key]
8. Optimization for Training Deep Models
   - **Gradient Descent and Structure of Neural Network Cost Functions** [.key] [.pdf] These slides describe how gradient descent behaves on different kinds of cost function surfaces. Intuition for the structure of the cost function can be built by examining a second-order Taylor series approximation of the cost function. This quadratic function can give rise to issues such as poor conditioning and saddle points. Visualization of neural network cost functions shows how these and some other geometric features of neural

Free copy of the book and useful materials are available at

https://www.deeplearningbook.org/lecture_slides.html

Schedule

10/27 (Today)	Introduction	Chap. 1
	probability, information theory, numerical computation	Chap. 2,3,4
11/10	Machine Learning Basics	Chap. 5
11/17, 11/27, 12/1	Deep Feedforward Networks	Chap. 6
	Regularization and Deep Learning	Chap. 7
	Optimization for Training Deep Models	Chap. 8
12/8	Convolutional Neural Networks	Chap. 9 and more

Machine Learning Basics

What is 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 measures P, improves with experience E (By Mitchell (1997))

- Task, T
  - Classification
  - Regression
  - Transcription
  - Machine translation
  - Structured output
  - Anomaly detection
  - Synthesis and sampling
  - Imputation of missing values
  - Denoising
  - Density estimation
  - …

- Performance Measure, P
  - Accuracy
  - Error rate
  - …

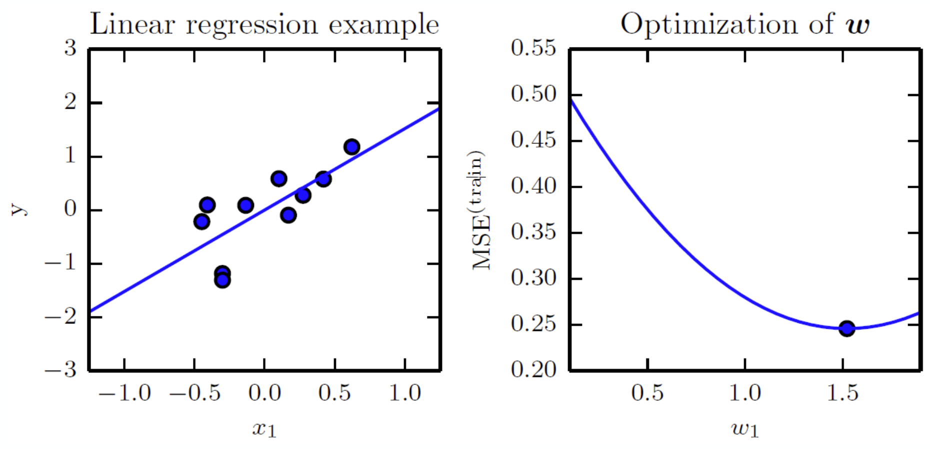
- Experience, E
  - Supervised (data with label)
  - Unsupervised (unlabeled data)
  - Semisupervised (both)
  - Reinforcement learning
  - …

Linear Regression $\hat{y} = \boldsymbol{\omega}^T \boldsymbol{x}$

- $\boldsymbol{\omega} \in \mathbb{R}^n$ is a vector of parameters (weights)
- ***Task*** T: Take a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input and predict the value $\hat{y} \in \mathbb{R}$ as $\boldsymbol{\omega}^T \boldsymbol{x}$
- ***Experience*** E: We have a set of $(\boldsymbol{x}, \boldsymbol{y})$: Divided into test and training sets
- ***Performance Measure*** P: $\mathrm{MSE}_{\mathrm{test}} = \frac{1}{m}\sum_i \left(\hat{y}_i^{\mathrm{test}} - y_i^{\mathrm{test}}\right)^2$
- To minimize MSE,

$$\nabla \mathrm{MSE}_{\mathrm{test}} = 0 \Leftrightarrow \nabla_{\boldsymbol{\omega}} \frac{1}{m} \left\| \hat{\boldsymbol{y}} - \boldsymbol{y}^{\mathrm{train}} \right\|_2^2 = 0 \Leftrightarrow \nabla_{\boldsymbol{\omega}} \frac{1}{m} \left\| X^{\mathrm{train}} \boldsymbol{\omega} - \boldsymbol{y}^{\mathrm{train}} \right\|_2^2 = 0$$

$$\boldsymbol{\omega} = \left(X^{\mathrm{train}^T} X^{\mathrm{train}} \right)^{-1} X^{\mathrm{train}^T} \boldsymbol{y}^{\mathrm{train}}$$



Generalization, Overfitting and Underfitting

- **Generalization**: The ability in performing on new test data

- **i.i.d (**independent and identically distributed**)** assumption:
    - Each data is independent
    - Training/test data are drawn from the same distribution

- **Overfitting**
    - Training error is quite small but test error is huge (E.g., model is to general)

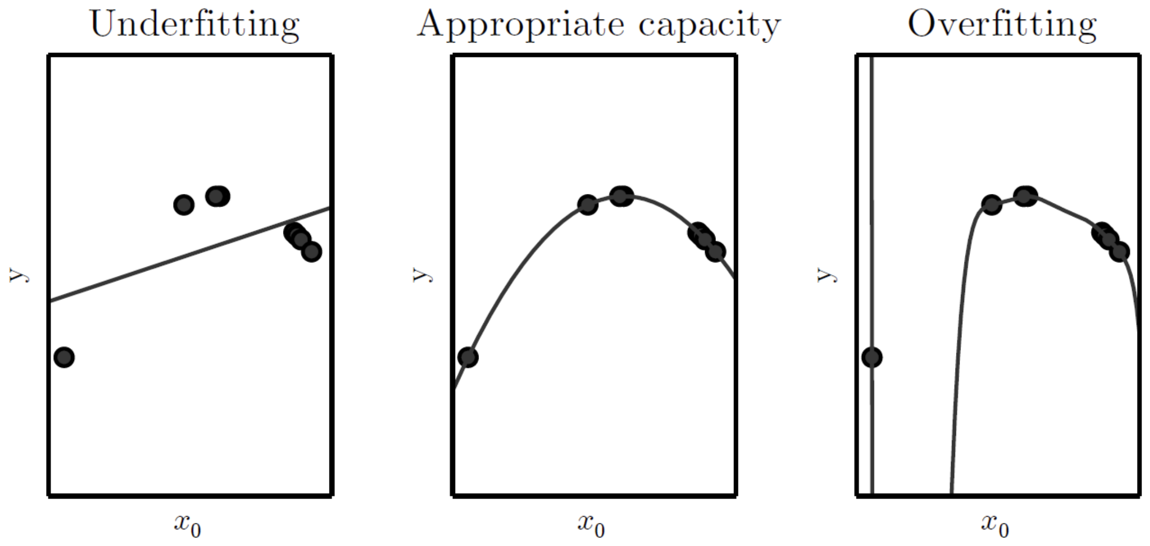
- **Underfitting**
    - Training error is not sufficiently small (E.g., model is too simple)

Capacity

- Ability to fit a wide variety of functions
- One way to control its capacity is hypothesis space
  - Set of functions acceptable (e.g., linear functions for linear regression)

- ***Representational capacity***: Defined by the model
  - E.g., capacity of $y = \omega x$ < capacity of $y = \omega_2 x^2 + \omega_1$

- ***Effective capacity***: Defined by the algorithm
  - E.g., convex algorithm cannot handle nonconvex function

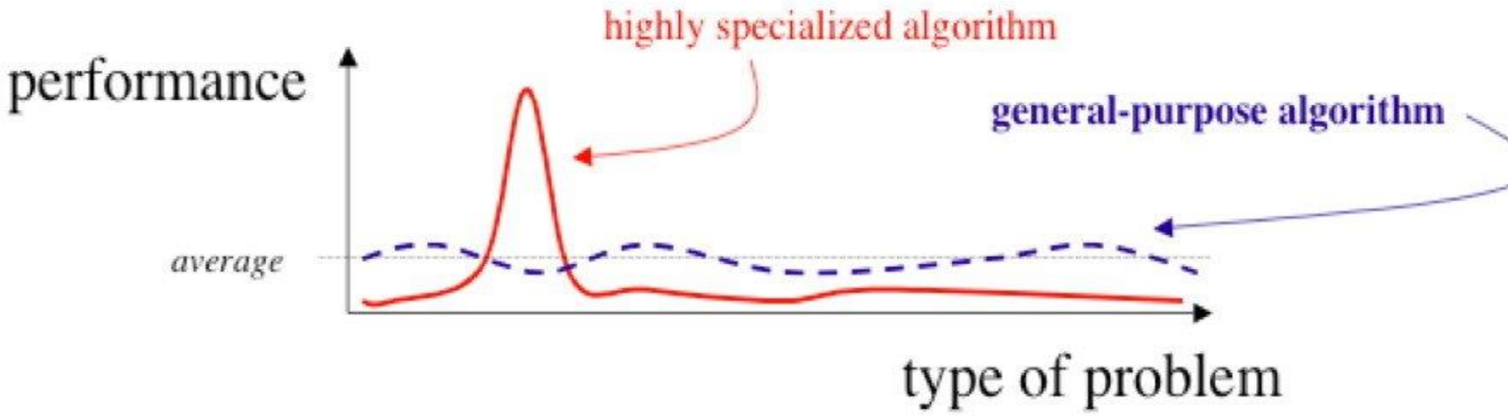


No Free Lunch Theorem

"No machine learning algorithm is universally any better than any other"

