Big Data Analytics & Applications

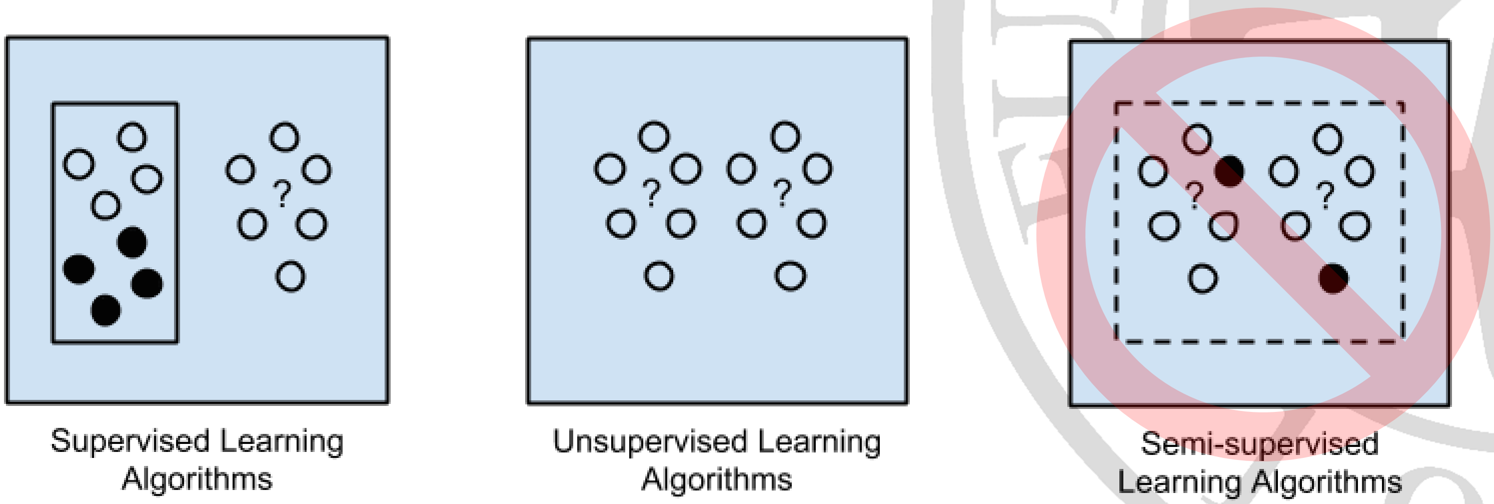
Bin Li

School of Computer Science

Fudan University

ML Problems in BDA

- In BDA, there are two commonly seen ML problem settings
  - Supervised learning: Classification, Regression
  - Unsupervised learning: Clustering, Dimensionality Reduction



[1] https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Classification Problem

- Inputs:
  - Instances: $x_1, x_2, \dots \in X$, where X is feature space
  - Class labels: $y_1, y_2, \dots \in Y$, where $Y = \{1,2,\dots,L\}$ is label set
- Classification problem setting:
  - Training data: $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
  - Test data: $\{(x', ?), (x'', ?), \dots, \}$
- Objective: Learn a function $f: X \rightarrow Y$ on the training data such that $f(x_N) = y_n$ for $n = 1,2,\dots,N$

$$\min_f \frac{1}{N} \sum_{n=1}^{N} classification\_loss(f(x_n), y_n)$$

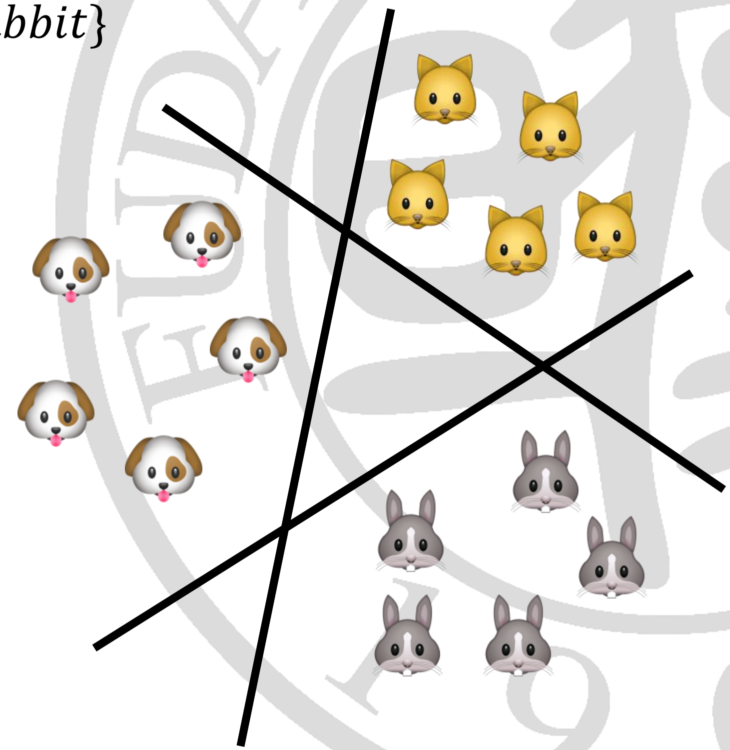
Classification Problem

- Supervised learning example: Image classification
  - Instances: $x_1, x_2, \ldots$ are images
  - Class labels: $y_1, y_2, \ldots \in \{dog, cat, rabbit\}$

$f($🐶$) = "dog"$

$f($🐱$) = "cat"$

$f($🐰$) = "rabbit"$



Classification Problem

- Application provides the dataset $\{(x_1, y_1), \ldots, (x_N, y_N)\}$
  - Image annotation: x_n pixel image
  - Document categorization: x_n bag-of-words representation
  - User classification: x_n user profile

- $f: X \to Y$ is a selected model for certain applications
  - Nearest Neighbors Classifier
  - Linear Classifier
  - Logistic Regression
  - Support Vector Machine
  - Multilayer Perceptron
  - Decision Tree

Regression Problem

- Inputs:
  - Instances: $x_1, x_2, \ldots \in X$, where X is feature space
  - Targets: $y_1, y_2, \ldots \in Y$, where $Y \subset \mathrm{R}^m$ is target domain
- Regression problem setting:
  - Training data: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
  - Test data: $\{(x', ?), (x'', ?), \ldots,\}$

