CHAPTER

Sample-Split Models

Chang, Chih-Hao

Department of Statistics, National Chengchi University, Taipei, Taiwan

Summary

This course provides a structured introduction to statistical modeling techniques for analyzing heterogeneous data structures. We begin with unsupervised learning approaches, focusing on clustering methods such as Gaussian Mixture Models (GMM) and K-means, which are applicable when class labels are not observed. We then move to supervised learning frameworks, introducing the logistic regression model, the stochastic gradient descent (SGD) algorithm for large-scale optimization, and the support vector machine (SVM) as a margin-based classifier. Finally, we focus on threshold regression models designed to capture structural changes within data. We present both Threshold Boundary Regression (TBR) and Threshold Boundary Logistic Regression (TBLR) models, highlighting their flexibility in modeling data-driven partitions. Numerical studies are provided to demonstrate the practical advantages and limitations of the proposed methods.

KEYWORDS:
Unsupervised Learning, Clustering, Gaussian Mixture Model, K-means, Logistic Regression, Stochastic Gradient Descent, Support Vector Machine, Threshold Regression, Structural Change, Data-driven Partitioning

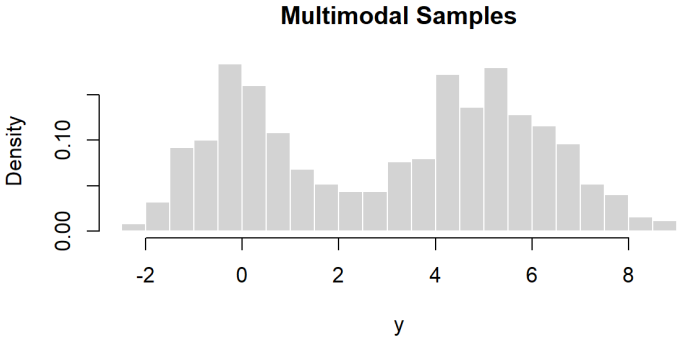
1 | INTRODUCTION

1. In this course, we consider a dataset $\mathcal{D} = \{(y_n, \boldsymbol{x}_n)\}_{n=1}^N$, where y_n denotes the response variable, which may be either continuous or discrete, and $\boldsymbol{x}_n = (1, x_{n,1}, \ldots, x_{n,d})^\top \in \mathbb{R}^{d+1}$ denotes the $(d+1)$-dimensional covariate vector.
2. The appropriate modeling approach depends on the available information and the structural characteristics of the data.
3. When only a univariate response variable y is observed and its empirical distribution displays multimodal behavior, we introduce the *Gaussian Mixture Model* (GMM) to uncover latent subpopulations that may explain the observed heterogeneity.
4. In contrast, when only the explanatory variables $\{\boldsymbol{x}_n\}$ are observed—without any associated responses—the task becomes one of *unsupervised clustering*, for which the *K-means algorithm* provides a foundational approach.
5. Finally, when both covariates and responses are observed and we suspect the presence of heterogeneous subgroups governed by distinct generating mechanisms, we employ *threshold regression models*.
6. In the context of threshold regression models, we further incorporate statistical and machine learning techniques for *binary classification*, which serve as building blocks for constructing flexible, data-driven, nonlinear decision rules that enable principled sample splitting.

2 | UNSUPERVISED LEARNING

1. In introductory statistics, the normal distribution is often presented as a prototypical unimodal distribution, where estimating the mean and variance provides insights into the central tendency and spread of the data.

2. However, in practice, data may exhibit multimodal patterns, reflecting underlying heterogeneity not captured by a single normal model.
3. **A fundamental question**: how can we decompose or model such data structures to reveal their latent components?
4. Addressing this leads us to a more advanced topic—mixture modeling.



5. Consider the Gaussian mixture model (GMM) for the dataset with no covariates: $\mathcal{D}_y = \{y_n\}_{n=1,\ldots,N}$:

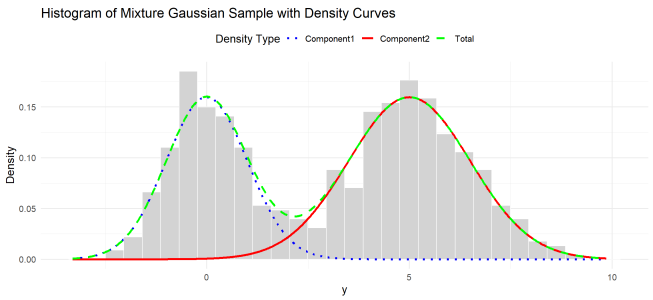
$$f(y_n) = \sum_{k=1}^{K} \pi_k \cdot \phi(y_n|\mu_k, \sigma_k^2), \tag{1}$$

where

   (a) y_n: the responses;
   (b) K: number of components;
   (c) $\phi(\mu_k, \sigma_k^2)$: the normal densities with mean μ_k and variance σ_k^2;
   (d) π_k: the mixing proportion parameters satisfying $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$.

6. A toy example with two components:

   (a) $K = 2$
   (b) $N = 500$
   (c) $\pi_1 = 0.4$ and $(\mu_1, \sigma_1^2) = (0, 1^2)$
   (d) $\pi_2 = 0.6$ and $(\mu_2, \sigma_2^2) = (5, 1.5^2)$



7. The generative formulation of GMMs:

$$Y_n|Z_n = k \sim N(\mu_k, \sigma_k^2), \quad \text{with } Z_n \sim \text{Multinomial}(1; \pi_1, \ldots, \pi_K), \tag{2}$$

where Z_n denotes a categorical latent variable.

8. How to estimate the GMM for the dataset $\mathcal{D}_y$?

   (a) Maximum likelihood estimator (MLE) of $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_K, \mu_1, \ldots, \mu_K, \sigma_1^2, \ldots, \sigma_K^2)$: to maximize the log-likelihood function:

$$\ell(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \cdot \phi(y_n|\mu_k, \sigma_k^2)\right),$$

   which is non-convex and involves a log-sum term, making direct optimization challenging.
   (b) The model includes latent (unobserved) variables: the component label z_n for each observation y_n.
   (c) **Expectation-Maximization (EM) Algorithm**: given initial values $\pi_k^{(0)}$, $\mu_k^{(0)}$ and $\sigma_k^{(0)}$, for $k = 1, \ldots, K$

- **E-step:** Compute the posterior probability that observation y_i belongs to component k:

$$\gamma_{n,k}^{(t)} = \frac{\pi_k^{(t)} \cdot \phi(y_n | \mu_k^{(t)}, \sigma_k^{2(t)})}{\sum_{\ell=1}^{K} \pi_\ell^{(t)} \cdot \phi(y_n | \mu_\ell^{(t)}, \sigma_\ell^{2(t)})}$$

- **M-step:** Update parameters to maximize the expected complete-data log-likelihood:

$$\pi_k^{(t+1)} = \frac{1}{N} \sum_{n=1}^{N} \gamma_{n,k}^{(t)}, \quad \mu_k^{(t+1)} = \frac{\sum_{n=1}^{N} \gamma_{n,k}^{(t)} y_n}{\sum_{n=1}^{N} \gamma_{n,k}^{(t)}}, \quad \sigma_k^{2(t+1)} = \frac{\sum_{n=1}^{N} \gamma_{n,k}^{(t)} (y_n - \mu_k^{(t+1)})^2}{\sum_{n=1}^{N} \gamma_{n,k}^{(t)}}$$

(d) **Convergence:**

i. The EM algorithm monotonically increases the observed-data log-likelihood at each iteration.

ii. It converges to a local maximum, so the initialization is critical.

iii. How to set the initial values $\pi_k^{(0)}$, $\mu_k^{(0)}$ and $\sigma_k^{(0)}$, for $k = 1, \ldots, K$ (suppose K is known)?

iv. The K-means Algorithm!