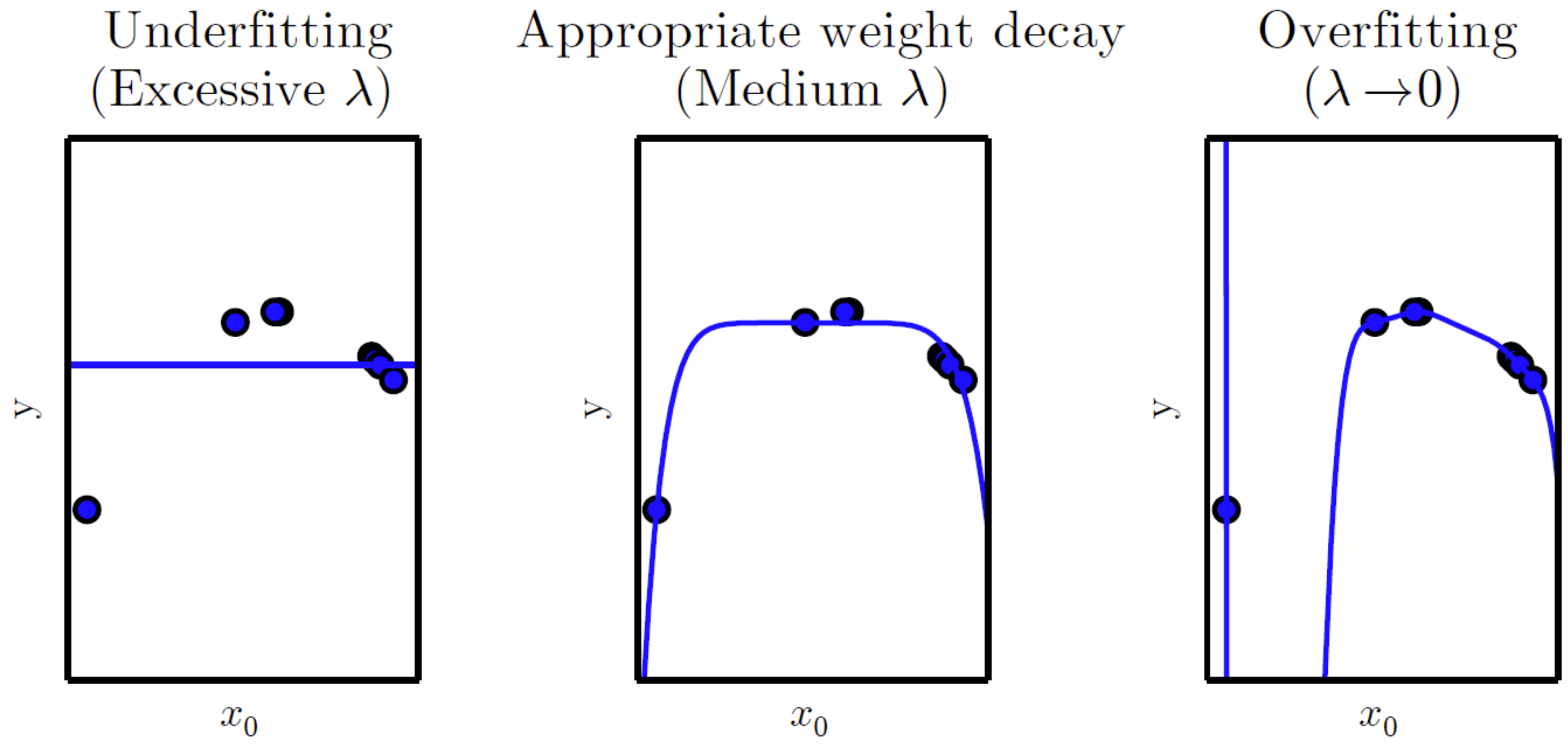
Averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

- In reality, we observe a specific probability distributions
- Our goal is to understand what kinds of distributions are relevant to the "real world" that an AI agent experiences, and what kinds of machine learning algorithms perform well on data drawn from the kinds of data-generateing distributions we care about



Regularization (L2, weight decay)

$$J(\boldsymbol{\omega}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{\omega}^T \boldsymbol{\omega}$$



Hyperparameters, Validation Sets, Cross-validation

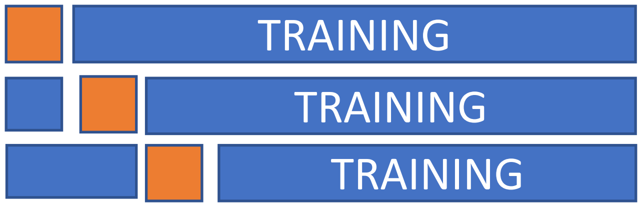
- Hyperparameter control the behavior/capacity of the learning algorithm

$$\hat{y} = \boldsymbol{\omega}^T \boldsymbol{x} \qquad J(\boldsymbol{\omega}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{\omega}^T \boldsymbol{\omega}$$

- Hard to control the hyperparameter to avoid over-under fitting

- Validation set is the samples outside the training
  - Generally, samples are drawn from the training set (e.g., 80% for training, 20% for validation)

- **K-fold Cross-validation** (When training/test data is limited)
  - Splitting dataset into k nonoverlapping subsets, the test error may then be estimated by taking the average test error across k trials
  - For each test, *i-th* subset is test, others are training



Estimators

- **Point Estimation (E.g., *Maximum Likelihood Estimation*)**
  - The single "best" prediction (e.g., single parameter or vector)
  - Given i.i.d data points $\boldsymbol{x}^i$, the point estimator is:

$$\widehat{\boldsymbol{\theta}}_m = g(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m)$$

  - Frequentist perspective
    - $\widehat{\boldsymbol{\theta}}_m$ is random and <u>does not require to be close to true</u>

- **Function Estimation**
  - Simply a point estimator in function space
  - E.g., the parameter $\boldsymbol{\omega}$ can be interpreted as both point/function estimator

Bias

- The expected deviation from the true value of the function or parameter
  - $\text{bias}(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta$

- An estimator $\hat{\theta}_m$ is unbiased if $\text{bias}(\hat{\theta}_m) = 0$, which implies $\mathbb{E}(\hat{\theta}_m) = \theta$
- An estimator $\hat{\theta}_m$ is asymptotically unbiased if $\lim_{m\to\infty} \text{bias}(\hat{\theta}_m) = 0$

- Example Gaussian Distribution: Consider a set of samples $\{x^1, \dots, x^m\}$ that are i.i.d according to a Gaussian distribution with mean μ as

$$p(x^i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x^i - \mu)^2}{\sigma^2}\right) \qquad \hat{\mu}_m = \frac{1}{m}\sum_i^m x^i$$ (Sample mean estimator)

$$\text{bias}(\hat{\mu}_m) = \mathbb{E}(\hat{\mu}_m) - \mu = \frac{1}{m}\sum_i^m \mathbb{E}[x^i] - \mu = \mu - \mu = 0$$ (unbiased)

- **Consistency**: $\text{plim}_{m\to\infty}\ \hat{\theta}_m = \theta$
  - As the number of data grows, the estimator converges to true
  - The unbiased estimator does not always has consistency (e.g., Gaussian when the number of sample in the estimator is one)

Variance and Standard Error

- The deviation from the expected estimator value that any particular sampling of the data is likely to cause

- Variance: $\mathrm{Var}(\hat{\theta})$, Standard error: $\mathrm{SE}(\hat{\theta}) = \sqrt{\mathrm{Var}(\hat{\theta})}$

- Example: Gaussian Distribution: Consider a set of samples $\{x^1, \dots, x^m\}$ that are i.i.d according to a Gaussian distribution with mean μ as

$$p(x^i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x^i - \mu)^2}{\sigma^2}\right) \qquad \hat{\mu}_m = \frac{1}{m}\sum_i^m x^i$$ (Sample mean estimator)

$$\mathrm{Var}(\hat{\mu}_m) = \mathrm{Var}\left(\frac{1}{m}\sum_i^m x^i\right) = \frac{1}{m^2}\sum_i^m \mathrm{Var}(x^i) = \frac{1}{m}\sigma$$

- Trade-off between bias and variance (How to design better estimator)
  - MSE (Mean Squared Error) = $\mathbb{E}\left[(\hat{\theta}_m - \theta)^2\right] = \mathrm{bias}(\hat{\theta}_m)^2 + \mathrm{Var}(\hat{\theta}_m)$

(*) $\mathrm{bias}(\hat{\mu}_m) = 0$

Maximum Likelihood Estimation

- Consider m examples $\mathbb{X} = \{x^1, \ldots, x^m\}$ from the true, unknown dist. $p_{\text{data}}(\boldsymbol{x})$
- Let $p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})$ maps any configuration x to a real number estimating the true probability $p_{\text{data}}(\boldsymbol{x})$

- The **maximum likelihood estimator** for θ is defined as

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} p_{\text{model}}(\mathbb{X}; \boldsymbol{\theta}) = \arg\max_{\theta} \Pi_i^m p_{\text{model}}(x^i; \boldsymbol{\theta})$$

$$\Leftrightarrow \boldsymbol{\theta}_{ML} = \arg\max_{\theta} \frac{1}{m} \sum_i^m \log p_{\text{model}}(x^i; \boldsymbol{\theta})$$

$$\Leftrightarrow \boldsymbol{\theta}_{ML} = \arg\max_{\theta} \mathbb{E}_{x \sim \hat{p}_{\text{data}}}[\log p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})]$$

- Equivalent to minimizing the dissimilarity between the empirical distribution and model distribution via KL divergence or cross entropy

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

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

Example (Unfair Coin) (1)

1. One wishes to determine how biased an unfair coin is. Call the probability of tossing a 'head' p. The goal then becomes to determine p. (so here p is θ).
2. Suppose the coin is tossed 80 times: i.e. the sample might be something like x1 = H, x2 = T, ..., x80 = T, and the count of the number of heads "H" is observed.
3. Suppose the outcome is 49 heads and 31 tails, and suppose the coin was taken from a box containing three coins: one which gives heads with probability p = 1/3 or p=1/2 or p = 2/3. The coins have lost their labels, so which one it was is unknown.