- Objective: Learn a function $f: X \to Y$ on the training data such that $f(x_N) = y_n$ for $n = 1,2, \ldots, N$

$$\min_f \frac{1}{N} \sum_{n=1}^{N} regression\_loss(f(x_n), y_n)$$

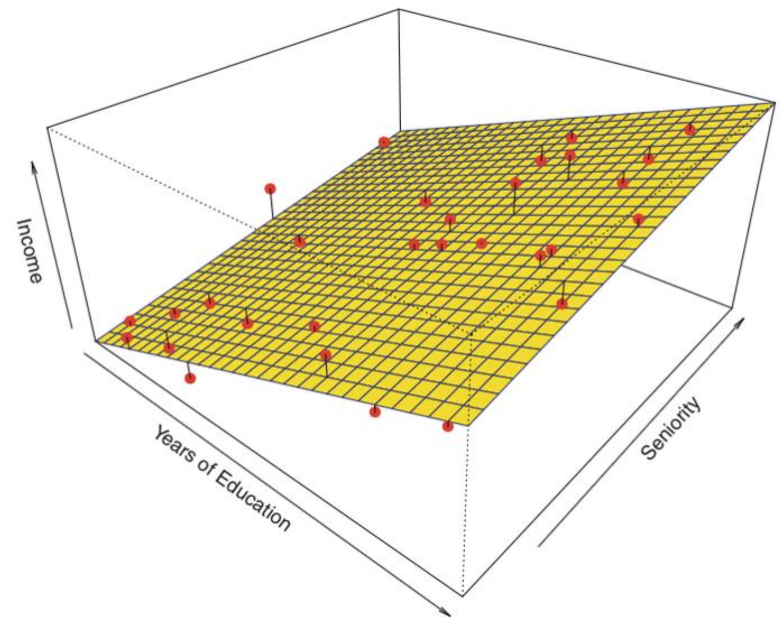
Regression Problem

- Supervised learning example: Income Prediction
  - Instances: $x_1, x_2, \ldots \in Education \times Seniority$
  - Targets: $y_1, y_2, \ldots \in Income$

$$f(12,3) = 50K$$

$$f(16,5) = 80K$$

$$f(19,8) = 200K$$



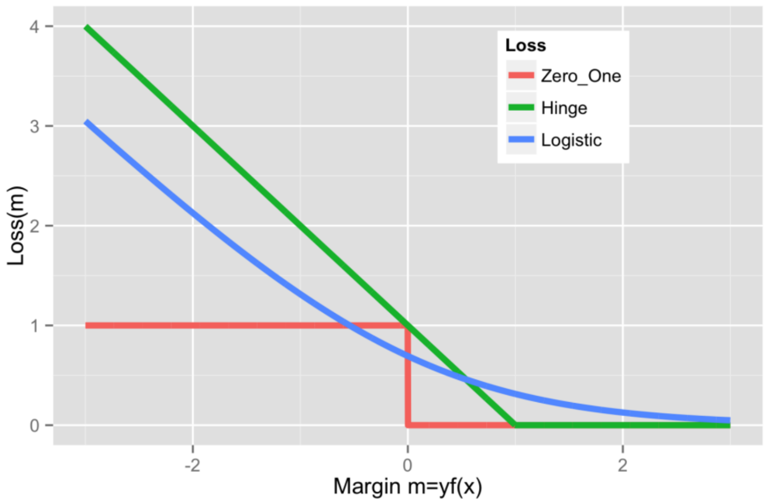
Regression Problem

- Application provides the dataset $\{(x_1, y_1), \dots, (x_N, y_N)\}$
  - Stock trend prediction: x_n variates, y_n index
  - Location prediction: x_n previous locations, y_n new location
  - Rating prediction: x_n user profile, y_n rating

- $f: X \to Y$ is a selected model for certain applications
  - Nearest Neighbors Regression
  - Linear Regression
  - Support Vector Regression
  - Gaussian Process Regression
  - Multi-Layer Perceptron
  - Decision Tree

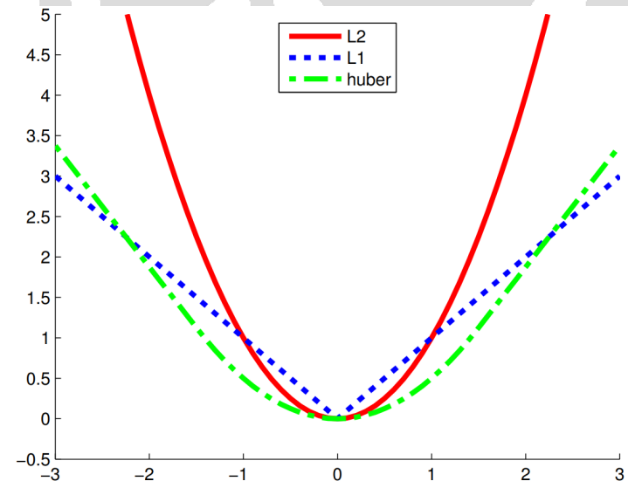
Classification vs Regression

- It seems that classification and regression are similar
- What makes them different? – Loss ※



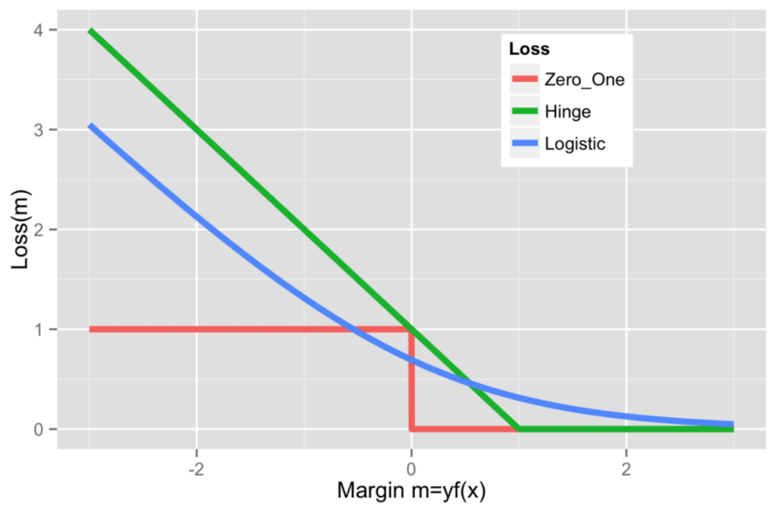
$$\min_f \frac{1}{N}\sum_{n=1}^{N} \textit{classification\_loss}(f(x_n), y_n)$$