A. Run K-means clustering on $y_1, \ldots, y_N$ to obtain temporary labels z_n, $n = 1, \ldots, N$

B. Set $\mu_k^{(0)}$ to the sample means of the K clusters

C. Set $\sigma_k^{2(0)}$ to the within-cluster sample variance of the K clusters

D. Set $\pi_k^{(0)} = \sum_{n=1}^{N} I_k(z_n)/N$, with indicator $I_k(z_n) = 1$ if $z_n = k$; and 0, otherwise.

9. **K-means Algorithm**:

(a) **Goal**: Partition N observations $\{y_1, \ldots, y_N\}$ into K clusters (or sets, denoted by $C_1, \ldots, C_k$) that minimize the total within-cluster sum of squares (WCSS):

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \sum_{y_n \in C_k} (y_n - \mu_k)^2, \quad \text{where } \mu_k = \frac{1}{|C_k|} \sum_{y_n \in C_k} y_n.$$

(b) **Algorithm Steps:**

i. **Initialization**: Randomly select K observations as initial cluster centers $\mu_1^{(0)}, \ldots, \mu_K^{(0)}$.

ii. **Assignment Step (E-step)**: Assign each y_n to the nearest cluster center:

$$z_n^{(t)} = \arg \min_{k \in \{1,\ldots,K\}} (y_n - \mu_k^{(t)})^2,$$

in other words, $y_n \in C_k^{(t)}$ if $z_n^{(t)} = k$.

iii. **Update Step (M-step)**: Recalculate the cluster centers using the new assignments:

$$\mu_k^{(t+1)} = \frac{1}{|C_k^{(t)}|} \sum_{y_n \in C_k^{(t)}} y_n.$$

(c) **Convergence Check:** Stop when the assignments no longer change (i.e., $z_n^{(t+1)} = z_n^{(t)}$ for all $n = 1, \ldots, N$) or the change in centroids is below a threshold (i.e., $\sum_{k=1}^{K} |\mu_k^{(t+1)} - \mu_k^{(t)}| < \epsilon$ with e.g., $\epsilon = 10^{-6}$).

(d) **Remarks**:

- K-means is an *unsupervised learning* algorithm;
- It may converge to a *local minimum* depending on initialization;
- Sensitive to outliers and poorly suited for irregular cluster shapes.

10. Generalization of K-means algorithm:

(a) Note that the K-means algorithm can be applied not only to clustering univariate responses but also to datasets consisting solely of covariates, denoted by $\mathcal{D}_x = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$.

(b) In this case, the scalar response y_n is replaced by the covariate vector $\boldsymbol{x}_n$, and the squared distance measure $(y_n - \mu_k^{(t)})^2$ is naturally generalized to the Euclidean norm $\|\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)}\|^2$.

11. What if K, the number of clusters, is unknown and need to be estimated?

(a) One possible way is to select K based on some information criteria such as Akaike's information criterion (AIC) or Bayesian information criterion (BIC).

(b) BIC for a GMM with $\bar{K}$ (a prefixed upper bound of the number of clusters) components is defined as:

$$\text{BIC}(K) = -2 \cdot \log L_K + p_K \cdot \log N, \quad K = 1, \ldots, \bar{K}$$

i. L_K is the maximized likelihood of the model with K components;

ii. p_K is the number of estimated parameters in the model:

$$p_K = K \text{ (means)} + K \text{ (variances)} + (K-1) \text{ (mixing proportions)} = 2K + (K-1) = 3K - 1.$$

iii. N is the total number of observations.

(c) Model selection rule: Choose the value of K that minimizes the BIC.

(d) BIC balances model fit (via the log-likelihood term) with model complexity (via the penalty term $p_K \log N$) to prevent overfitting by discouraging unnecessary components.

12. R Code for the simulation:

```
library(mclust)
set.seed(123)
n = 1000
pi = 0.4
m1 = 0
s1 = 1
m2 = 5
s2 = 1.5
z = rbinom(n, 1, prob = pi)
y = ifelse(z == 1,rnorm(n, mean = m1, sd = s1),rnorm(n, mean = m2, sd = s2))
fit.GMM =  Mclust(y, G = 2, modelNames = "V")
sorted_idx = order(fit.GMM$parameters$mean)
pi_hat  = fit.GMM$parameters$pro[sorted_idx]
mu_hat  = fit.GMM$parameters$mean[sorted_idx]
sd_hat  = sqrt(fit$parameters$variance$sigmasq)[sorted_idx]
```

13. **Checkpoint: GMM or K-means?**

(a) To close our discussion of unsupervised learning in this section, consider the following scenarios:

i. Suppose a dataset is formed by pooling samples from two underlying sources, but the source labels are unobserved.

ii. Which method—GMM or K-means—is more appropriate, and why?

iii. In modeling consumer purchasing behavior where decisions vary probabilistically across time, might a GMM be more suitable than hard clustering?

(b) **Interpretation goal**:

i. Do we need probabilistic cluster membership and uncertainty quantification (favoring GMM)?

ii. Or just a geometric partition of the sample space (favoring K-means)?

3 | SUPERVISED LEARNING

3.1 | Logistic Regression Model

1. Consider the binary dataset $\mathcal{D}_b = \{(z_n, \boldsymbol{x}_n)\}$ with $z_n \in \{+1, -1\}$ and $\boldsymbol{x}_n \in \mathbb{R}^{d+1}$.

2. Consider logistic regression model for soft binary classification:

(a) ideal (noiseless) data:

$$\mathcal{D}_b = \begin{cases} (\boldsymbol{x}_1, z_1 = 0.9 = P(z_n = +1|\boldsymbol{x}_1)) \\ (\boldsymbol{x}_2, z_2 = 0.2 = P(z_n = +1|\boldsymbol{x}_2)) \\ \vdots \\ (\boldsymbol{x}_N, z_N = 0.6 = P(z_n = +1|\boldsymbol{x}_N)) \end{cases}$$

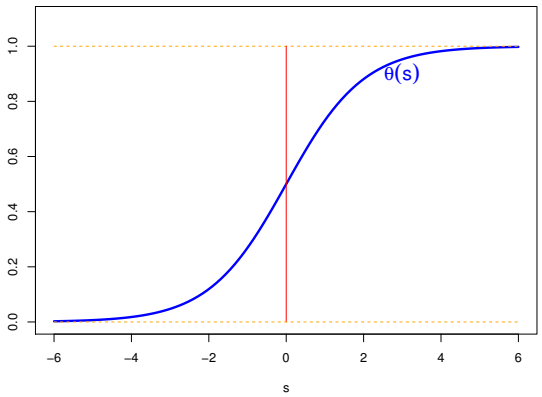
(b) actual (noisy) data:

$$\mathcal{D}_b = \begin{cases} (\boldsymbol{x}_1, y_1 = 1 \sim f(z|\boldsymbol{x}_1)) \\ (\boldsymbol{x}_2, y_2 = -1 \sim f(z|\boldsymbol{x}_2)) \\ \vdots \\ (\boldsymbol{x}_N, y_N = -1 \sim f(z|\boldsymbol{x}_N)) \end{cases}$$

3. In logistic regression, we seek to estimate the target density (mass) function $f(z|\boldsymbol{x})$; however, the concept of model "error" differs fundamentally from that in linear regression.
   (a) For continuous outcomes $y_n \in \mathbb{R}$, prediction accuracy is often evaluated using the squared error loss, defined as $(y_n - \hat{y}_n)^2$, where $\hat{y}_n$ is the predicted value.
   (b) This squared error is intuitive, computationally simple, and aligns with the assumptions of the classical linear regression model.
   (c) In linear regression, $\hat{y}_n$ is modeled as a linear function of the covariates, typically written as $\boldsymbol{\beta}^\top \boldsymbol{x}_n$, where $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$ is estimated via least squares to minimizing $\sum_{n=1}^{N}(y_n - \boldsymbol{\beta}^\top \boldsymbol{x}_n)^2$, the in-sample error, which is denoted by $\mathrm{E}_{\text{in}}(\boldsymbol{\beta})$.
   (d) However, when the response y_n is binary (e.g., $y_n \in \{0, 1\}$), using squared error is no longer appropriate. This raises two fundamental questions:
      i. How can we construct a linear predictor that respects the binary nature of the outcome?
      ii. How should we define and evaluate prediction error in this context?
4. In logistic regression, we deal with the relationship between the categories of observed values and the probabilities predicted by the model.
5. For example, suppose that for some $\boldsymbol{x}$, we know that $P(z = +1|\boldsymbol{x}) = 1$.
   (a) If $z = +1$ is observed, we expect no errors in this case.
   (b) But if $z = -1$ is observed, how large the error it would be?
6. Logistic hypothesis:
   (a) features of patient: $\boldsymbol{x} = (x_0, x_1, \ldots, x_d)'$;
   (b) weighted risk score:

$$s = \sum_{i=0}^{d} w_i x_i;$$

   (c) convert the score to estimated probability by logistic function: $\theta(s) \in [0, 1]$;



   (d) logistic model: $h(\boldsymbol{x}) = \theta(\boldsymbol{w}'\boldsymbol{x})$.
7. Logistic function:

$$\theta(s) = \frac{\exp(s)}{1 + \exp(s)} = \frac{1}{1 + \exp(-s)}.$$

   (a) $\theta(-\infty) = 0$;
   (b) $\theta(0) = 0.5$;

(c) $\theta(\infty) = 1$;

(d) smooth, monotonic, sigmoid function of s;

8. Logistic regression: use

$$h(\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x})},$$

to approximate target function $P(z = +1|\boldsymbol{x})$.