$P = 1/3$

$P = 1/2$

$P = 2/3$

Example (Unfair Coin) (2)

1. $P\left[H = 49 \middle| p = \frac{1}{3}\right] = \binom{80}{49}\left(\frac{1}{3}\right)^{49}\left(1-\frac{1}{3}\right)^{31} \approx 0.000$
2. $P\left[H = 49 \middle| p = \frac{1}{2}\right] = \binom{80}{49}\left(\frac{1}{2}\right)^{49}\left(1-\frac{1}{2}\right)^{31} \approx 0.012$
3. $P\left[H = 49 \middle| p = \frac{2}{3}\right] = \binom{80}{49}\left(\frac{2}{3}\right)^{49}\left(1-\frac{2}{3}\right)^{31} \approx 0.054$

The likelihood maximized when p = 2/3, and so this is the maximum likelihood estimate for *p*

$P = 1/3$

$P = 1/2$

$P = 2/3$

Conditional Log-Likelihood and Mean Squared Error

- MLE that is generalized to estimate a conditional probability $P(\boldsymbol{y}|X;\boldsymbol{\theta})$

  E.g., given X then predict Y

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} P(\boldsymbol{y}|X;\boldsymbol{\theta}) = \arg\max_{\theta} \sum_{i}^{m} \log P(y^i|\boldsymbol{x}^i;\boldsymbol{\theta})$$

- Example: Linear regression as Maximum Likelihood;
  - $p\left(y^i\middle|\boldsymbol{x}^i\right) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(\frac{\left\|y_i-\boldsymbol{\omega}^T\boldsymbol{x}_i\right\|^2}{\sigma^2}\right)\right)$

- The conditional log-likelihood is given by:

$$\sum_{i}^{m} \log P(y^i|\boldsymbol{x}^i;\boldsymbol{\omega}) = \underbrace{-m\log\sigma - \frac{m}{2}\log(2\pi)}_{\text{const.}} - \sum_{i}^{m} \frac{\boxed{\left\|\hat{y}^i - y^i\right\|^2}}{2\sigma^2}$$

y^i :output of the linear regression on the *i*-th input

m: the number of training examples

Properties of Maximum Likelihood

- Under the appropriate conditions, the maximum likelihood estimator has the property of consistency (though the statistical efficiency is different)

  - The true distribution $p_{\text{data}}(\boldsymbol{x})$ lies within the model family $p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})$
  - The true distribution $p_{\text{data}}(\boldsymbol{x})$ corresponds to exactly one value of θ

- For consistency and efficiency, maximum likelihood is often considered the preferred estimator to use for machine learning

Bayesian Statistics

- Frequent statistics
  - estimates a single value $\boldsymbol{\theta}$, then makes predictions
  - True parameter θ is unknown but fixed

- ***Bayesian statistics***
  - considers full distribution over $\boldsymbol{\theta}$ when making a prediction
  - True parameter θ is uncertain thus is a random variable
  - Instead, the *prior probability distribution* $p(\theta)$ is known

- Consider that we have a set of data samples $\{x^1, \dots, x^m\}$, Bayes' rule gives

$$\underbrace{p(\boldsymbol{\theta}|x^1, \dots, x^m)}_{\text{Posterior probability}} = \frac{\overbrace{p(x^1, \dots, x^m|\boldsymbol{\theta})}^{\text{Data likelihood}}\overbrace{p(\boldsymbol{\theta})}^{\text{Prior}}}{p(x^1, \dots, x^m)}$$

- After observing m examples, the predicted distribution over the next data sample is given by

$$p(x^{m+1}|x^1, \dots, x^m) = \int p(x^{m+1}|\boldsymbol{\theta})p(\boldsymbol{\theta}|x^1, \dots, x^m)d\boldsymbol{\theta}$$

Frequent vs Bayesian in Linear Regression

$$\hat{y} = \boldsymbol{\omega}^T \boldsymbol{x} \qquad \left(y_i^{\text{train}}, \boldsymbol{x}_i^{\text{train}}\right)\ i = 1, \ldots, \text{m} \Rightarrow \ (\boldsymbol{y}, X)$$

$$\sum_{i=1}^{m} \log P(y^i | \boldsymbol{x}^i; \omega)$$

$$= -m \log \sigma - \frac{m}{2} \log(2\pi) - \sum_{i=1}^{m} \frac{\left\|\hat{y}^i - y^i\right\|^2}{2\sigma^2}$$

$$\max_{\theta} \sum_{i=1}^{m} \log P(y^i | \boldsymbol{x}^i; \boldsymbol{\theta})$$

$$\Leftrightarrow \ \min_{\boldsymbol{\omega}} \sum_{i=1}^{m} \left\|\boldsymbol{\omega}^T \boldsymbol{x}^i - y^i\right\|^2$$

Maximum Likelihood

$$p(\boldsymbol{y}|X, \boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{y}; X\boldsymbol{\omega}, I) \propto \exp\left(-\frac{1}{2}(\boldsymbol{y} - X\boldsymbol{\omega})^T(\boldsymbol{y} - X\boldsymbol{\omega})\right)$$

$$p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \boldsymbol{\mu}_0, \Lambda_0) \propto \exp\left(-\frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\mu}_0)^T \Lambda_0^{-1}(\boldsymbol{\omega} - \boldsymbol{\mu}_0)\right)$$

$$(= \text{diag}(\lambda_0))$$

$$p(\boldsymbol{\omega}|X, \boldsymbol{y}) \propto p(\boldsymbol{y}|X, \boldsymbol{\omega}) p(\boldsymbol{\omega})$$

Weight decay

$$\propto \exp\left(-\frac{1}{2}(\boldsymbol{y} - X\boldsymbol{\omega})^T(\boldsymbol{y} - X\boldsymbol{\omega})\right) \exp\left(-\frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\mu}_0)^T \Lambda_0^{-1}(\boldsymbol{\omega} - \boldsymbol{\mu}_0)\right)$$

$$\propto \exp\left(-\frac{1}{2}(-2\boldsymbol{y}^T X\boldsymbol{\omega} + \boldsymbol{\omega}^T X^T X\boldsymbol{\omega} + \boldsymbol{\omega}^T \Lambda_0^{-1}\boldsymbol{\omega} - 2\boldsymbol{\mu}_0^T \Lambda_0^{-1}\boldsymbol{\omega})\right)$$

$$\propto \exp\left(-\frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\mu}_m)^T \Lambda_m^{-1}(\boldsymbol{\omega} - \boldsymbol{\mu}_m) + \frac{1}{2}\boldsymbol{\mu}_m^T \Lambda_{\text{m}}^{-1}\boldsymbol{\mu}_m\right)$$

$$(\Lambda_{\text{m}} \triangleq (X^T X + \Lambda_0^{-1})^{-1}, \boldsymbol{\mu}_m \triangleq \Lambda_{\text{m}}(X^T \boldsymbol{y} + \Lambda_0^{-1}\boldsymbol{\mu}_0))$$

$$\propto \exp\left(-\frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\mu}_m)^T \Lambda_m^{-1}(\boldsymbol{\omega} - \boldsymbol{\mu}_m)\right)$$

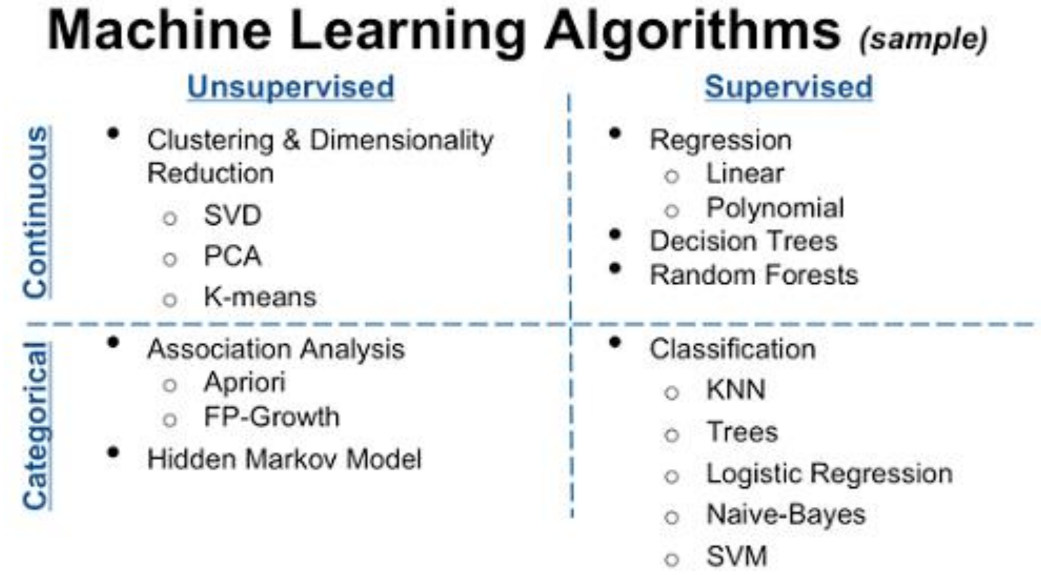
$$\omega_{MAP} = \arg\max_{\boldsymbol{\omega}} p(\boldsymbol{\omega}|X, \boldsymbol{y}) = \arg\max_{\boldsymbol{\omega}} p(\boldsymbol{y}|X, \boldsymbol{\omega}) + \log p(\boldsymbol{\omega})$$

Maximum A Posteri (MAP) Estimation
(maximize Likelihood + Prior)

Machine Learning Algorithm

- Supervised: given training set of examples of inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$
- Unsupervised: no outputs $\boldsymbol{y}$

- Probabilistic Supervised Learning:
  - Maximum likelihood estimation ($\max_{\theta} p(y|x;\theta)$)
  - For classification problem, we need further techniques to handle discrete value
    - E.g., $p(y=1|x;\theta) = \sigma(\theta^T x)$ (logistic regression)