$$\min_f \frac{1}{N}\sum_{n=1}^{N} \textit{regression\_loss}(f(x_n), y_n)$$



Classification Loss

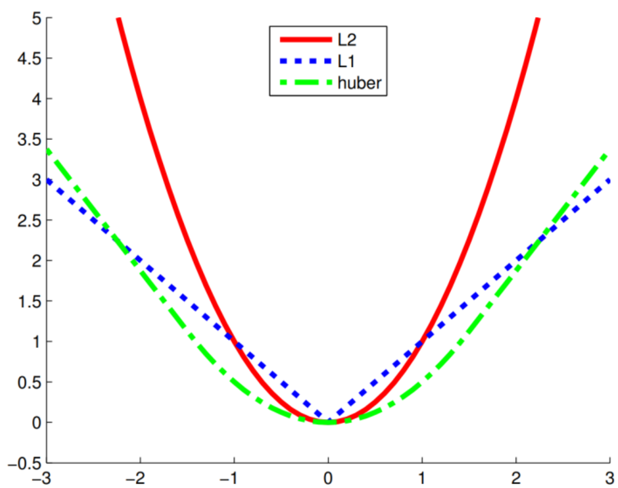
- Zero-One Loss: $l_{0-1}(f(x_n), y_n) = 1(y_n f(x_n) \leq 0)$
- Hinge Loss: $l_{hinge}(f(x_n), y_n) = \max(1 - y_n f(x_n), 0)$
- Logistic Loss: $l_{logistic}(f(x_n), y_n) = \ln(1 + exp(-y_n f(x_n)))$



$$\min_f \frac{1}{N} \sum_{n=1}^{N} \color{red}{classification\_loss}(f(x_n), y_n)$$

Regression Loss

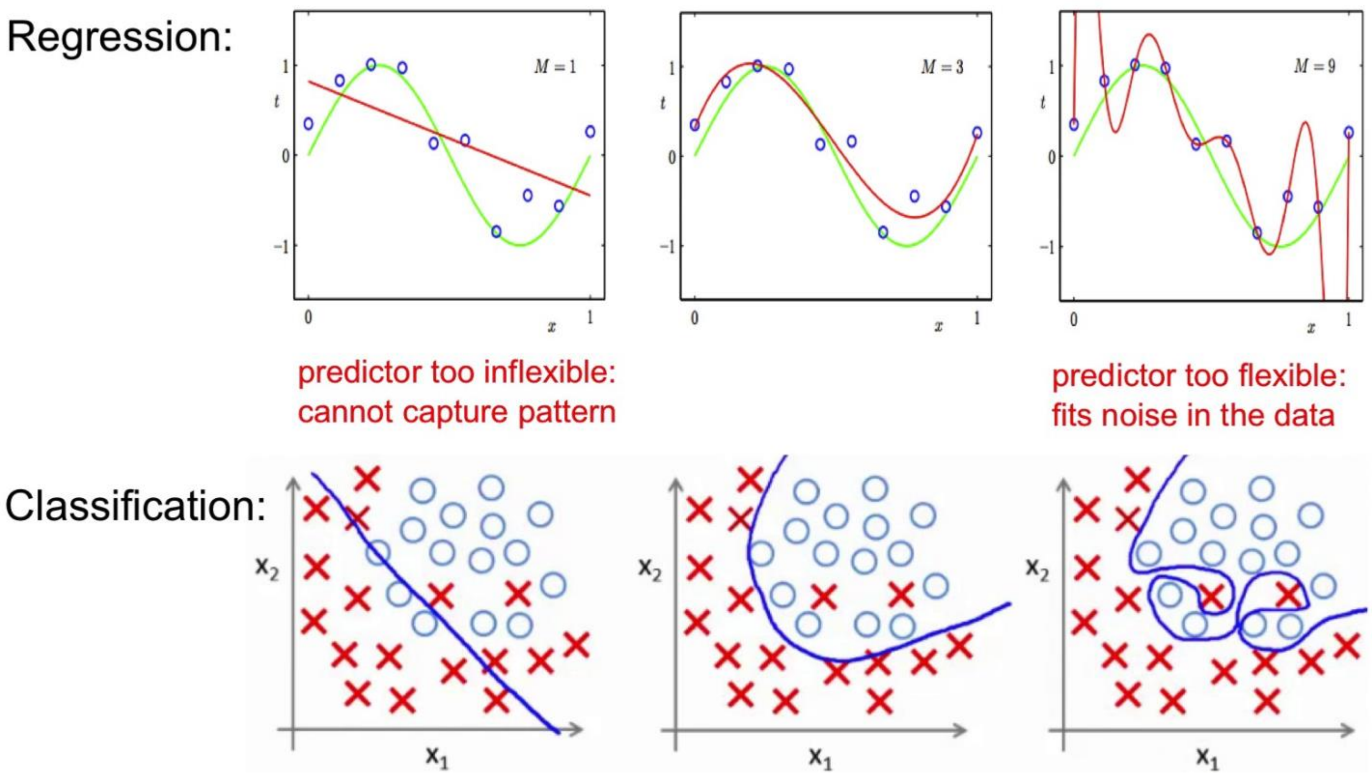
- Quadratic Loss ($L2$-Loss): $l_{L2}(f(x_n), y_n) = \left(y_n - f(x_n)\right)^2$
- Absolute Loss ($L1$-Loss): $l_{L1}(f(x_n), y_n) = |y_n - f(x_n)|$
- Huber Loss: $l_{Huber}(f(x_n), y_n) = \begin{cases} \left(y_n - f(x_n)\right)^2, & |y_n - f(x_n)| \le \delta \\ |y_n - f(x_n)|, & |y_n - f(x_n)| > \delta \end{cases}$



$$\min_f \frac{1}{N} \sum_{n=1}^{N} \textcolor{red}{regression\_loss}(f(x_n), y_n)$$