9. Likelihood:

$$p(\boldsymbol{x}) = P(z = +1|\boldsymbol{x}) \Leftrightarrow f(z|\boldsymbol{x}) = \begin{cases} p(\boldsymbol{x}), & z = +1; \\ 1 - p(\boldsymbol{x}), & z = -1. \end{cases}$$

(a) $\mathcal{D}_b = \{(\boldsymbol{x}_1, z_1), \ldots, (\boldsymbol{x}_N, z_N)\}$;

(b) probability that f generates $\mathcal{D}_b$:

$$\prod_{n=1}^{N} f(\boldsymbol{x}_n, z_n) = \prod_{n=1}^{N} f(\boldsymbol{x}_n) f(z_n|\boldsymbol{x}_n);$$

(c) likelihood that h generates $\mathcal{D}_b$:

$$\prod_{n=1}^{N} f_h(\boldsymbol{x}_n, z_n) = \prod_{n=1}^{N} f(\boldsymbol{x}_n) f_h(z_n|\boldsymbol{x}_n),$$

where

$$f_h(z|\boldsymbol{x}) = \begin{cases} h(\boldsymbol{x}), & z = +1; \\ 1 - h(\boldsymbol{x}), & z = -1. \end{cases}$$

(d) if $h \approx p$, then likelihood(h) $\approx$ probability using p, where we generally assume that probability using p large!

10. Likelihood of logistic hypothesis: likelihood(h) $\approx$ probability using $p \approx$ large,

$$\text{aim: } g = \underset{h}{\text{argmax}} \ \text{likelihood}(h).$$

(a) for logistic: $h(\boldsymbol{x}) = \theta(\boldsymbol{w}'\boldsymbol{x})$:

$$1 - h(\boldsymbol{x}) = h(-\boldsymbol{x}),$$

which gives that

$$f_h(z|\boldsymbol{x}) = h(z\boldsymbol{x}),$$

(b) the resulting likelihood of $\mathcal{D}_b$:

$$\text{likelihood}(h) = \prod_{n=1}^{N} f(\boldsymbol{x}_n) h(z_n \boldsymbol{x}_n).$$

11. In optimization problems, it is common to seek the minimum of a cost or error function.

(a) We minimize error measures, or loss functions, because every model inherently possesses imperfections.

(b) Our objective is to find a model that incurs minimal loss when fitted to the dataset.

(c) By taking the negative log of the likelihood function, we effectively turn the maximization of likelihood into a minimization problem of an error measure.

(d) The error measure is then called the **cross-entropy error**:

$$\max_{h} \quad \text{likelihood}(h) \propto \prod_{n=1}^{N} h(z_n \boldsymbol{x}_n),$$

which is equivalent to

$$\max_{\boldsymbol{w}} \quad \text{likelihood}(\boldsymbol{w}) \propto \prod_{n=1}^{N} \theta(z_n \boldsymbol{w}^\top \boldsymbol{x}_n),$$

and recall $\theta(s) = (1 + \exp(-s))^{-1}$:

$$\begin{aligned} & \min_{\boldsymbol{w}} \quad \frac{1}{N} \sum_{n=1}^{N} -\ln \theta(z_n \boldsymbol{w}^\top \boldsymbol{x}_n) \\ \Rightarrow & \min_{\boldsymbol{w}} \quad \frac{1}{N} \sum_{n=1}^{N} \underbrace{\ln(1 + \exp(-z_n \boldsymbol{w}^\top \boldsymbol{x}_n))}_{\text{err}(\boldsymbol{w}^\top \boldsymbol{x}_n, z_n)} \\ \Rightarrow & \min_{\boldsymbol{w}} \quad \underbrace{\frac{1}{N} \sum_{n=1}^{N} \text{err}(\boldsymbol{w}^\top \boldsymbol{x}_n, z_n)}_{E_{\text{in}}(\boldsymbol{w})} \end{aligned}$$

12. Recall the question that for two cases $P(z = +1|\boldsymbol{x}) = 1$ and $z = 1$ is observed, and $P(z = +1|\boldsymbol{x}) = 1$ but $z = -1$ is observed, how the cross-entropy error reasonably performs in these two cases?

13. For cross-entropy error, $\text{err}(\boldsymbol{w}^\top \boldsymbol{x}_n, z_n) = \ln(1 + \exp(-z_n \boldsymbol{w}^\top \boldsymbol{x}_n))$ define $E_{\text{in}}(\boldsymbol{w})$ of logistic regression:

    (a) for any $\boldsymbol{w}^\top \boldsymbol{x}$ and y, $\text{err}(\boldsymbol{w}^\top \boldsymbol{x}_n, z_n) > 0$;

    (b) for $y = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x})$, $\text{err}(\boldsymbol{w}^\top \boldsymbol{x}_n, z_n) < \ln 2$;

    (c) for $y \neq \text{sign}(\boldsymbol{w}^\top \boldsymbol{x})$, $\text{err}(\boldsymbol{w}^\top \boldsymbol{x}_n, z_n) \geq \ln 2$;

14. In summary, the term cross-entropy in the context of binary classification refers to the measure of discrepancy between predicted probabilities and true class labels, and it is named as such due to its connection to information theory concepts.

15. The negative log-likelihood function is equivalent to the cross-entropy error, providing a convenient and interpretable measure of model performance in classification tasks.

3.2 | Stochastic Gradient Descent Algorithm

1. Minimizing $E_{\text{in}}(\boldsymbol{w})$: continuous, differentiable, twice differentiable, convex,

$$\min_{\boldsymbol{w}} E_{\text{in}}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln \left(\underbrace{1 + \exp(\overbrace{-z_n \boldsymbol{w}^\top \boldsymbol{x}_n}^{\bigcirc})}_{\square} \right).$$

2. Gradient $\nabla E_{\text{in}}(\boldsymbol{w})$:

$$\begin{aligned} \frac{\partial}{\partial w_i} E_{\text{in}}(\boldsymbol{w}) &= \frac{1}{N} \sum_{n=1}^{N} \left(\frac{\partial \ln(\square)}{\partial \square} \right) \left(\frac{\partial(1 + \exp(\bigcirc))}{\partial \bigcirc} \right) \left(\frac{\partial -z_n \boldsymbol{w}^\top \boldsymbol{x}_n}{\partial w_i} \right) \\ &= \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{\square} \right) (\exp(\bigcirc))(-z_n x_{n,i}) \\ &= \frac{1}{N} \sum_{n=1}^{N} \left(\frac{\exp(\bigcirc)}{1 + \exp(\bigcirc)} \right) (-z_n x_{n,i}) \\ &= \frac{1}{N} \sum_{n=1}^{N} \theta(\bigcirc)(-z_n x_{n,i}), \end{aligned}$$

which gives that

$$\nabla E_{\text{in}}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta(-z_n \boldsymbol{w}^\top \boldsymbol{x}_n)(-z_n \boldsymbol{x}_n).$$

3. want $\nabla E_{\text{in}}(\boldsymbol{w}) \equiv \mathbf{0}$: a scaled θ-weighted sum of $-z_n \boldsymbol{x}_n$;

   (a) $\theta(\cdot) = 0$ for all n: only if $z_n \boldsymbol{w}^\top \boldsymbol{x}_n \gg 0$ (i.e., linear separable $\mathcal{D}$);

   (b) θ-weighted sum $= \mathbf{0}$: nonlinear equation of $\boldsymbol{w}$ (i.e., no closed form of $\hat{\boldsymbol{w}}$).