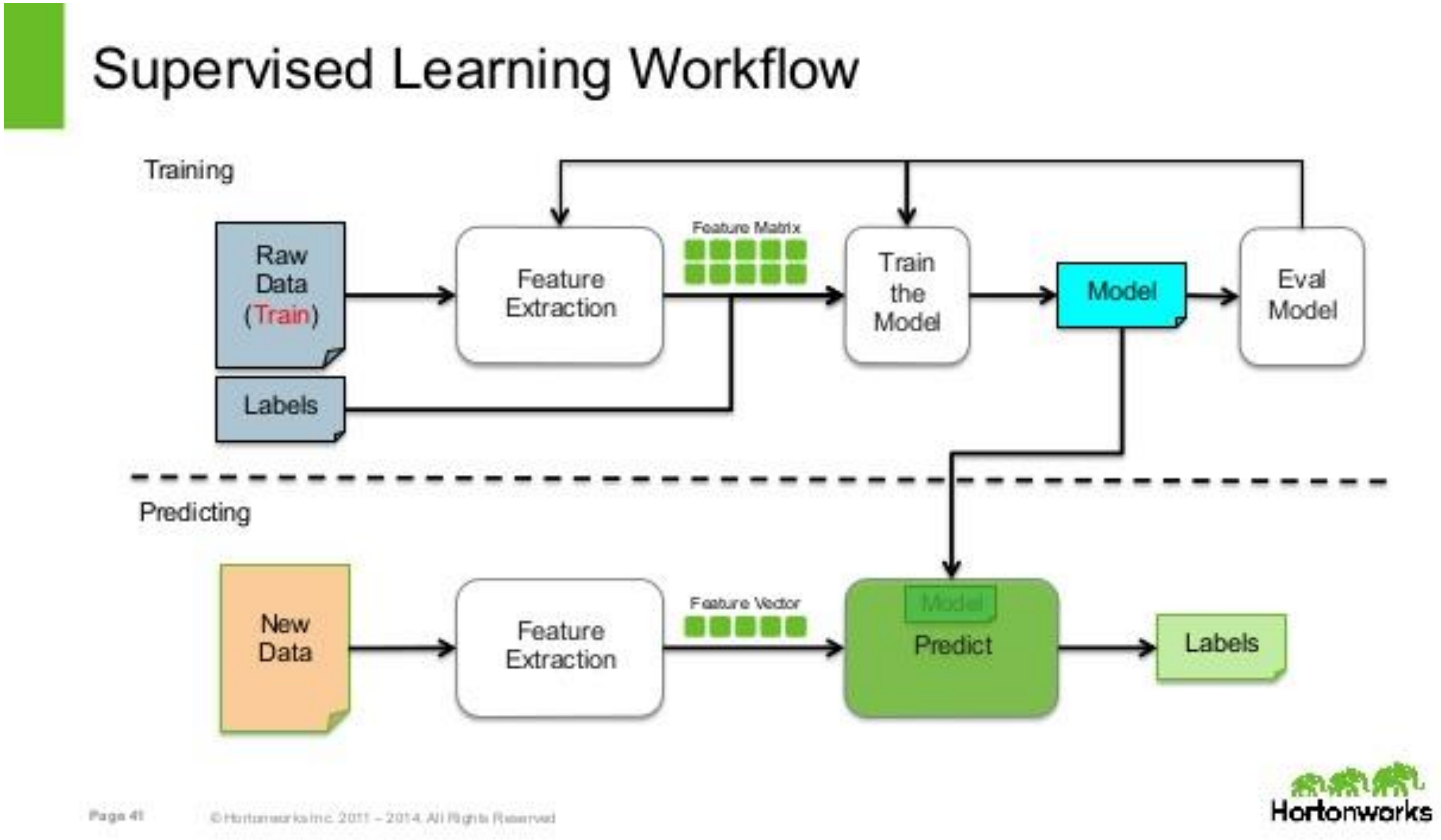


Machine Learning Algorithms *(sample)*

	Unsupervised	Supervised
Continuous	• Clustering & Dimensionality Reduction ◦ SVD ◦ PCA ◦ K-means	• Regression ◦ Linear ◦ Polynomial • Decision Trees • Random Forests
Categorical	• Association Analysis ◦ Apriori ◦ FP-Growth • Hidden Markov Model	• Classification ◦ KNN ◦ Trees ◦ Logistic Regression ◦ Naive-Bayes ◦ SVM

http://www.differencebetween.net/technology/differences-between-supervised-learning-and-unsupervised-learning/

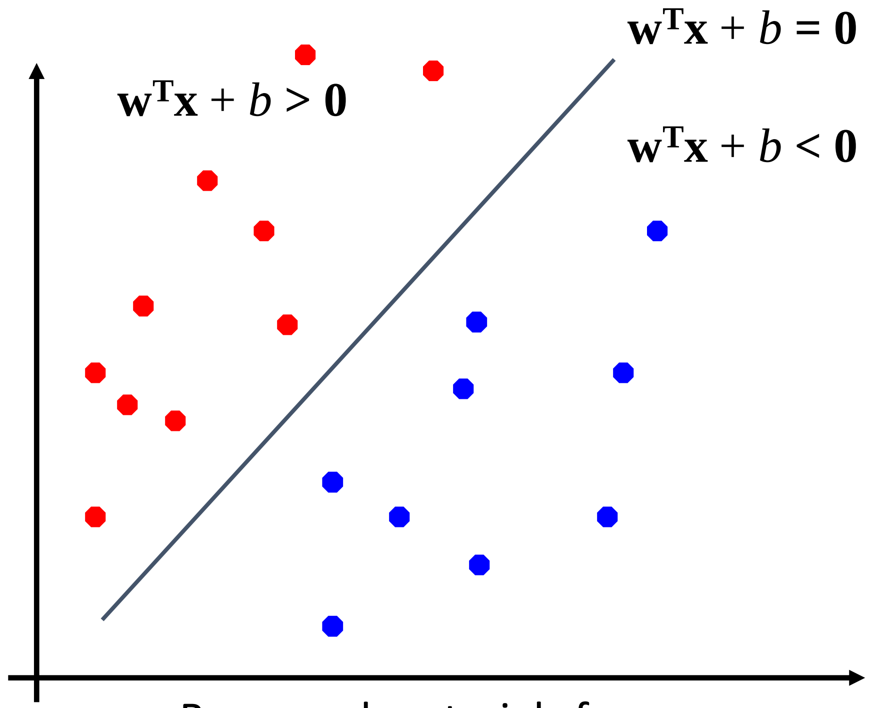
Workflow of Supervised Learning



https://en.proft.me/2015/12/24/types-machine-learning-algorithms/

Linear Binary Classification Problem

- Binary classification can be viewed as the task of separating classes in feature space:

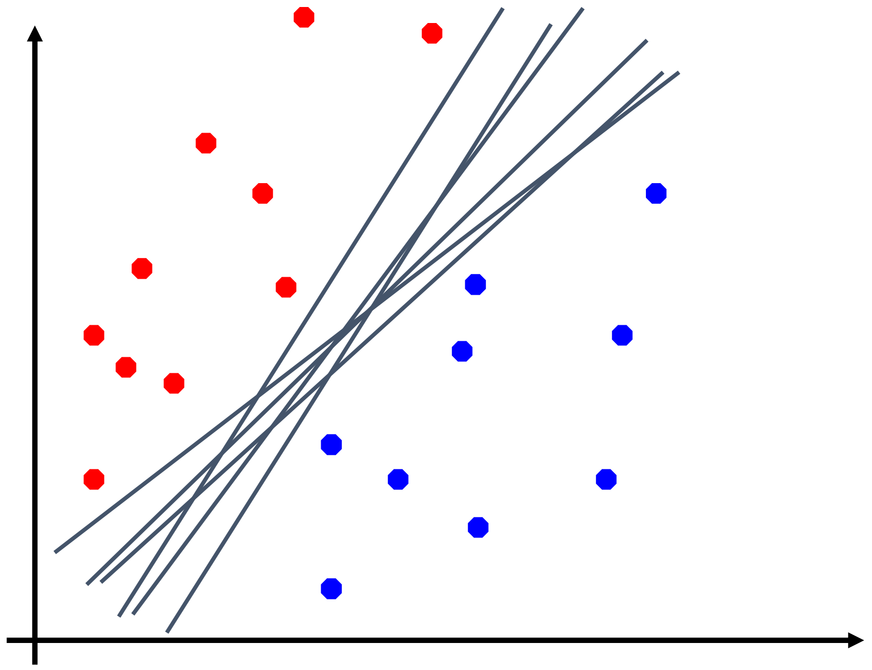

$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = \mathbf{0}$
$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b > \mathbf{0}$
$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b < \mathbf{0}$

$f(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$

Borrowed materials from
https://www.cs.utexas.edu/~mooney/cs391L/slides/svm.ppt

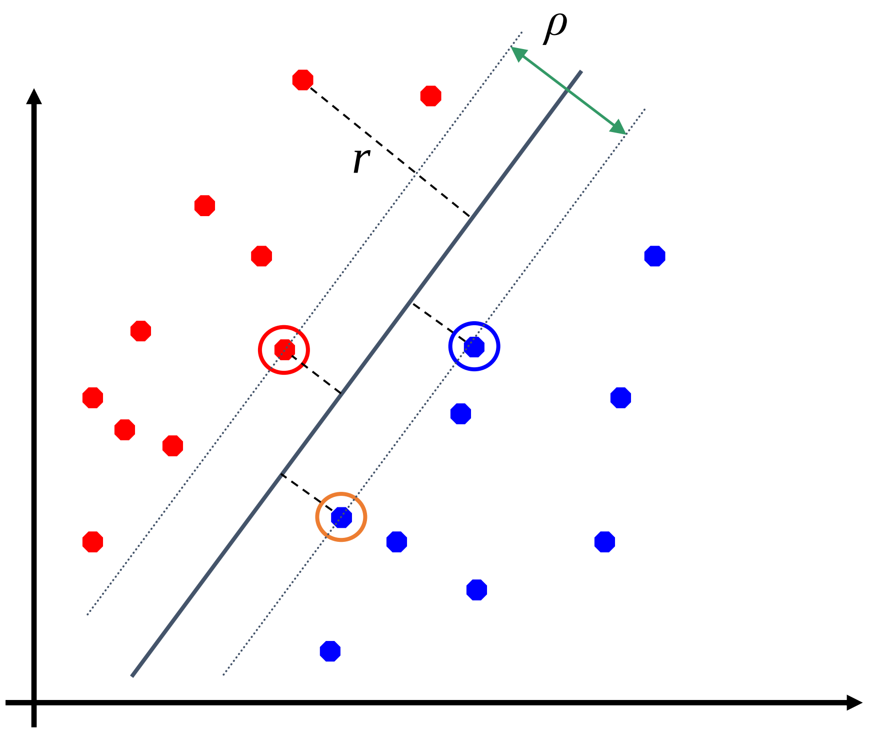
Linear Separators

- Which of the linear separators is optimal?