Generalization

- Generalization error is a measure of how accurately a model is able to predict outcome values for previously unseen data ※



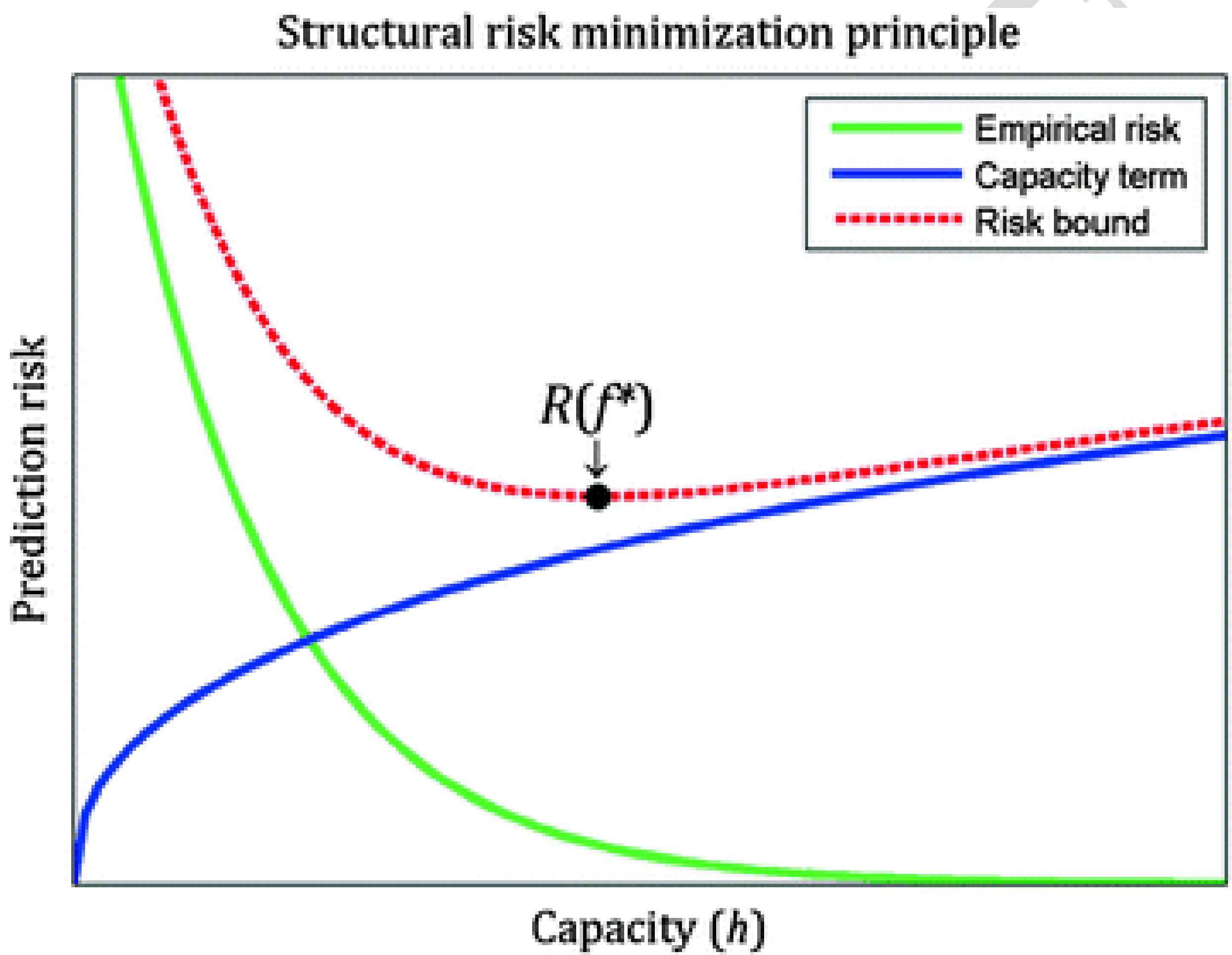
Regularization

- In ML, regularization is a process of introducing additional information in order to prevent overfitting
- Regularized Empirical Risk Minimization:

$$\min_f \frac{1}{N}\sum_{n=1}^{N} loss(f(x_n), y_n) + \lambda R(f)$$

- Some explanations for regularization
  - Solve ill-posed (underdetermined) problem
  - Impose Occam's razor on the solution
  - Impose certain prior distributions on model parameters

Structural Risk Minimization



Objective Function

- General form of objective function for supervised learning

$$\min_{f} \frac{1}{N}\sum_{n=1}^{N} loss(f(x_n), y_n) + \lambda R(f)$$

- Different combinations of $f(\cdot)$, $loss(\cdot)$, and $R(\cdot)$ will result different machine learning models

- Ordinary Linear Model: Ridge Regression
  - Linear model: $f(x_n) = w^\top x_n$
  - Quadratic loss: $loss(f(x_n), y_n) = (y_n - w^\top x_n)^2$
  - $L2$-norm regularization: $R(f) = ||w||^2$

$$\min_{w} \frac{1}{N}\sum_{n=1}^{N} (y_n - w^\top x_n)^2 + \lambda||w||^2$$

Ridge Regression

- Ridge Regression
  - Linear model: $f(x_n) = w^\top x_n$
  - Quadratic loss: $loss(f(x_n), y_n) = (y_n - w^\top x_n)^2$
  - $L2$-norm regularization: $R(f) = ||w||^2$

$$\min_w \frac{1}{N}\sum_{n=1}^{N}(y_n - w^\top x_n)^2 + \lambda||w||^2$$

← Convex Optimization (quadratic programming)

$$\Rightarrow \min_w (Y - Xw)^\top (Y - Xw) + \lambda w^\top w$$

  - Let $J(w) = (Y - Xw)^\top (Y - Xw) + \lambda w^\top w$ ※

$$\frac{\partial J(w)}{\partial w} = -2X^\top (Y - Xw) + 2\lambda w = 0$$

$$w = (X^\top X + I\lambda)^{-1} X^\top Y$$

Logistic Regression

- Logistic Regression
  - Linear model: $f(x_n) = w^\top x_n$
  - Logistic loss: $loss(f(x_n), y_n) = -y_n \ln(\sigma_n) - (1 - y_n) \ln(1 - \sigma_n)$, where $\sigma_n = \frac{1}{1+\exp(-w^\top x_n)}$ and $y_n \in \{0,1\}$
  - $L2$-norm regularization: $R(f) = ||w||^2$

$$\min_{w} \frac{1}{N} \sum_{n=1}^{N} -y_n \ln(\sigma_n) - (1 - y_n) \ln(1 - \sigma_n) + \lambda ||w||^2$$