4. Iterative optimization: choice of $(\eta, \boldsymbol{v})$ and stopping conditions defines iterative optimization approach:

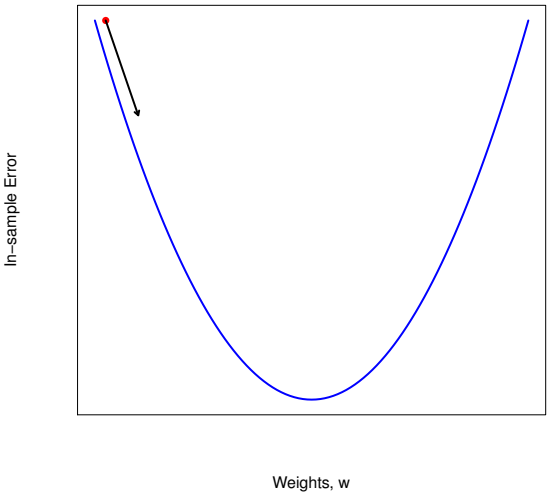
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \boldsymbol{v}; \quad t = 0, 1, \dots,$$

   when stop, return the last $\boldsymbol{w}$ as g.

   (a) smooth $E_{\text{in}}(\boldsymbol{w})$ for logistic regression: choose $\boldsymbol{v}$ to get the ball roll downhill:

   i. direction $\boldsymbol{v}$: (assumed) of unit length;

   ii. step size η: (assumed) positive;



   (b) a greedy approach for some given $\eta > 0$:

$$\min_{\|\boldsymbol{v}\|=1} E_{\text{in}}(\underbrace{\boldsymbol{w}_t + \eta \boldsymbol{v}}_{\boldsymbol{w}_{t+1}}).$$

5. Linear approximation: $\min_{\|\boldsymbol{v}\|=1} E_{\text{in}}(\boldsymbol{w}_t + \eta \boldsymbol{v})$;

   (a) still nonlinear optimization, now with constraints, which is not any easier than $\min_{\boldsymbol{w}} E_{\text{in}}(\boldsymbol{w})$;

   (b) local approximated by linear formula makes problem easier:

$$E_{\text{in}}(\boldsymbol{w}_t + \eta \boldsymbol{v}) \approx E_{\text{in}}(\boldsymbol{w}_t) + \eta \boldsymbol{v}^\top \nabla E_{\text{in}}(\boldsymbol{w}_t),$$

   if η really small (Taylor expansion);

6. an approximate greedy approach for some given small η:

$$\min_{\|\boldsymbol{v}\|=1} \underbrace{E_{\text{in}}(\boldsymbol{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \boldsymbol{v}^\top \underbrace{\nabla E_{\text{in}}(\boldsymbol{w}_t)}_{\text{known}}.$$

   (a) optimal $\boldsymbol{v}$: opposite direction of $\nabla E_{\text{in}}(\boldsymbol{w}_t)$:

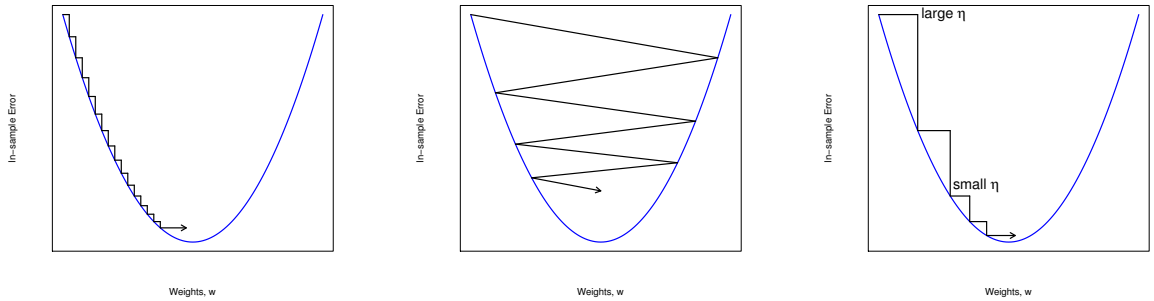
$$\boldsymbol{v} = -\frac{\nabla E_{\text{in}}(\boldsymbol{w}_t)}{\|\nabla E_{\text{in}}(\boldsymbol{w}_t)\|};$$

   (b) gradient descent: for small η,

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \cdot \frac{\nabla E_{\text{in}}(\boldsymbol{w}_t)}{\|\nabla E_{\text{in}}(\boldsymbol{w}_t)\|};$$

7. choice of η: η better be monotonic of $\|\nabla E_{\text{in}}(\boldsymbol{w}_t)\|$.

   (a) too small: (b) too large: (c) just right:



8. Simple heuristic for changing η: if $\eta \propto \|\nabla E_{\text{in}}(\boldsymbol{w}_t)\|$ said $\eta = \eta\|\nabla E_{\text{in}}(\boldsymbol{w}_t)\|$, then

$$\begin{aligned}\boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \eta \frac{\nabla E_{\text{in}}(\boldsymbol{w}_t)}{\|\nabla E_{\text{in}}(\boldsymbol{w}_t)\|} \\ &= \boldsymbol{w}_t - \eta \nabla E_{\text{in}}(\boldsymbol{w}_t),\end{aligned}$$

   where η is called the fixed learning rate (note that here the orange η is different from the red η).

9. **Gradient Descent (GD) Algorithm** for logistic regression:

   (a) initialize $\boldsymbol{w}_0$ and set $t = 0$;

   (b) compute

$$\nabla E_{\text{in}}(\boldsymbol{w}_t) = \frac{1}{N} \sum_{n=1}^{N} \theta(-z_n \boldsymbol{w}_t' \boldsymbol{x}_n)(-z_n \boldsymbol{x}_n);$$

   (c) update by

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla E_{\text{in}}(\boldsymbol{w}_t);$$

   (d) set $t = t + 1$;

   (e) repeat (b)-(d) until $\nabla E_{\text{in}}(\boldsymbol{w}_{t+1}) \approx \boldsymbol{0}$ or enough iterations;

   (f) return the last $\boldsymbol{w}_{t+1}$ as the final model, say g.

10. Efficiency of GD:

   (a) logistic regression: check $\mathcal{D}_b$ and decide $\boldsymbol{w}_{t+1}$ by all samples;
   (i.e., $O(N)$ time per iteration).

   (b) Question: can logistic regression with $O(1)$ time per iteration?

   (c) Logistic regression revisited:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \underbrace{\frac{1}{N} \sum_{n=1}^{N} \theta(-z_n \boldsymbol{w}_t^{\top} \boldsymbol{x}_n)(z_n \boldsymbol{x}_n)}_{-\nabla E_{\text{in}}(\boldsymbol{w}_t)}$$

   (d) want: update direction $\boldsymbol{v} \approx (-1) \cdot \nabla E_{\text{in}}(\boldsymbol{w}_t)$ while computing $\boldsymbol{v}$ by by one single sample $(\boldsymbol{x}_n, z_n)$;

   (e) technique on removing $\frac{1}{N} \sum_{n=1}^{N}$: by viewing as an expectation E over uniform choice of samples;

11. **Stochastic Gradient Descent (SGD) Algorithm** for logistic regression:

$$\text{stochastic gradient} = \text{true gradient} + \text{zero-mean noise directions};$$

(a) stochastic gradient: $\nabla_{\boldsymbol{w}} \text{err}(\boldsymbol{w}^\top \boldsymbol{x}_n, z_n)$ with some random n;

(b) replace the true gradient by the randomly chosen direction: $\nabla_{\boldsymbol{w}} E_{\text{in}}(\boldsymbol{w}) = \text{E}\nabla \text{err}(\boldsymbol{w}^\top \boldsymbol{x}_n, z_n)$;

(c) SGD logistic regression:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \cdot \underbrace{\theta(-z_n \boldsymbol{w}_t^\top \boldsymbol{x}_n)(z_n \boldsymbol{x}_n)}_{-\nabla \text{err}(\boldsymbol{w}_t^\top \boldsymbol{x}_n, z_n)}.$$

(d) After enough steps: average true gradient $\approx$ average stochastic gradient;

(e) Pros: simple and cheaper computation; useful for big data or online learning;

(f) Cons: less stable in nature.