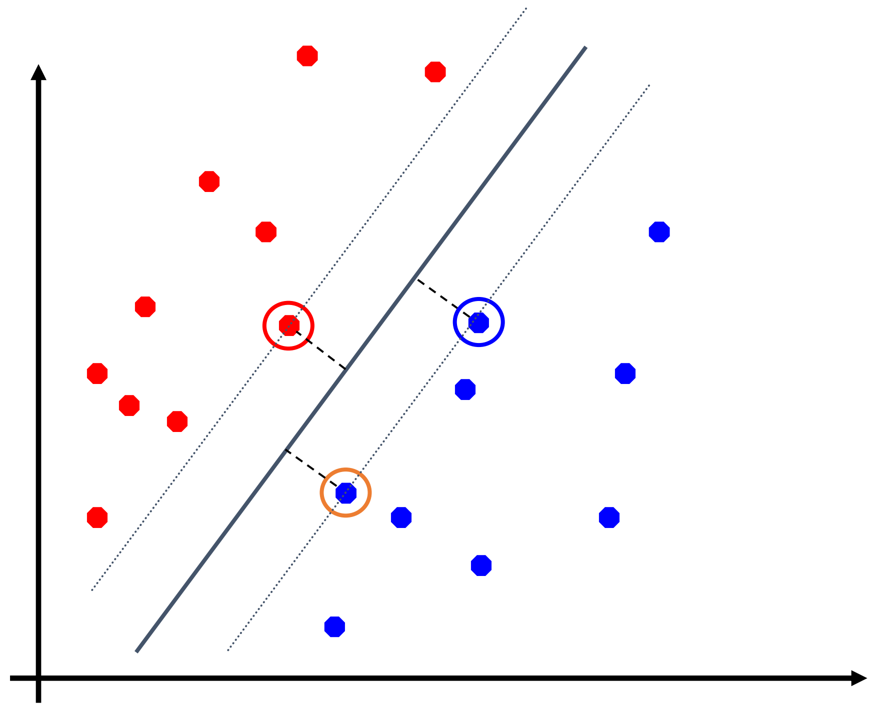
Classification Margin

- Distance from example $\boldsymbol{x}$ to the separator is $r = \dfrac{\boldsymbol{\omega}^T \boldsymbol{x} + b}{\|\boldsymbol{\omega}\|}$
- Examples closest to the hyperplane are ***support vectors***
- ***Margin*** ρ of the separator is the distance between support vectors.



Maximum Margin Classification

- Maximizing the margin is good according to intuition and probably approximately correct learning (PAC) theory
- Implies that only support vectors matter; other training examples are ignorable



Support Vector Machine (Vapnik1995)

- Let training set $\{\boldsymbol{x}_i, y_i\}, y_i \in \{-1,1\}$ be separated by a hyperplane with margin ρ. Then for each training example $\{\boldsymbol{x}_i, y_i\}$:

$$\begin{aligned} \boldsymbol{\omega}^T\boldsymbol{x}_i + b &\le -\frac{\rho}{2} \ if\ y_i = -1 \\ \boldsymbol{\omega}^T\boldsymbol{x}_i + b &\ge \frac{\rho}{2} \ if\ y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i(\boldsymbol{\omega}^T\boldsymbol{x}_i + b) \ge \frac{\rho}{2}$$

- For every support vector $\boldsymbol{x}_s$ the above inequality is an equality. After rescaling $\boldsymbol{\omega}$ and $\boldsymbol{b}$ by $\rho/2$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is

$$r = \frac{y_s(\boldsymbol{\omega}^T\boldsymbol{x_s} + b)}{\|\boldsymbol{\omega}\|} = \frac{1}{\|\boldsymbol{\omega}\|}$$

- Then the margin can be expressed through (rescaled) **w** and b as:

$$\rho = 2r = \frac{2}{\|\boldsymbol{\omega}\|}$$

Linear Support Vector Machine (cont.)

- We can formulate the *quadratic optimization problem:*

$$\boxed{\begin{array}{l}\text{Find } \boldsymbol{\omega} \text{ and } b \text{ such that the margin } \rho \text{ is maximized} \\ \text{s.t., for all } \{\boldsymbol{x}_i, y_i\} : y_\mathrm{i}(\boldsymbol{\omega}^T \boldsymbol{x}_i + b) \geq 1\end{array}}$$

- Which can be reformulated as:

$$\boxed{\begin{array}{l}\text{Find } \boldsymbol{\omega} \text{ and } b \text{ such that } \boldsymbol{\phi}(\boldsymbol{\omega}) = \frac{\mathbf{1}}{\mathbf{2}} \|\boldsymbol{\omega}\|^{\mathbf{2}} \text{ is minimized} \\ \text{s.t., for all } \{\boldsymbol{x}_i, y_i\} : y_\mathrm{i}(\boldsymbol{\omega}^T \boldsymbol{x}_i + b) \geq 1\end{array}}$$

Need to optimize a *quadratic* function subject to *linear* constraints.

- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every inequality constraint in the primal (original) problem

Solution to Linear SVM

- We can convert the constrained minimization to an unconstrained problem by KKT Lagrange multiplier as

$$\min_{\omega,b} \frac{1}{2}\|\boldsymbol{\omega}\|^{2} + \sum_{i=1}^{n} \max_{\alpha_i \geq 0} \alpha_i(1 - y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b))$$

$$\min_{\omega,b} \max_{\alpha_i \geq 0} \frac{1}{2}\|\boldsymbol{\omega}\|^{2} + \sum_{i=1}^{n} \alpha_i(1 - y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b))$$

$$\max_{\alpha_i \geq 0} \min_{\omega,b} \underbrace{\frac{1}{2}\|\boldsymbol{\omega}\|^{2} + \sum_{i=1}^{n} \alpha_i(1 - y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b))}_{J(\boldsymbol{\omega}, b; \boldsymbol{\alpha})}$$

- We first minimize J w.r.t $\{\boldsymbol{\omega}, \mathrm{b}\}$ for any fixed setting of α

Solution to Linear SVM

$$\begin{cases} \dfrac{\partial}{\partial \boldsymbol{\omega}} J(\boldsymbol{\omega}, b; \boldsymbol{\alpha}) = \boldsymbol{\omega} - \displaystyle\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = 0 \\ \dfrac{\partial}{\partial b} J(\boldsymbol{\omega}, b; \boldsymbol{\alpha}) = -\displaystyle\sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

$$\max_{\alpha_i \geq 0} \min_{\omega, b} \frac{1}{2} \|\boldsymbol{\omega}\|^{\mathbf{2}} + \sum_{i=1}^{n} \alpha_i (1 - y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b))$$

$$\Rightarrow \max_{\substack{\alpha_i \geq 0 \\ \sum_i \alpha_i y_i = 0}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \left(\boldsymbol{x}_i^T \boldsymbol{x}_j \right) \text{ (Dual form)}$$

Quadratic programming problem

- Prediction on a new example is the sign of:

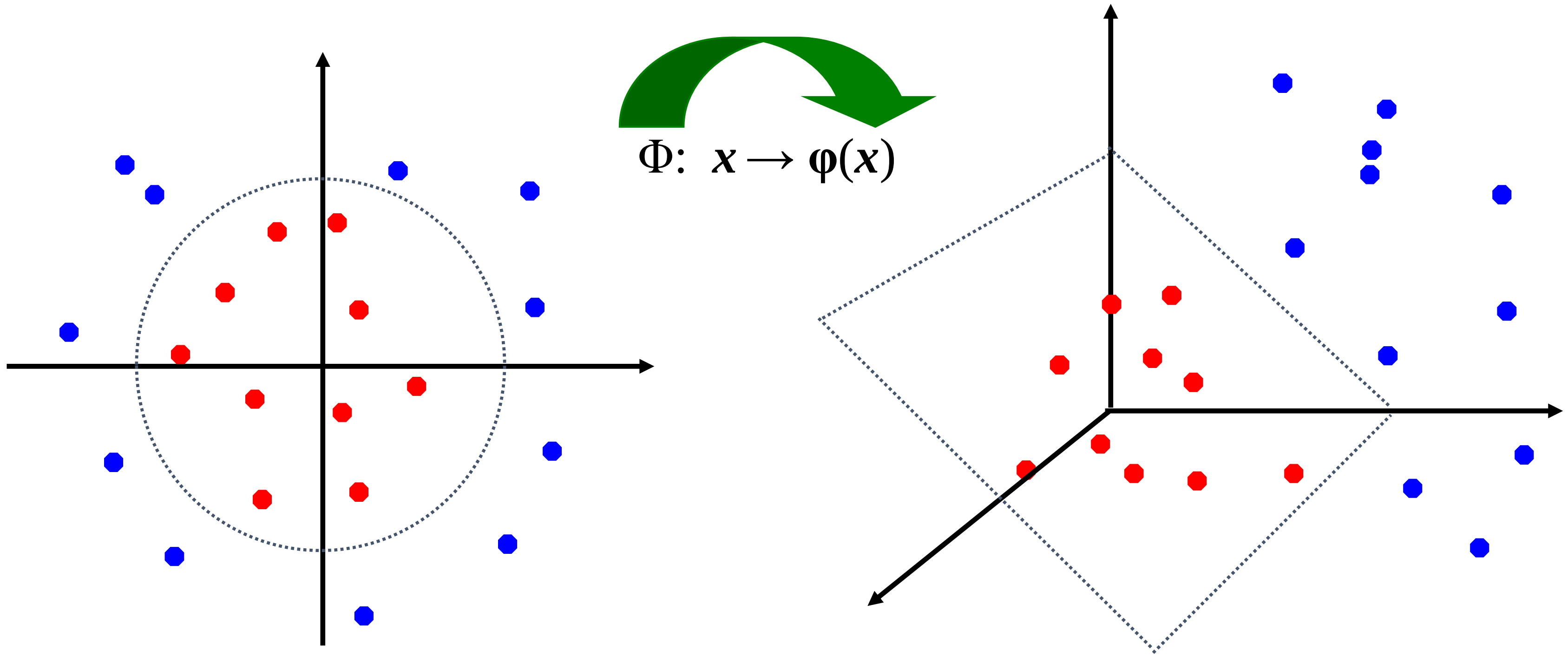
$$b + w^T x = b + \boldsymbol{x}^T \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = b + \sum_{i \in SV} \alpha_i y_i (\boldsymbol{x}^T \boldsymbol{x}_i)$$

Why solving the Dual Problem?