  - Let $J(w) = -\frac{1}{N} \sum_{n=1}^{N} (y_n \ln(\sigma_n) - (1 - y_n) \ln(1 - \sigma_n)) + \lambda ||w||^2$

$$\frac{\partial J(w)}{\partial w} = \frac{1}{N} \sum_{n=1}^{N} -y_n (1 - \sigma_n) x_n + (1 - y_n) \sigma_n x_n + 2\lambda w$$

$$= \frac{1}{N} \sum_{n=1}^{N} x_n (\sigma_n - y_n) + 2\lambda w$$

Maximum Likelihood Estimation

- In statistics, Maximum Likelihood Estimation (MLE) is a method of estimating the parameters of a statistical model, given observations $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$
- MLE attempts to find the parameter values that maximize the likelihood function ※

$$\max_{\theta\in\Theta} p(D|\theta) \Rightarrow \max_{\theta\in\Theta} \prod_{n=1}^{N} p(x_n, y_n|\theta) \Rightarrow \min_{\theta\in\Theta} \frac{1}{N} \sum_{n=1}^{N} -\log p(x_n, y_n|\theta)$$

  - $p(x_n, y_n|\theta)$ is Gaussian distribution ➔ Quadratic loss
  - $p(x_n, y_n|\theta)$ is Laplacian distribution ➔ Absolute loss
  - $p(x_n, y_n|\theta)$ is Logistic function ➔ Logistic loss

Maximum *a posteriori*

- In Bayesian statistics, Maximum *a posteriori* (MAP) is an estimate of an unknown quantity, that equals the mode of the posterior distribution.
- MAP estimation can be seen as a regularization of MLE.

$$\arg\max_{\theta\in\Theta} p(\theta|D) = \arg\max_{\theta\in\Theta} \frac{p(D|\theta)p(\theta)}{\int_{\theta'} p(D|\theta')p(\theta')d\theta'} = \arg\max_{\theta\in\Theta} p(D|\theta)p(\theta)$$

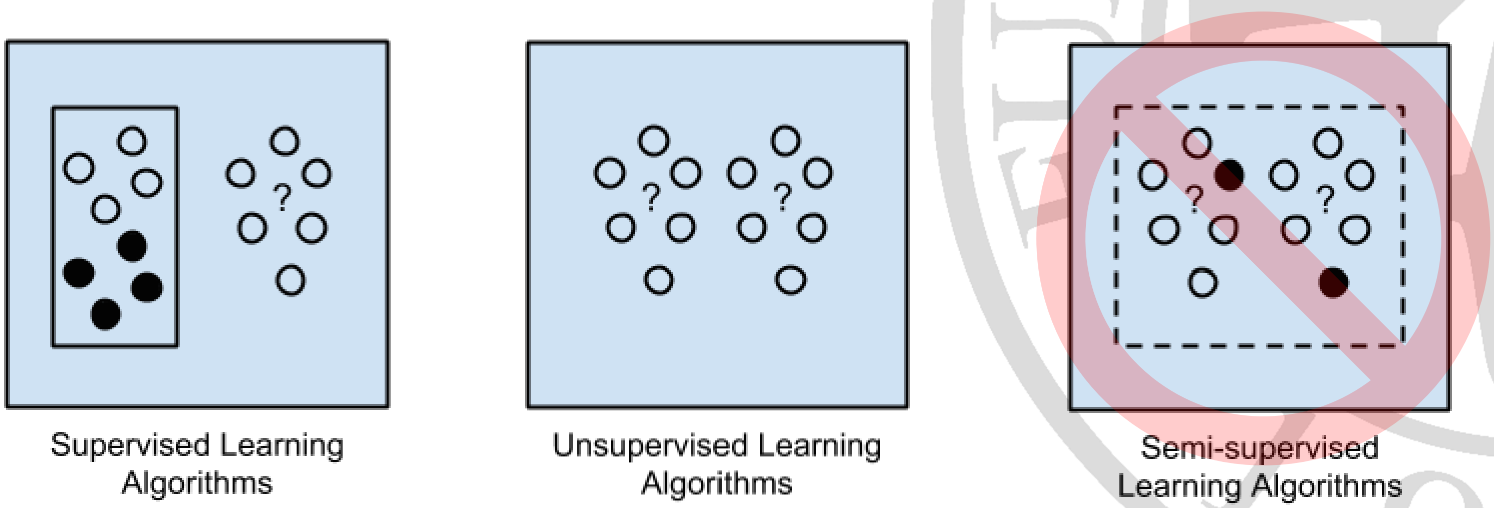
$$\max_{\theta\in\Theta} p(D|\theta)p(\theta) \Rightarrow \max_{\theta\in\Theta} \prod_{n=1}^{N} p(x_n, y_n|\theta)p(\theta)$$

$$\Rightarrow \min_{\theta\in\Theta} \frac{1}{N}\sum_{n=1}^{N} -\log p(x_n, y_n|\theta) - \log p(\theta)$$

- Compare to $\min_{f} \frac{1}{N}\sum_{n=1}^{N} loss(f(x_n), y_n) + \lambda R(f)$ ※

ML Problems in BDA

- In BDA, there are two commonly seen ML problem settings
  - Supervised learning: Classification, Regression
  - Unsupervised learning: Clustering, Dimensionality Reduction



[1] https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Clustering Problem

- Instances: $x_1, x_2, \ldots \in X$, where X is feature space
- Objective: Input instances $x_1, x_2, \ldots \in X$ and output corresponding cluster indicators $z_1, z_2, \ldots \in \{0,1\}^K$ for each instance, satisfying certain optimization criteria

$$\min_{\{z\},\{\theta\}} \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \, clustering\_loss(x_n, \theta_k)$$