3.3 | Functions of Logistic Regression in R/RStudio

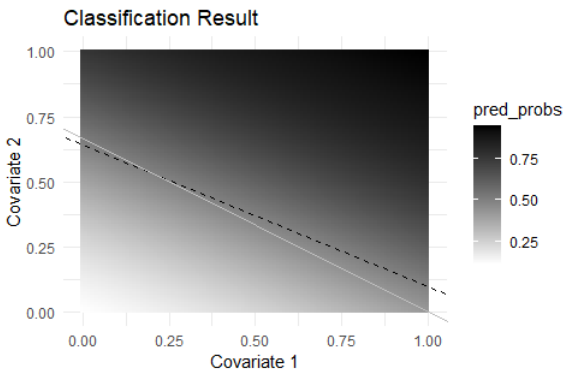
1. Code in R:

```
# Load necessary libraries
library(ggplot2)
# Set seed for reproducibility
set.seed(123)
# Simulate data
n = 1000
covariate1 = runif(n)
covariate2 = runif(n)
intercept = rep(1, n)
# Simulated outcome using logistic regression formula
outcome = rbinom(n, 1, plogis(2 * covariate1 + 3 * covariate2 - 2))
# Create dataframe
data = data.frame(covariate1, covariate2, intercept, outcome)
# Fit logistic regression model
model = glm(outcome ~ covariate1 + covariate2, data = data, family = binomial)
intercept = -coef(model)[1] / coef(model)[3]
slope = -coef(model)[2] / coef(model)[3]
# Generate prediction grid
seq.x = seq(0,1,length=100)
pred_grid = expand.grid(covariate1 = seq.x, covariate2 = seq.x)
# Predict probabilities using the model
pred_probs = predict(model, newdata = pred_grid, type = "response")
# Create dataframe for prediction grid
pred_df = cbind(pred_grid, pred_probs)
# Plot classification result with logistic regression decision boundary
ggplot(data = pred_df, aes(x = covariate1, y = covariate2, fill = pred_probs)) +
geom_tile() +
scale_fill_gradient(low = "white", high = "black") +
# Add logistic regression decision boundary
geom_abline(intercept = intercept, slope = slope, color = "black", linetype = "dashed") +
geom_abline(intercept = 2/3, slope = -2/3, color = "grey", linetype = "solid") +
labs(x = "Covariate 1", y = "Covariate 2", title = "Classification Result") +
theme_minimal()
```

2. **Generate Data**:
   (a) In this step, we generate simulated data for two covariates (covariate1 and covariate2) that are uniformly distributed in the interval $(0, 1)$.
   (b) We also simulate a binary outcome variable (outcome) based on a logistic regression formula (i.e., Binomial distributed with one experiment and the probability evaluated from using function **plogis()**).
3. **Create Dataframe**:
   (a) Here, we create a dataframe data (i.e., function **data.frame()**) to organize the generated covariates and outcome into a single dataset.
   (b) In most regression models in R, the input dataset shall be formed as a dataframe.
4. **Fit Logistic Regression Model**:
   (a) We fit a logistic regression model using the **glm()** function.
   (b) The formula **outcome ~ covariate1 + covariate2** specifies that outcome is the response variable, and covariate1 and covariate2 are the predictor variables.
   (c) We specify **family = binomial** to indicate that we are fitting a logistic regression model.
5. **Generate Prediction Grid**:
   (a) We create a prediction grid **pred_grid** with 100 evenly spaced values for each covariate, ranging from 0 to 1.
6. **Predict Probabilities**:
   (a) We apply the function **predict()** to the fitted logistic regression model for predicting the probabilities of the outcome being 1 for each combination of covariate values in the prediction grid.
   (b) The linear classifier satisfies

$$\hat{\beta}_0 + \hat{\beta}_1 \text{Covariate 1} + \hat{\beta}_2 \text{Covariate 2} = 0$$
$$\Rightarrow \text{Covariate 2} = \left(-\frac{\hat{\beta}_0}{\hat{\beta}_1}\right) + \left(-\frac{\hat{\beta}_2}{\hat{\beta}_0}\right)\text{Covariate 1} \equiv \textbf{intercept} + \textbf{slope} \cdot \text{Covariate 1},$$

   where we use **coef(model)** to obtain the vector $(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2)'$.
7. **Create Dataframe for Prediction Grid**:
   (a) We create a **dataframe pred_df** to store the prediction grid along with the predicted probabilities.
8. **Plot Classification Result**:
   (a) Finally, we use **ggplot2** to create a plot of the classification result.
   (b) We use **geom_tile()** to create a heatmap of predicted probabilities.
   (c) We apply **geom_abline()** to add the logistic regression decision boundary as a dashed black line, and the underlying classifier as a solid grey line.
   (d) Various other functions are also applied to customize the appearance of the plot.



3.4 | Support Vector Machine (SVM)

3.4.1 | Linear SVM

1. SVM is an algorithm for binary classification.
2. SVM focuses on maximizing the margin between classes, making it robust and effective for high-dimensional data and handling outliers.
3. A linear classifier shall be away from the data points as far as possible to avoid the noise disturbance from the data, where the more noise tolerance means the more robust to overfitting.

   (a) $\mathcal{D}_b = \{(z_n, \boldsymbol{x}_n)\}_{n=1}^N$: $z_n \in \{+1(\circ), -1(\triangle)\}$ and $\boldsymbol{x}_n \in \mathbb{R}^d$.

   (b) As shown in the figure, the dataset is linear separable and how many linear classifiers you can draw over the figures?

   (c) Consider a scenario where you are identified with a label of $+1(\bullet)$, and your height and weight represent two covariates.

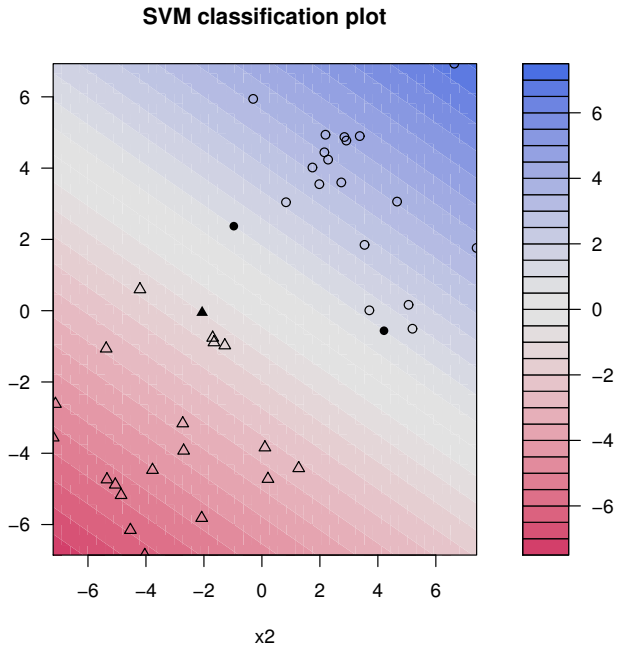
   (d) Suppose there exists a linear classifier positioned in close proximity to you, accurately assigning you to the correct category.

   (e) In this scenario, the classifier's decision may be highly contingent on your weight, rendering it particularly sensitive to fluctuations in this covariate.

   (f) Consequently, even minor changes in your weight could lead to vastly different classification outcomes by this linear classifier.

   (g) To mitigate such sensitivity, it would be prudent to draw a decision boundary that lies significantly distant from the data points represented by the black triangle and circle in the figure.

   (h) This approach helps ensure that classification decisions are less influenced by individual data points and more indicative of the overall trend in the data.



4. Hence, SVM try to fit a linear classifier with **the maximal margin to separate all the data points successfully**.
5. In other words, for a linear classifier, $h(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = 0$:

$$\begin{aligned} \max \quad & \text{margin}(h), \\ \text{subject to} \quad & h \text{ classifies every } (\boldsymbol{x}_n, z_n) \text{ correctly}, \\ & \text{margin}(h) = \min_{n=1,\dots,N} \text{distance}(\boldsymbol{x}_n, h). \end{aligned} \tag{3}$$

6. We say the classifier is correct if $z_n = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_n + b)$; $n = 1, \dots, N$.
7. We can then rewrite (3) into:

$$\begin{aligned} \max \quad & \text{margin}(h) \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) > 0, \\ & \text{margin}(h) = \min_{n=1,\dots,N} \text{distance}(\boldsymbol{x}_n, (b, \boldsymbol{w})). \end{aligned} \tag{4}$$

8. We have the distance from $\boldsymbol{x}^*$ to the hyperplane $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$:

$$\text{distance}(\boldsymbol{x}^*, (b, \boldsymbol{w})) = \frac{1}{\|\boldsymbol{w}\|} |\boldsymbol{w}^\top \boldsymbol{x}^* + b|.$$

9. Note that $z_n \in \{-1, +1\}$, and the hyperplane ask for correctness, $z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) > 0$, we can rewrite the distance to separating hyperplane as:

$$\text{distance}(\boldsymbol{x}_n, (b, \boldsymbol{w})) = \frac{1}{\|\boldsymbol{w}\|} z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b).$$

10. We then parameterize the problem of (4):

$$\begin{aligned} \max \quad & \text{margin}(h), \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) > 0; \quad n = 1, \dots, N, \\ & \text{margin}(h) = \min_{n=1,\dots,N} \frac{1}{\|\boldsymbol{w}\|} z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b). \end{aligned} \tag{5}$$

11. Note that the two linear classifiers: $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ and $3\boldsymbol{w}^\top \boldsymbol{x} + 3b = 0$ are the same, where scaling 3 does not matter.