- KKT theorem: the solution we find here will be the same as the solution to the original problem
- Q: But why are we doing this? (why not just solve the original problem?)
- Ans: Because this will let us solve the problem by computing just the inner products of $\boldsymbol{x}_i^T \boldsymbol{x}_j$ (which will be very important later on when we want to solve non-linearly separable classification problems)

Nonlinear SVM

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



Φ: $\boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$

Kernel Trick

$$\max_{\substack{\alpha_i \geq 0 \\ \sum_i \alpha_i y_i = 0}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \left(\boldsymbol{x}_i^T \boldsymbol{x}_j \right) \text{ (Dual form)}$$

$$\Downarrow \qquad k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

$$\max_{\substack{\alpha_i \geq 0 \\ \sum_i \alpha_i y_i = 0}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ (Dual form)}$$

- If there is a ”kernel function” k such that $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \phi(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j)$ then we do not need to know or compute $\boldsymbol{\phi}$. That is, the kernel function defines inner products in the transformed space.

Example: let $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left(1 + \boldsymbol{x}_i^T \boldsymbol{x}_j\right)^2 \qquad \boldsymbol{x} \in \mathbb{R}^{\mathbf{2}}$

$$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left(1 + \boldsymbol{x}_i^T \boldsymbol{x}_j\right)^2 = \left[1\ x_{i1}^2\ \sqrt{2} x_{i1} x_{i2}\ x_{i2}^2 \sqrt{2} x_{i1} \sqrt{2} x_{i2}\right]^T \left[1\ x_{j1}^2\ \sqrt{2} x_{j1} x_{j2}\ x_{j2}^2 \sqrt{2} x_{j1} \sqrt{2} x_{j2}\right]$$

$$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \phi(\boldsymbol{x}_i)^T \phi\left(\boldsymbol{x}_j\right) \, where \; \phi(\boldsymbol{x}) = \left[1\ \ x_1^2\ \ \sqrt{2} x_1 x_2\ \ x_2^2\ \ \sqrt{2} x_1\ \ \sqrt{2} x_2\right]$$

Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each **φ**(**x**) explicitly).

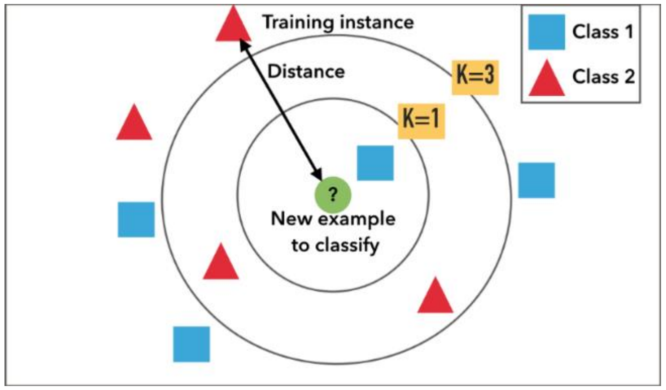
Popular Kernel Function

$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(x_i^T x_j + 1\right)^{\mathrm{p}}$	Polynomial function
$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp \dfrac{\left\Vert x_i - x_j \right\Vert}{2\sigma^2}$	Radial basis function
$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(k x_i^T x_j - \delta)$	Sigmoid function

Other Supervised Learning Algorithms

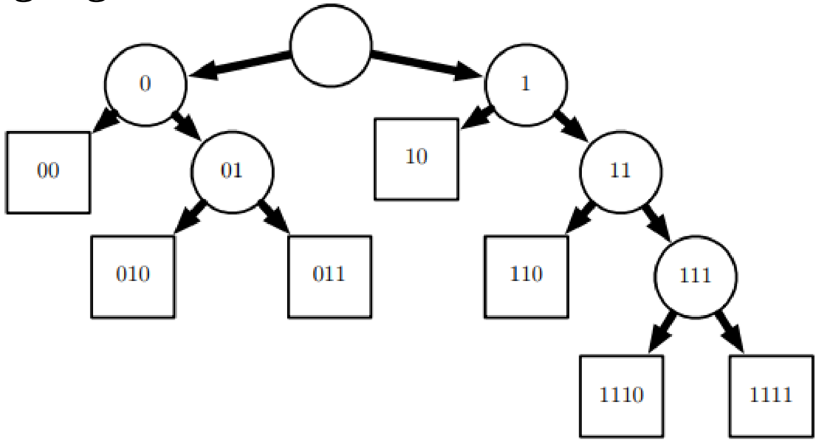
■ ***k*-Nearest Neighbor** (NN) Regression

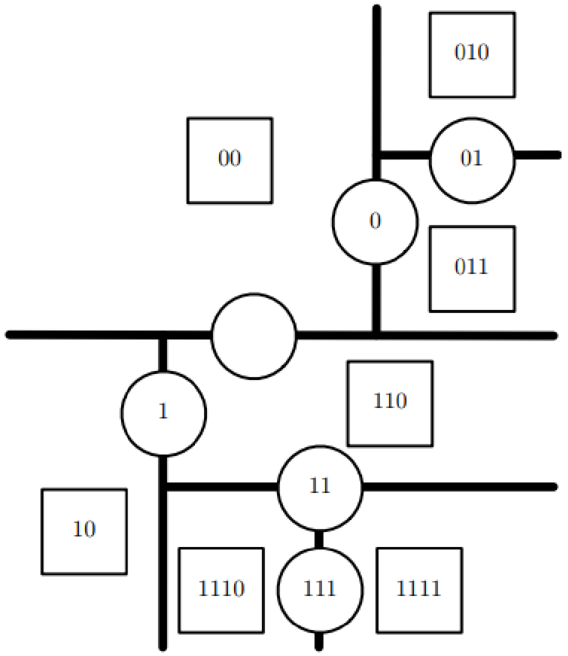
- We find the k-nearest neighbors to ***x*** in the training data *X*. We then return the average of the corresponding y values in the training set
- Very high capacity (larger data, better accuracy)
- Cannot be generalized (learning nothing from data)



■ ***Decision Tree*** (Breiman et al., 1984)

- Each "Question" divide the internal non-overlappng regions





Unsupervised Learning Algorithms

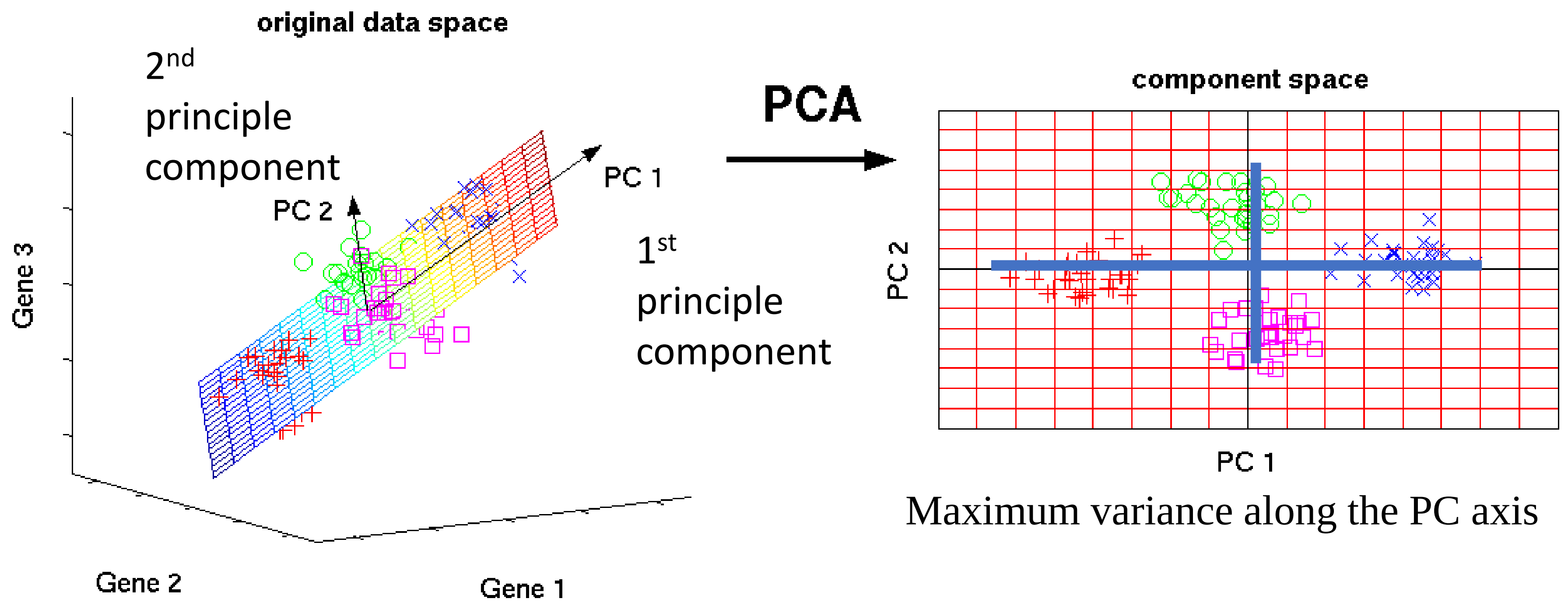
- A classical supervised learning task is to find the “best” representation on of the data