  - θ_k denotes the parameter set of the k-th cluster
  - $z_{n,k} \in \{0,1\}$ denotes whether the n-th instance belongs to the k-th cluster
  - Goal: Find values of θ_k and $z_{n,k}$ to minimize the objective

K-Means Clustering

- Minimize $J = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{n,k}\, ||x_n - \mu_k||^2$ in terms of the mean of the cluster μ_k and the cluster indicator $z_{n,k}$
- Given $\{\theta_1, \ldots, \theta_K\}$ fixed, since J is a linear function of $z_{n,k}$, this optimization can be easily obtained by

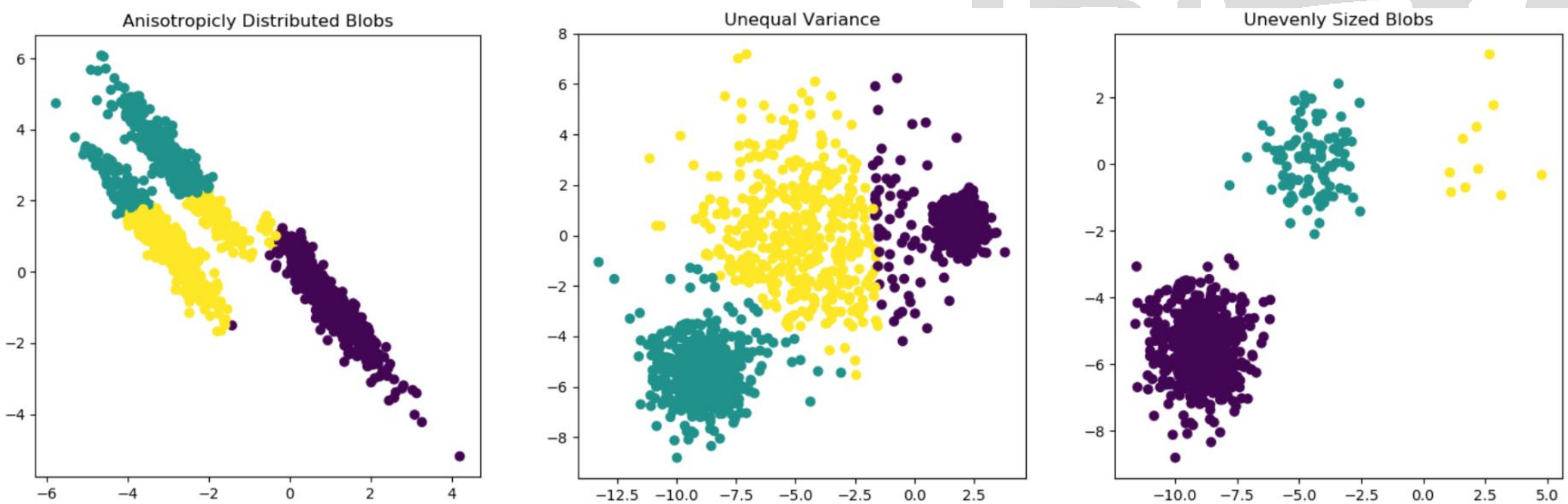
$$z_{n,k} = \begin{cases} 1, & \text{if } k = \arg\min_j ||x_n - \mu_j||^2 \\ 0, & \text{otherwise} \end{cases}$$

- Given $\{z_{1,1}, \ldots, z_{N,K}\}$ fixed, J is a quadratic function of μ_k, it can be minimized by setting its derivative w.r.t. μ_k to zero

$$2\sum_{n=1}^{N} z_{n,k}\,(x_n - \mu_k) = 0 \Rightarrow \mu_k = \frac{\sum_{n=1}^{N} z_{n,k}\, x_n}{\sum_{n=1}^{N} z_{n,k}}$$

Assumptions of K-Means

- Assumptions (sometimes limitations) in K-Means
  - Isotropically distributed
  - Distributed with equal variances
  - Clusters are evenly sized



Gaussian Mixture Model (GMM)

- K-Means is hard-membership clustering technique
  - $z_{n,k} \in \{0,1\}$: Each instance can or cannot belong to a cluster
  - $\sum_{k=1}^{K} z_{n,k} = 1$: Each instance can belong to only one cluster
- Is there a soft-membership clustering technique?
  - $z_{n,k} \in \{0,1\} \Rightarrow \gamma(z_{n,k}) \in [0,1]$
  - $\sum_{k=1}^{K} \gamma(z_{n,k}) = 1$ still holds
- Gaussian Mixture Model

$$p(x_n|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$$

  - Anisotropically distributed
  - Distributed with different variances
  - Clusters are variously sized

Maximum Likelihood of GMM

- Given $\{x_1, \ldots, x_N\}$ and we wish to model these instances using a GMM. The log likelihood function is

$$\ln p(X|\{\pi_k, \mu_k, \Sigma_k\}_k) = \sum_{n=1}^{N} \ln\left\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)\right\}$$

- Set the derivatives of $\ln p(X|\{\pi_k, \mu_k, \Sigma_k\}_k)$ w.r.t. the means μ_k of the Gaussian components to zero

$$-\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(x_n|\mu_{k'}, \Sigma_{k'})} \Sigma_k (x_n - \mu_k) = 0$$

$$\gamma(z_{n,k}) = p\left(z_{n,k} = 1|x_n\right) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(x_n|\mu_{k'}, \Sigma_{k'})}$$

$$-\sum_{n=1}^{N} \gamma\left(z_{n,k}\right) \Sigma_k (x_n - \mu_k) = 0 \Rightarrow \mu_k = \frac{\sum_{n=1}^{N} \gamma(z_{n,k})\, x_n}{\sum_{n=1}^{N} \gamma(z_{n,k})}$$

Maximum Likelihood of GMM

- Set the derivatives of $\ln p(X|\pi,\mu,\Sigma)$ w.r.t. the covariance matrix Σ_k of the Gaussian components to zero