12. Hence, we can assume that the minimal distance special scaling at

$$\min_{n=1,\dots,N} z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) = 1. \tag{6}$$

13. Hence by (6), we can rewrite the SVM problem of (5) as:

$$\begin{aligned} \max_{b,\boldsymbol{w}} \quad & \frac{1}{\|\boldsymbol{w}\|}, \\ \text{subject to} \quad & \min_{n=1,\dots,N} z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) = 1. \end{aligned} \tag{7}$$

14. We now release the constraints $\min_{n=1,\dots,N} z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) = 1$ to $z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1$, for all n and hence yields another optimization problem:

$$\begin{aligned} \min_{b,\boldsymbol{w}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w}, \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1; \quad n = 1, \dots, N. \end{aligned} \tag{8}$$

15. Quadratic programming (QP):

    (a) QP is a mathematical optimization problem that deals with finding the minimum (or maximum) of a quadratic function subject to linear constraints.

    (b) QP is widely used in various fields such as finance (portfolio optimization), engineering (control theory, structural optimization), operations research, and many other areas where optimization problems arise.

    (c) It's an essential tool for solving complex optimization problems efficiently.

16. Use QP to find $(b, \boldsymbol{w})$ of SVM.

    (a) Solution from the process of QP:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{Q}\boldsymbol{u} + \boldsymbol{p}^\top \boldsymbol{u}, \\ \text{subject to} \quad & \boldsymbol{a}_m \boldsymbol{u} \geq c_m; \quad m = 1, \dots, M. \end{aligned} \tag{9}$$

    (b) From (8) and (9), by letting

$$\begin{aligned} \boldsymbol{u} &= (b, \boldsymbol{w})^\top; \\ \boldsymbol{Q} &= \text{diag}(0, \boldsymbol{I}); \\ \boldsymbol{p} &= \boldsymbol{0}; \\ M &= N; \\ c_m &= 1; \quad n = 1, \dots, N; \\ \boldsymbol{a}_n &= y_n(1, \boldsymbol{x}_n^\top); \quad n = 1, \dots, N. \end{aligned}$$

    (c) This is a linear **hard-margin SVM**.

3.4.2 | Dual SVM

1. The linear SVM:

$$\begin{aligned} \min_{b,\boldsymbol{w}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1; \quad n = 1, \ldots, N, \end{aligned}$$

2. Solution from the method of Lagrange multipliers:

$$\mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n (1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)),$$

where

$$\text{SVM} = \min_{b,\boldsymbol{w}} \left(\max_{\alpha_n \geq 0} \mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\alpha}) \right)$$

3. Why

$$\text{SVM} = \min_{b,\boldsymbol{w}} \left(\max_{\alpha_n \geq 0} \left(\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n (1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)) \right) \right)?$$

   (a) Can $z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) < 1$ be happened?

   (b) If yes, $1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) > 0$, then $\alpha_n = \infty$.

   (c) This cannot be a solution of SVM.

   (d) If $1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) < 0$, $\alpha_n = 0$.

   (e) Hence $\alpha_n > 0$ can only be happened on $1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) = 0$ (complementary slackness).

4. **Complementary slackness**:

   (a) It says that if a dual variable (i.e., α_n) is greater than zero (slack) then the corresponding primal constraint (i.e., $1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)$) must be an equality (tight).

   (b) It also says that if the primal constraint is slack then the corresponding dual variable is tight (or zero.)

   (c) This property is a fundamental aspect of duality in optimization and plays a key role in understanding the relationships between primal and dual solutions in optimization problems.

5. In conclusion,

$$\text{SVM} = \min_{b,\boldsymbol{w}} \left(\max_{\text{all } \alpha_n \geq 0} \mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\alpha}) \right)$$

   (a) For any violating $(b, \boldsymbol{w})$ such that $1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) > 0$ for some n, we have

$$\max_{\alpha_n \geq 0} \left(\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n (1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)) \right) \to \infty,$$

   since $\alpha_n = \infty$ shall be chosen in this case.

   (b) For any feasible $(b, \boldsymbol{w})$ such that $1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \leq 0$ for all n, we have

$$\max_{\alpha_n \geq 0} \left(\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n (1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)) \right) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w},$$

   since $\alpha_n(1 - y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)) = 0$; for all n, shall be chosen in this case.

   (c) We are now exactly minimizing $\boldsymbol{w}^\top \boldsymbol{w}$ under constraints $1 - z_n(\boldsymbol{w}^\top \boldsymbol{z}_n + b) \leq 0$ for all n.

6. Under **strong duality**:

$$\text{SVM} = \min_{b,\boldsymbol{w}} \left(\max_{\alpha_n \geq 0} \mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\alpha}) \right) = \max_{\alpha_n \geq 0} \left(\min_{b,\boldsymbol{w}} \mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\alpha}) \right).$$

7. Simplify the dual problem: under the strong duality,

$$\begin{aligned}\text{SVM} &= \max_{\alpha_n \geq 0} \left(\min_{b,\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n (1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)) \right) \\ &= \max_{\alpha_n \geq 0, \sum z_n \alpha_n = 0} \left(\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n (1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n)) \right),\end{aligned}$$

since

$$\frac{\partial \mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\alpha})}{\partial b} = -\sum_{n=1}^{N} \alpha_n z_n = 0.$$

8. Further simplify the dual problem: under the strong duality and by

$$\frac{\partial \mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\alpha})}{\partial w_i} = w_i - \sum_{n=1}^{N} \alpha_n z_n x_{n,i} = 0 \quad \Rightarrow \quad \boldsymbol{w} = \sum_{n=1}^{N} \alpha_n z_n \boldsymbol{x}_n,$$

and hence

$$\begin{aligned}\text{SVM} &= \max_{\alpha_n \geq 0, \sum z_n \alpha_n = 0} \left(\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n (1 - z_n(\boldsymbol{x}_n{}^\top \boldsymbol{w})) \right) \\ &= \max_{\alpha_n \geq 0, \sum z_n \alpha_n = 0, \boldsymbol{w} = \sum \alpha_n z_n \boldsymbol{x}_n} \left(\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n - \boldsymbol{w}^\top \boldsymbol{w} \right) \\ &= \max_{\alpha_n \geq 0, \sum z_n \alpha_n = 0, \boldsymbol{w} = \sum \alpha_n z_n \boldsymbol{x}_n} \left(-\frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n z_n \boldsymbol{x}_n \right\|^2 + \sum_{n=1}^{N} \alpha_n \right).\end{aligned}$$

9. This suggests that conditions for primal-dual optimal $(b, \boldsymbol{w}, \boldsymbol{\alpha})$ are:
   (a) Primal feasible:
   $$z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1;$$
   (b) Dual feasible:
   $$\alpha_n \geq 0;$$
   (c) Dual-inner optimal:
   $$\sum z_n \alpha_n = 0, \text{ and } \boldsymbol{w} = \sum \alpha_n z_n \boldsymbol{x}_n;$$
   (d) Primal-inner optimal (complementary slackness):
   $$\alpha_n (1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)) = 0.$$
   (e) The above conditions are called Karush-Kuhn-Tucker (KKT) conditions and are necessary for optimality.
   (f) We will use KKT to solve $(b, \boldsymbol{w})$ from optimal $\boldsymbol{\alpha}$.

10. The dual problem:

$$\max_{\alpha_n \geq 0, \sum z_n \alpha_n = 0, \boldsymbol{w} = \sum \alpha_n z_n \boldsymbol{x}_n} \left(-\frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n z_n \boldsymbol{x}_n \right\|^2 + \sum_{n=1}^{N} \alpha_n \right)$$

11. The equivalent standard dual SVM problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m z_n z_m \boldsymbol{x}_n^\top \boldsymbol{x}_m - \sum_{n=1}^{N} \alpha_n, \\ \text{subject to} \quad & \sum_{n=1}^{N} z_n \alpha_n = 0; \\ & \alpha_n \geq 0; \quad \text{for } n = 1, \dots, N, \end{aligned} \tag{10}$$

which has N variables and $N+1$ constraints, and is a convex quadratic programming problem.