- “Best” is often the simpler representation
  - ***Low dimensional representation***
    - compresses as much information about *x* as possible in a smaller representation (e.g., PCA)
  - ***Sparse representation***
    - embeds the dataset into a representation whose entries are mostly zero for most inputs (e.g., sparse coding)
  - ***Independent representation***
    - attempts to disentangle the sources of variation underlying the data distribution such that the dimensions of the representation are statistically independent (e.g., PCA)

Unsupervised Learning as Clustering

Principle Component Analysis (1)

- ***Principle Component Analysis*** (PCA) is a technique to emphasize variation and bring out strong patterns in data
- Criteria: lower dimension and independent



Maximum variance along the PC axis

http://www.nlpca.org/pca-principal-component-analysis-matlab.html

Principle Component Analysis (2)

- Design matrix: $X \in R^{m \times n}$; $\mathbb{E}[\boldsymbol{x}] = 0$ (If not, normalize the matrix)
- The unbiased sample covariance matrix
  - $\mathrm{Var}[\boldsymbol{x}] = 1/(m-1)X^T X$ (since $\mathbb{E}[\boldsymbol{x}] = 0$)

- PCA finds a representation $\boldsymbol{z} = W^T \boldsymbol{x}$, where $\mathrm{Var}[\boldsymbol{z}]$ is diagonal

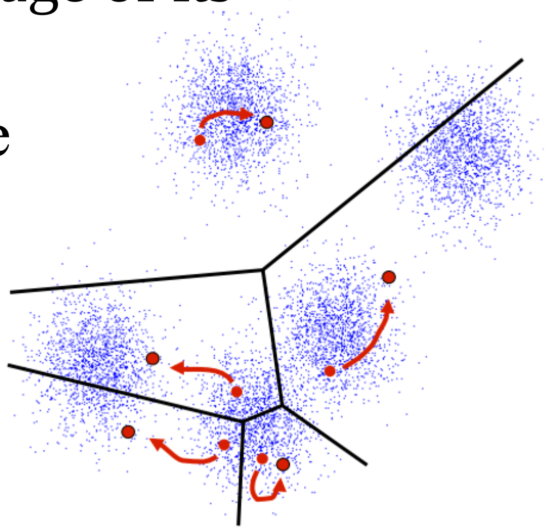
- *Principle component of X is given by the SVD* ($\mathrm{X} = U\Sigma W^T$)
  - $X^T X = (U\Sigma W^T)^T U\Sigma W^T = W\Sigma^2 \mathrm{W}^\mathrm{T}$
  - $\mathrm{Var}[\boldsymbol{x}] = \frac{1}{(m-1)} X^T X = \frac{1}{(m-1)} W\Sigma^2 \mathrm{W}^\mathrm{T}$
  - $\mathrm{Var}[\boldsymbol{z}] = \frac{1}{(m-1)} W^T X^T X W = \frac{1}{(m-1)} \Sigma^2$ (diagonal matrix)

- When we project the data *x* to *z* via the linear transformation *W*, the resulting representation has a diagonal covariance matrix, which immediately implies that the individual elements of *z* are mutually uncorrelated (independent representation)

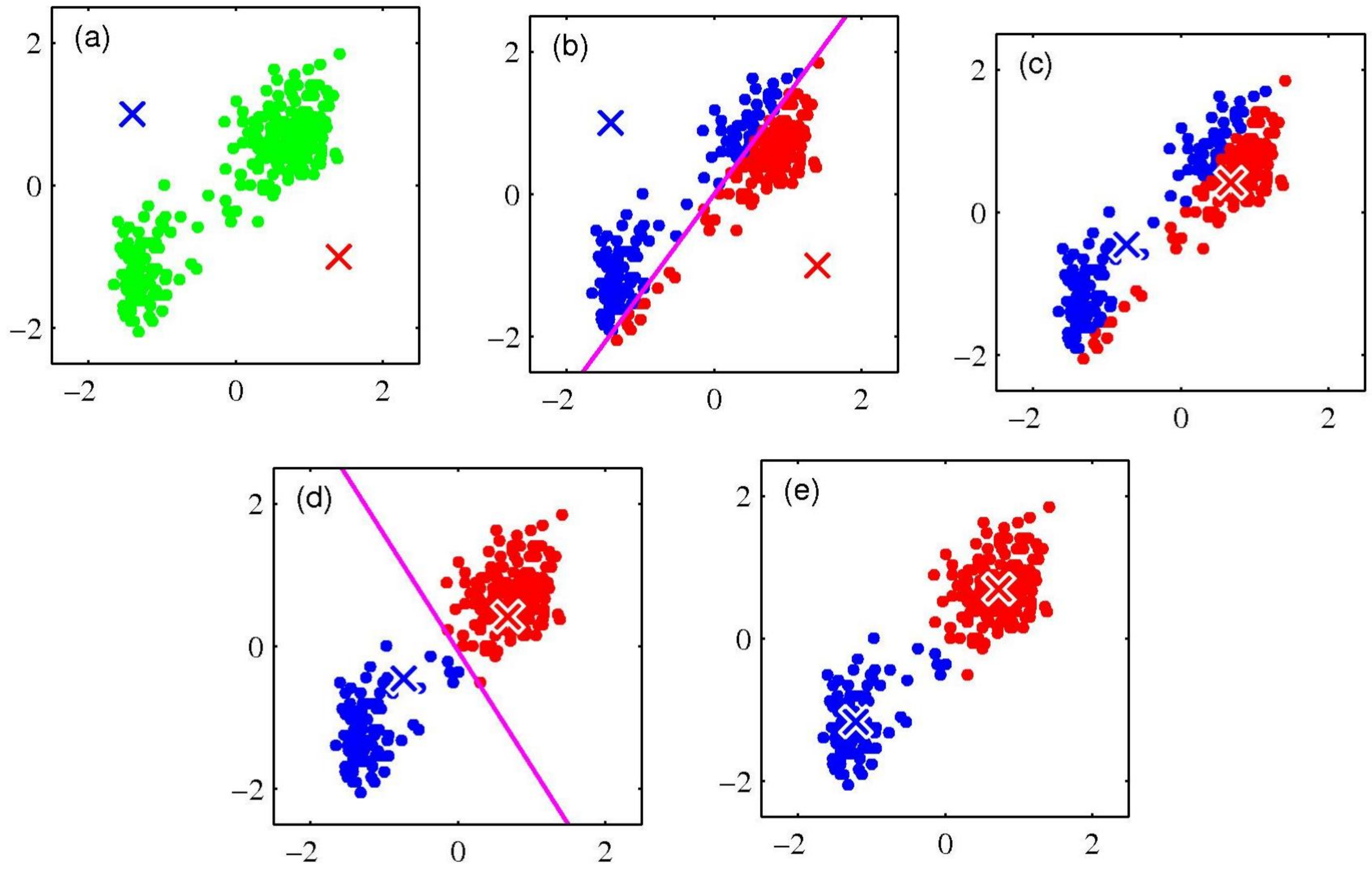
K-means Clustering (1)

- Divide the feature space into *K* clusters
  - For image clustering, the feature vector can be average color, RGB histogram, Bag-of-Visual-Words and so on

- An iterative clustering algorithm
  - Initialize: Pick *K* random cluster centers (mean vector)
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignment change



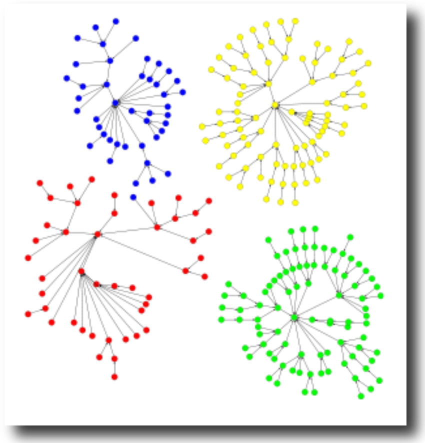
K-means Clustering (2)

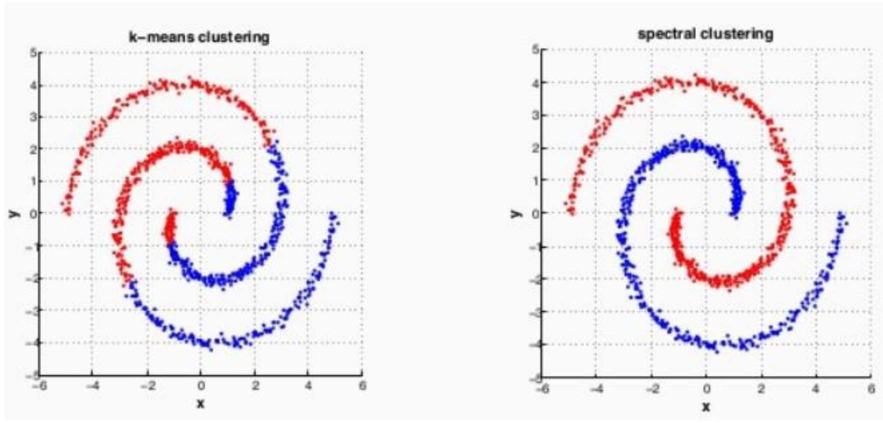


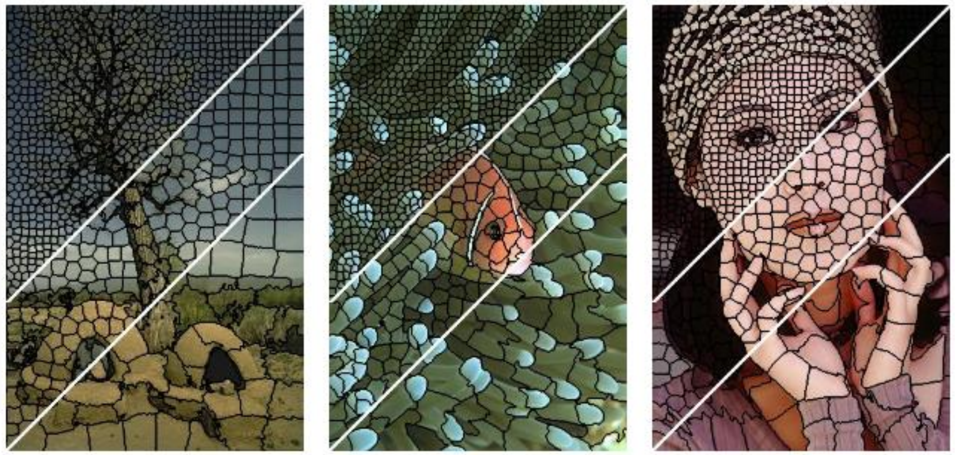
Borrowed from http://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