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma(z_{n,k})(x_n - \mu_k)(x_n - \mu_k)^\top}{\sum_{n=1}^{N} \gamma(z_{n,k})}$$

- Set the derivatives of $\ln p(X|\pi,\mu,\Sigma)$ w.r.t. the mixing coefficients π_k subject to $\sum_{k=1}^{K} \pi_k = 1$

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{n,k})}{N}$$

- In the MLE of (π_k, μ_k, Σ_k) we assume $\gamma(z_{n,k})$ is given, which however is also conditioned on (π_k, μ_k, Σ_k)

Expectation-Maximization (EM) Algorithm for GMM

- Initialize (π_k, μ_k, Σ_k) and evaluate $\ln p(X|\pi, \mu, \Sigma)$
- E-Step: Evaluate $\gamma(z_{n,k})$ using the current (π_k, μ_k, Σ_k)

$$\gamma(z_{n,k}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(x_n|\mu_{k'}, \Sigma_{k'})}$$

- M-Step. Estimate (π_k, μ_k, Σ_k) using the current $\gamma(z_{n,k})$

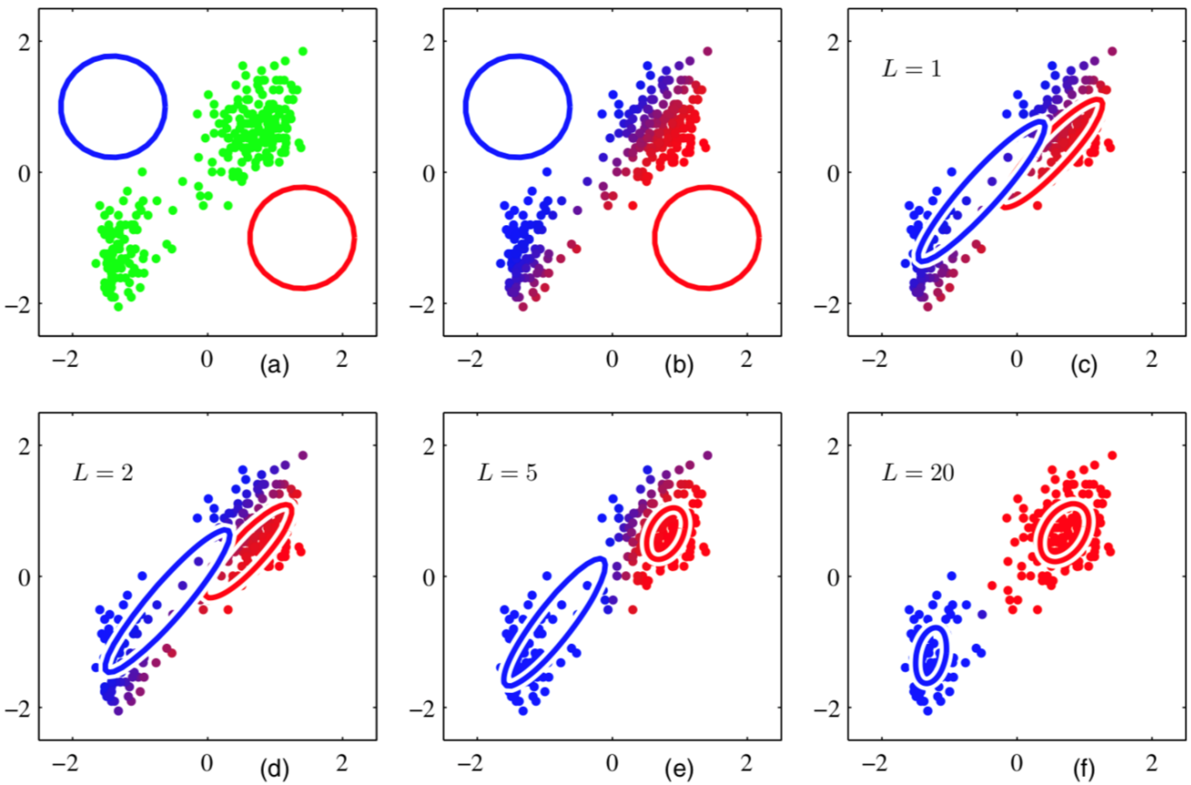
$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{n,k})}{N} \qquad \mu_k = \frac{\sum_{n=1}^{N} \gamma(z_{n,k})\, x_n}{\sum_{n=1}^{N} \gamma(z_{n,k})}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma(z_{n,k})(x_n - \mu_k)(x_n - \mu_k)^\top}{\sum_{n=1}^{N} \gamma(z_{n,k})}$$

- Evaluate $\ln p(X|\pi, \mu, \Sigma)$ and check the convergence

EM Algorithm for GMM

- Illustration of EM iterations for fitting a GMM



[1] Bishop, Pattern Recognition and Machine Learning, page 437.

EM Algorithm for Latent Variable Models

- Given a joint distribution $p(X, Z|\Theta)$ over observed variables X and latent variables Z, governed by parameters Θ, the goal is to maximize the likelihood function $p(X|\Theta)$ w.r.t. Θ.
- The general EM algorithm:
  - Initialize the parameters Θ^{old};
  - **E-Step**: Evaluate $p\left(Z|X, \Theta^{\text{old}}\right)$;
  - **M-Step**: Evaluate Θ^{new} given by

$$\Theta^{\text{new}} = \underset{\Theta}{\operatorname{argmax}} \sum_{Z} p\left(Z|X, \Theta^{\text{old}}\right) p(X, Z|\Theta)$$

  - Check the convergence of the parameter values; if convergence condition is not satisfied, set $\Theta^{\text{old}} = \Theta^{\text{new}}$ and go to E-step.