12. Again we can apply the process of QP given in (9):

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{Q}\boldsymbol{u} + \boldsymbol{p}^\top \boldsymbol{u}, \\ \text{subject to} \quad & \boldsymbol{a}_m \boldsymbol{u} \geq c_m; \quad m = 1, \dots, M. \end{aligned}$$

to solve (10) by setting:

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{\alpha}; \\ \boldsymbol{Q} &= \{q_{n,m}\}; \quad q_{n,m} = z_n z_m \boldsymbol{x}_n^\top \boldsymbol{x}_m; \\ \boldsymbol{p} &= -\mathbf{1}_N; \\ \boldsymbol{a}_{0,1} &= \boldsymbol{z}; \quad \text{with } c_{0,1} = 0 \text{ and } \boldsymbol{z} = (z_1, \dots, z_N)^\top; \\ \boldsymbol{a}_{0,2} &= -\boldsymbol{z}; \quad \text{with } c_{0,2} = 0; \\ \boldsymbol{a}_n &= (0, \dots, 0, a_{n,n}, 0, \dots, 0)^\top; \quad a_{n,n} = 1 \text{ with } c_n = 0, \quad n = 1, \dots, N. \end{aligned}$$

13. By the KKT conditions, we can obtain $\boldsymbol{w}$ easily. For the optimal b, for any $\alpha_n > 0$, $b = z_n - \boldsymbol{w}^\top \boldsymbol{x}_n$.

    (a) By KKT conditions, we have $\boldsymbol{w} = \sum \alpha_n z_n \boldsymbol{x}_n$ and for complementary slackness condition: $\alpha_n(1 - z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)) = 0$; $n = 1, \dots, N$,

$$b = z_n - \boldsymbol{w}^\top \boldsymbol{x}_n, \quad \text{with } \alpha_n > 0,$$

    (b) It is also interesting to note that when $\alpha_n > 0$, $z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) = 1$ which are points on the margin boundary and are support vectors.

    (c) Hence by letting $\mathcal{I} = \{s \in \{1, \dots, N\} : \alpha_s > 0\}$, we can obtain

$$b = \frac{1}{|\mathcal{I}|} \sum_{s \in \mathcal{I}} z_s - \boldsymbol{w}^\top \boldsymbol{x}_s, \tag{11}$$

    where $\mathcal{I}$ is the set of support vectors: $z_s(\boldsymbol{w}^\top \boldsymbol{x}_n + b) = 1$; $s \in \mathcal{I}$.

    (d) Or alternatively, by letting $\mathcal{I}^+ = \{s \in \mathcal{I} : z_s = +1\}$ and $\mathcal{I}^- = \{s \in \mathcal{I} : z_s = -1\}$,

$$b = -\frac{1}{2}\left(\min_{s \in \mathcal{I}^+} \boldsymbol{w}^\top \boldsymbol{x}_s + \max_{s \in \mathcal{I}^-} \boldsymbol{w}^\top \boldsymbol{x}_s \right). \tag{12}$$

14. Compare the dual hard-margin SVM to the primal hard-margin SVM.

    (a) Primal linear SVM (suitable for small d):

$$\begin{aligned} \min_{b, \boldsymbol{w}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1; \quad n = 1, \dots, N. \end{aligned}$$

        i. $d + 1$ variables;
        ii. N constraints;

    (b) Dual SVM (suitable for small N):

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} - \mathbf{1}^\top \boldsymbol{\alpha}; \quad \boldsymbol{Q} = \{q_{n,m}\};\ q_{n,m} = z_n z_m \boldsymbol{x}_n^\top \boldsymbol{x}_m, \\ \text{subject to} \quad & \boldsymbol{z}^\top \boldsymbol{\alpha} = 0; \\ & \alpha_n \geq 0; \quad \text{for } n = 1, \dots, N, \end{aligned}$$

        i. N variables;
        ii. $N + 1$ constraints;
        iii. Support vectors with $\alpha_n > 0$.

3.4.3 | Kernel SVM

1. A nonlinear decision boundary can be obtained via **Transformation** + **Linear Models**.
2. Let $\boldsymbol{\Phi} : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ and $\tilde{\boldsymbol{x}}_n = \boldsymbol{\Phi}(\boldsymbol{x}_n)$; $n = 1, \dots, N$.
3. For the dual SVM, we need compute $\tilde{\boldsymbol{x}}_n^\top \tilde{\boldsymbol{x}}_m = \boldsymbol{\Phi}(\boldsymbol{x}_n)^\top \boldsymbol{\Phi}(\boldsymbol{x}_m)$; for $n, m = 1, \dots, N$, which we want to calculate faster than $O(\tilde{d})$.
4. The goal is to do SVM without dependence on $\tilde{d}$ (**kernel trick**).
5. Consider a 2nd order polynomial transform:

$$\boldsymbol{\Phi}_2(\boldsymbol{x}) = (1, x_1, x_2, \dots, x_d, x_1^2, x_1x_2, \dots, x_1x_d, x_2x_1, x_2^2, \dots, x_2x_d, \dots, x_dx_1, x_dx_2, \dots, x_d^2),$$

   and hence we have

$$\begin{aligned}\boldsymbol{\Phi}_2(\boldsymbol{x})^\top \boldsymbol{\Phi}_2(\boldsymbol{x}^*) &= 1 + \sum_{i=1}^{d} x_i x_i^* + \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j x_i^* x_j^* \\ &= 1 + \sum_{i=1}^{d} x_i x_i^* + \sum_{i=1}^{d} x_i x_i^* \sum_{j=1}^{d} x_j x_j^* \\ &= 1 + \boldsymbol{x}^\top \boldsymbol{x}^* + (\boldsymbol{x}^\top \boldsymbol{x}^*)^2,\end{aligned}$$

   which suggests that for $\boldsymbol{\Phi}_2$ transform, the inner product can be carefully done in $O(d)$ instead of $O(\tilde{d} = d^2)$.
6. We call this transform and inner product of $\boldsymbol{x}$ the kernel function (the 2nd order polynomial kernel function):

$$K_{\boldsymbol{\Phi}_2}(\boldsymbol{x}, \boldsymbol{x}^*) \equiv \boldsymbol{\Phi}_2(\boldsymbol{x})^\top \boldsymbol{\Phi}_2(\boldsymbol{x}^*), \tag{13}$$

   where $K_{\boldsymbol{\Phi}_2}(\boldsymbol{x}, \boldsymbol{x}^*) = 1 + (\boldsymbol{x}^\top \boldsymbol{x}^*) + (\boldsymbol{x}^\top \boldsymbol{x}^*)^2$.
7. For kernel function:

$$K_{\boldsymbol{\Phi}}(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{\Phi}(\boldsymbol{x}_n)^\top \boldsymbol{\Phi}(\boldsymbol{x}_m),$$

   the dual problem on the transform function $\boldsymbol{\Phi}$:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}_\Phi \boldsymbol{\alpha} - \mathbf{1}^\top \boldsymbol{\alpha};, \\ \text{subject to} \quad & \boldsymbol{z}^\top \boldsymbol{\alpha} = 0; \\ & \alpha_n \geq 0; \quad \text{for } n = 1, \dots, N, \end{aligned}$$

   where $\boldsymbol{Q}_\Phi = \{q_{n,m}\}$ and $q_{n,m} = z_n z_m K_{\boldsymbol{\Phi}}(\boldsymbol{x}_n, \boldsymbol{x}_m)$.
8. Polynomial kernel function:

$$K_{\boldsymbol{\Phi}_q}(\boldsymbol{x}, \boldsymbol{x}^*) = (\zeta + \gamma \boldsymbol{x}^\top \boldsymbol{x}^*)^q; \quad \gamma > 0, \zeta \geq 0,$$

   which is commonly applied for polynomial SVM.
9. Gaussian kernel function:

$$K_G(\boldsymbol{x}, \boldsymbol{x}^*) = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}^*\|^2), \quad \gamma > 0,$$

   which is a kernel of infinite dimensional transform and for $d = 1$ (i.e. $\boldsymbol{x} = x$),

$$\boldsymbol{\Phi}(x) = \exp(-x^2) \cdot \left(1, \sqrt{\frac{2}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \dots\right),$$