Other Unsupervised Clustering Algorithms

- K-medoids (Assign cluster centers from samples in data)
- Mean-shift (Adaptively choose the number of clusters)
- Spectral Clustering (Clustering on the graph)
- Gaussian Mixture Model (GMM)
- Affinity propagation (Give hierarchical structure)
- And so on…







Stochastic Gradient Descent

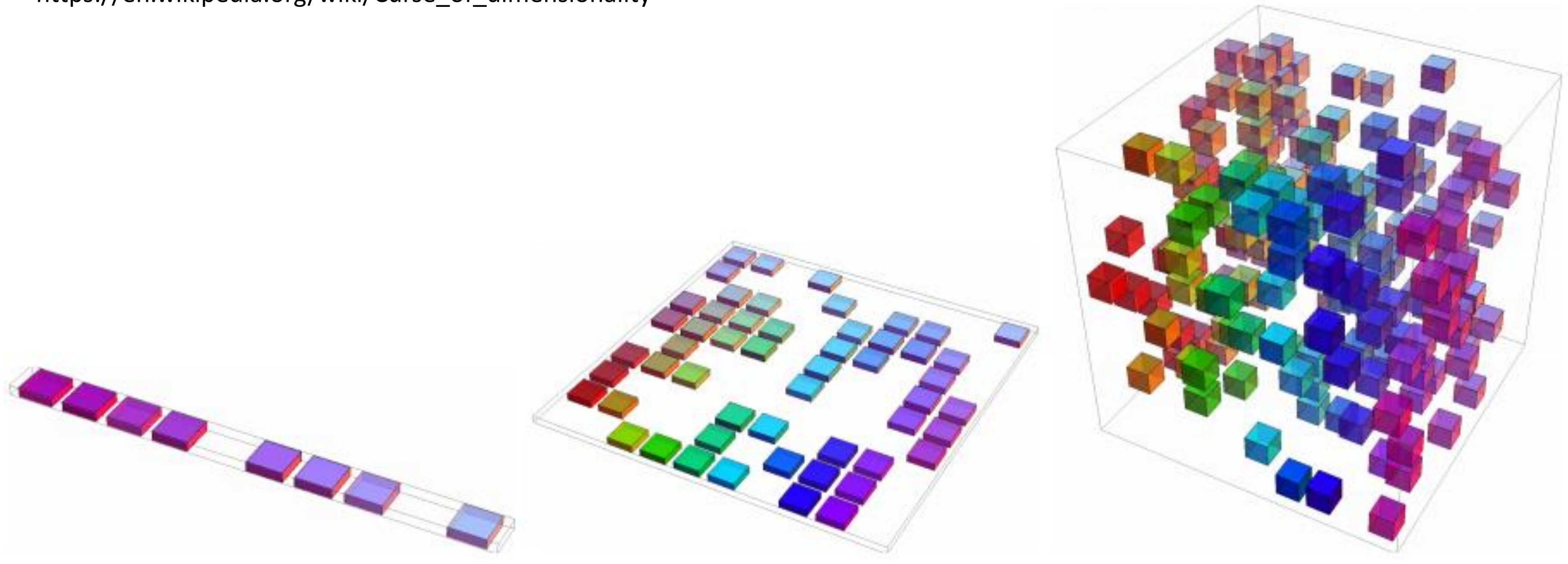
- If we compute derivatives for all samples, it costs O(*m*)
    - $J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},y\sim\tilde{p}_{\text{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_i^m L(\boldsymbol{x}^i, y^i, \boldsymbol{\theta})$
    - $\nabla_\theta J(\boldsymbol{\theta}) = \frac{1}{m}\sum_i^m \nabla_\theta L(\boldsymbol{x}^i, y^i, \boldsymbol{\theta})$

- Instead, we can sample ***minibatch*** of examples $\mathbb{B} = \{x^1, \dots, x^{m'}\}$
    - $\nabla_\theta J(\boldsymbol{\theta}) = \frac{1}{m'}\sum_i^{m'} \nabla_\theta L(\boldsymbol{x}^i, y^i, \boldsymbol{\theta})$
- Pros.
    - Good for the large scale training
    - If *m* goes infinity, the function converges to minimum
- Cons. (We will learn the "better" algorithm later)
    - Difficult to set "best" learning rate
    - The preprocessing is necessary (e.g., normalization of data)

The Curse of Dimensionality

The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces. The expression was coined by Richard E. Bellman when considering problems in dynamic optimization. The common theme of these problems is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. This sparsity is problematic for any method that requires statistical significance. In order to obtain a statistically sound and reliable result, the amount of data needed to support the result often grows exponentially with the dimensionality.

https://en.wikipedia.org/wiki/Curse_of_dimensionality

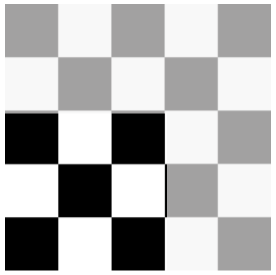


Local Constancy and Smoothness Regularization

- Smoothness prior or local constancy prior states that the function we learn should not change much within a small region $f(\boldsymbol{x}) \approx f(\boldsymbol{x} + \epsilon)$ (e.g., k-nearest neighbors classifier)

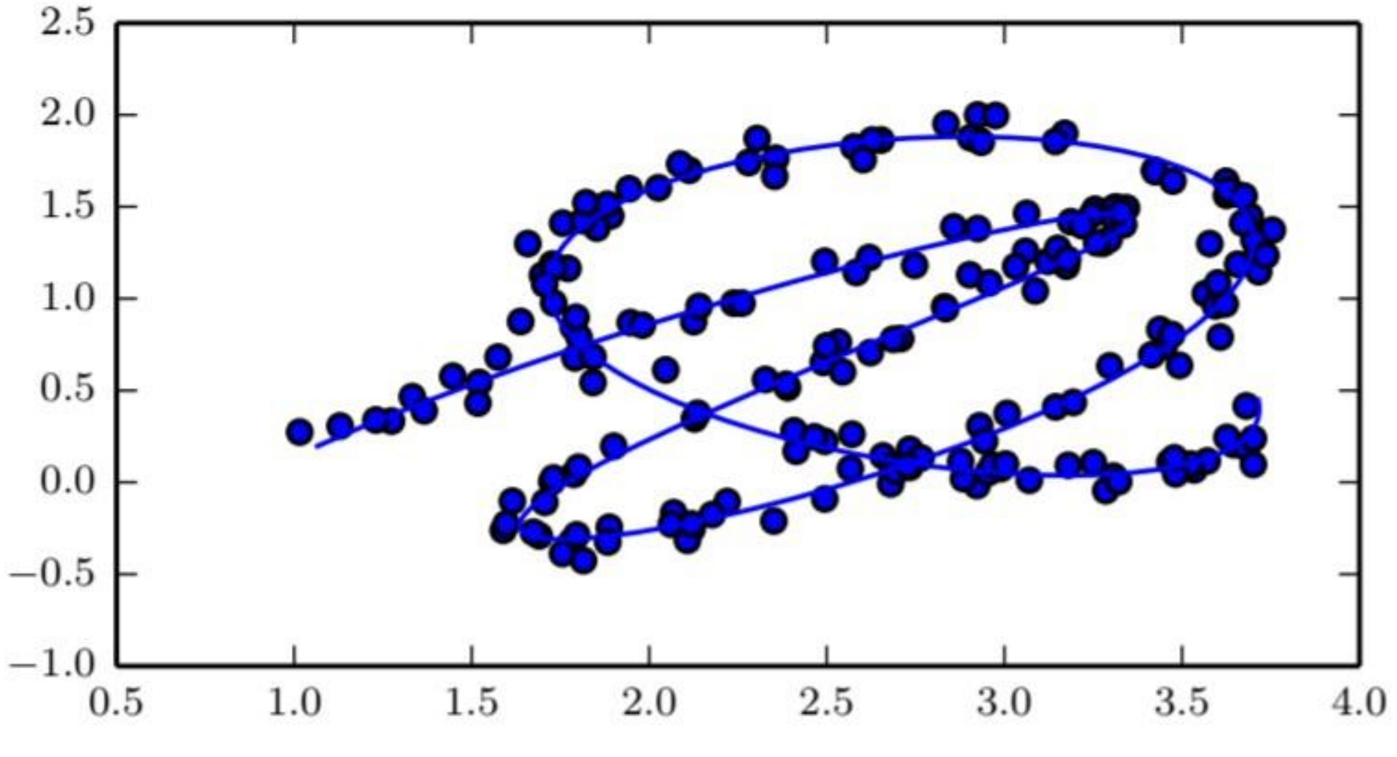
- Smoothness prior itself is not sufficient to represent a complex function that has many more regions to be distinguished than the number of training examples (e.g., to extend the repetitive pattern in the space such as a checker board).

- Data-specific or task specific assumption (knowledge) may help to do better learning (e.g., the data could be *decomposed* into factors)



Manifold Learning (1)

- In machine learning, manifold is a connected set of points that can be approximated well by considering only a small number of degrees of freedom, or dimensions, embedded in a higher-dimensional space.



Manifold Learning (2)

- Manifold learning algorithms assume that the data is lying on a low-dimensional manifold and represent the data in terms of the coordinate of the manifold.
- Given a training example of human faces, e.g., variational autoencoder learns the 2-D coordinate of systems (Chapter20)