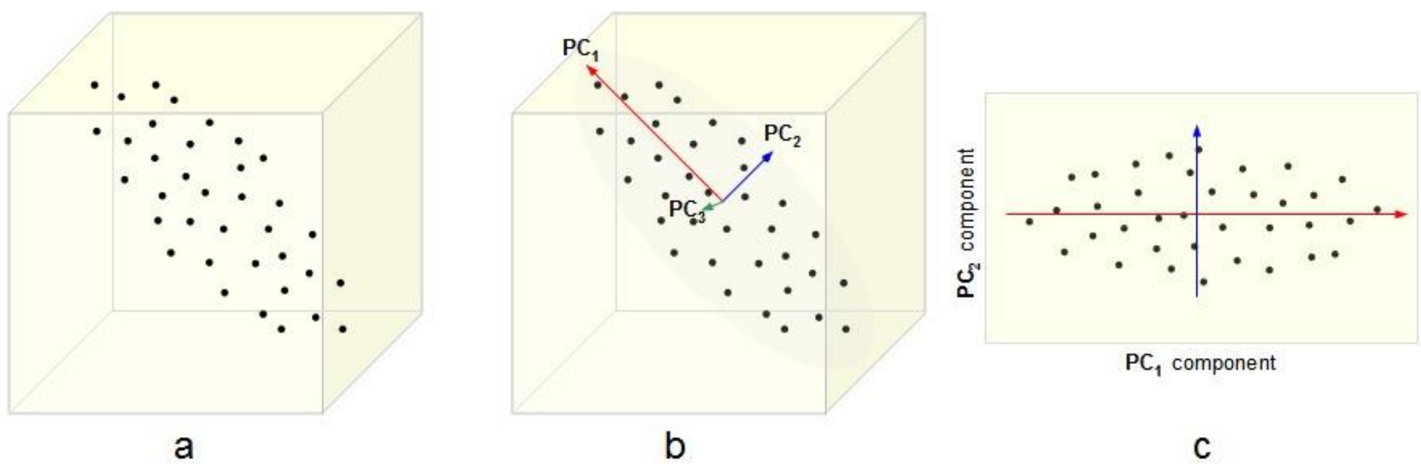
Latent Variable Models

- Latent Class Analysis (discrete latent variable model) is usually based on cluster assumption
  - Mixture Model
  - Topic Model
  - Relational Model
  - Latent Feature Model
  - etc.

- Latent Factor Analysis (continuous latent variable model) is usually based on subspace assumption
  - Principal Component Analysis
  - Matrix Factorization
  - etc.

Principal Component Analysis

- Given $\{x_1, \ldots, x_N\} \in R^D$, PCA is to project the data on to a space that maximizes the variance of the projected data
  - Noise filtering
  - Dimensionality reduction
  - Data visualization
  - Data compression
  - etc.



Principal Component Analysis

- Suppose the first principal component is u_1, then each data point is projected onto a one-dimensional space $u_1^\top x_n$
- The variance of the projected data is given by

$$\frac{1}{N}\sum_{n=1}^{N}\left(u_1^\top x_n - u_1^\top \bar{x}\right)^2 = u_1^\top\left(\frac{1}{N}\sum_{n=1}^{N}(x_n-\bar{x})(x_n-\bar{x})^\top\right)u_1 = u_1^\top \Sigma u_1$$

- Maximize $u_1^\top \Sigma u_1$ w.r.t. u_1 subject to $u_1^\top u_1 = 1$

$$\max_{u_1} u_1^\top \Sigma u_1 + \lambda_1(1 - u_1^\top u_1) \Rightarrow u_1^\top \Sigma u_1 = \lambda_1$$

- $\Sigma u_1 = \lambda_1 u_1$ indicates u_1 is an eigenvector of Σ and this can be solved using eigendecomposition.

Dimensionality Reduction

- In statistics and machine learning, dimensionality reduction is to reduce the number of random variables by obtaining a set of principal variables.
  - Principal Component Analysis (PCA)
  - Canonical Correlation Analysis (CCA)
  - Nonnegative Matrix Factorization (NMF)
  - Autoencoder
  - Learning to Hash
  - Random Projection
  - Locality-Sensitive Hashing (LSH)
  - etc.

Model Selection

- A common problem in machine learning is to select a hyper-parameter which usually determines the structure (or complexity) of the model
  - Number of components in a GMM
  - Number of latent dimensions in matrix factorization
  - Hyper-parameters in neural networks
  - Hyper-parameters in kernel methods
  - Hyper-parameters in Bayesian methods
  - etc.
- Criteria for model selection
  - Cross-validation – most frequently used
  - Information theory based criteria (AIC, BIC, MDL, etc.)
  - Bayesian nonparametric methods

Bayesian Methods

- Recall that Maximum *a posteriori* (MAP) is an estimate of an unknown quantity, that equals the mode of the posterior distribution – point estimation.

$$\arg\max_{\theta\in\Theta} p(\theta|D) = \arg\max_{\theta\in\Theta} \frac{p(D|\theta)p(\theta)}{\int_{\theta'} p(D|\theta')p(\theta')d\theta'} = \arg\max_{\theta\in\Theta} p(D|\theta)p(\theta)$$

- Bayesian methods treat parameters as random variables and infer the posterior distribution

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int_{\theta'} p(D|\theta')p(\theta')d\theta'}$$

- Bayesian methods use predictive distribution for prediction

$$p(x|D) = \int_{\theta} p(x|\theta)p(\theta|D)d\theta$$

Point Estimation vs Bayesian Inference

- Point estimation (MLE & MAP) is simpler and more efficient to solve
  - Build MLE/MAP (objective function) conditioned on Θ
  - Find optimal Θ^* using some optimization techniques
  - Substitute Θ^* into the model for prediction

- Bayesian inference: There are integrals in both inference stage and prediction stage
- Advantages of Bayesian inference
  - Avoid local optima
  - Predict with confidence
  - Incorporate rich prior knowledge

Thanks

Email: libin@fudan.edu.cn

MAP & Generalized Linear Model

- MAP: $\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} -\log p(x_n, y_n | \theta) - \log p(\theta)$
- In statistics, the generalized linear model (GLM) is a flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution.
- GLM generalizes linear regression by allowing the linear model to be related to the target via a link function.
  - $p(x_n, y_n | \theta)$ is Gaussian distribution ➔ Link function: $w^\top x_n = \mu$ (Identity) ➔ Mean function: $\mu = w^\top x_n$
  - $p(x_n, y_n | \theta)$ is Logistic function ➔ Link function: $w^\top x_n = \ln \frac{\mu}{1-\mu}$ (Logit) ➔ Mean function: $\mu = \frac{1}{1+\exp(-w^\top x_n)}$