10. Hence for some kernel function $K(\cdot, \cdot)$, the quadratic coefficients $q_{n,m}$ can be rewritten as:

$$q_{n,m} = z_n z_m \tilde{\boldsymbol{x}}_n^\top \tilde{\boldsymbol{x}}_m = z_n z_m K(\boldsymbol{x}_n, \boldsymbol{x}_m),$$

and hence the optimal b (for some $\alpha_s > 0$ such that $1 - z_s(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_s + b) = 0$).

$$b = z_s - \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_s = z_s - \Big(\sum_{n=1}^{N} \alpha_n z_n \tilde{\boldsymbol{x}}_n \Big)^\top \tilde{\boldsymbol{x}}_s$$

$$= z_s - \sum_{n=1}^{N} \alpha_n z_n K(\boldsymbol{x}_n, \boldsymbol{x}_s),$$

$$\boldsymbol{w}^\top \Phi(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n z_n \tilde{\boldsymbol{x}}_n^\top \tilde{\boldsymbol{x}} = \sum_{n=1}^{N} \alpha_n z_n K(\boldsymbol{x}_n, \boldsymbol{x}).$$

11. The classifier of kernel svm, $g_{\text{SVM}}(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}) + b)$, we don't need to know $\boldsymbol{w}$ and the hyperplane classifier can be a mystery and this is called the kernel trick.

12. For the Gaussian SVM, the hypothesis is

$$g_{\text{SVM}}(\boldsymbol{x}) = \text{sign}\Big(\sum_{\text{SV}} \alpha_n z_n K(\boldsymbol{x}_n, \boldsymbol{x}) + b \Big)$$

$$= \text{sign}\Bigg(\sum_{\text{SV}} \alpha_n z_n \exp\Big(-\gamma \|\boldsymbol{x} - \boldsymbol{x}_n\|^2 \Big) \Bigg) + b \Bigg),$$

    which is a linear combination of Gaussian centered at SV's $\boldsymbol{x}_n$, and is also called Radial Basis Function (RBF) kernel.

13. The Gaussian SVM find α_n to combine Gaussians centered at $\boldsymbol{x}_n$ and achieve large margin in infinite dimensional space.

3.4.4 | Soft-Margin SVM

1. If the dataset is linearly separable in high-dimensional transformed feature space, the dataset is then separable nonlinearly in the original feature space.

2. If we ignore the problem of overfitting, theoretically, the Gaussian kernel SVM can transform any dataset into a nonlinearly separable space.

3. The Gaussian kernel allows for a highly flexible decision boundary by implicitly mapping the input features into a potentially infinite-dimensional space.

4. However, if always insisting on separable, SVM tends to overfitting to noise.

5. Can SVM automatically detect that some data points could be outlier, or some data points are able to be ignored for a parsimonious decision boundary?

6. Hard-margin SVM:

$$\begin{aligned} \min_{b,\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w}, \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b) \geq 1. \end{aligned}$$

7. Tolerance on the noise by allowing the data points being incorrectly classified.

8. Record the margin violation by ξ_n; $n = 1, \dots, N$.

9. Soft-margin SVM:

$$\begin{aligned} \min_{b,\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{n=1}^{N} \xi_n \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b) \geq 1 - \xi_n, \\ & \xi_n \geq 0; \quad n = 1, \dots, N, \end{aligned}$$

    where C is the trade-off of large margin and margin violation:

(a) Large C: less margin violation;

(b) Small C: large margin.

10. Quadratic programming with $\tilde{d}+1+N$ variables and $2N$ constraints:

$$\begin{aligned} \min_{b,\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{n=1}^{N}\xi_n \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b) \geq 1-\xi_n, \\ & \xi_n \geq 0; \quad n=1,\dots,N, \end{aligned}$$

11. The dual problem: $\text{SVM} = \min_{b,\boldsymbol{w},\boldsymbol{\xi}}\Big(\max_{\alpha_n\geq 0,\gamma_n\geq 0}\mathcal{L}(b,\boldsymbol{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\gamma})\Big)$,

$$\begin{aligned} \mathcal{L}(b,\boldsymbol{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\gamma}) = \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} + C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}\gamma_n(-\xi_n) \\ + \sum_{n=1}^{N}\alpha_n(1-\xi_n - z_n(\boldsymbol{w}^\top\tilde{\boldsymbol{x}}_n + b)). \end{aligned}$$

12. Simplify the soft-margin dual problem: under the strong duality:

$$\begin{aligned} \text{SVM} &= \max_{\alpha_n\geq 0,\gamma_n\geq 0}\Big(\min_{b,\boldsymbol{w},\boldsymbol{\xi}}\frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} + C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}\gamma_n(-\xi_n) \\ &\quad + \sum_{n=1}^{N}\alpha_n(1-\xi_n - z_n(\boldsymbol{w}^\top\tilde{\boldsymbol{x}}_n+b))\Big) \\ &= \max_{0\leq\alpha_n\leq C,\gamma_n=C-\alpha_n}\Big(\min_{b,\boldsymbol{w}}\frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} + \sum_{n=1}^{N}\alpha_n(1-z_n(\boldsymbol{w}^\top\tilde{\boldsymbol{x}}_n+b))\Big), \end{aligned}$$

where note that γ_n and ξ_n are removed, and the last equation follows by

$$\frac{\partial\mathcal{L}}{\partial\xi_n} = C-\alpha_n-\gamma_n = 0 \Rightarrow \gamma_n = C-\alpha_n \geq 0.$$

gives that $0\leq\alpha_n\leq C$.

13. Further simplify the soft-margin dual problem: under the strong duality and by

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial b} = 0 \;\Rightarrow\quad & \sum_{n=1}^{N}\alpha_n z_n = 0, \\ \frac{\partial\mathcal{L}}{\partial w_i} = 0 \;\Rightarrow\quad & \boldsymbol{w} = \sum_{n=1}^{N}\alpha_n z_n\tilde{\boldsymbol{x}}_n, \end{aligned}$$

14. The soft-margin dual SVM:

$$\text{SVM} = \max_{0\leq\alpha_n\leq C,\gamma_n=C-\alpha_n,\sum z_n\alpha_n=0,\boldsymbol{w}=\sum\alpha_n z_n\tilde{\boldsymbol{x}}_n}\Big\{-\frac{1}{2}\|\sum_{n=1}^{N}\alpha_n z_n\tilde{\boldsymbol{x}}_n\|^2 + \sum_{n=1}^{N}\alpha_n\Big\}.$$

15. QP with N variables and $2N+1$ constraints:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m z_n z_m\tilde{\boldsymbol{x}}_n^\top\tilde{\boldsymbol{x}}_m - \sum_{n=1}^{N}\alpha_n, \\ \text{subject to} \quad & \sum_{n=1}^{N}z_n\alpha_n = 0,\ 0\leq\alpha_n\leq C;\ n=1,\dots,N, \\ \text{implicity} \quad & \boldsymbol{w} = \sum_{n=1}^{N}\alpha_n z_n\tilde{\boldsymbol{x}}_n,\ \gamma_n = C-\alpha_n;\ n=1,\dots,N, \end{aligned}$$

which is only different from the hard-margin SVM with an upper bound on α_n.

16. We are now able to fit the kernel soft-margin SVM which is similar to the hard-margin kernel SVM: QP with N variables and $2N + 1$ constraints,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m z_n z_m K_\Phi(\boldsymbol{x}_n,\boldsymbol{x}_m) - \sum_{n=1}^{N}\alpha_n, \\ \text{subject to} \quad & \sum_{n=1}^{N} z_n\alpha_n = 0,\ 0 \le \alpha_n \le C;\ n = 1,\dots,N, \\ \text{implicity} \quad & \boldsymbol{w} = \sum_{n=1}^{N}\alpha_n z_n \tilde{\boldsymbol{x}}_n,\ \gamma_n = C - \alpha_n;\ n = 1,\dots,N, \end{aligned}$$

17. The hypothesis hyperplane of soft-margin kernel svm:

$$g_{\text{svm}}(\boldsymbol{x}) = \text{sign}\Big(\sum_{n=1}^{N}\alpha_n z_n K_\Phi(\boldsymbol{x}_n,\boldsymbol{x}) + b\Big),$$

where b can be solved by the complementary slackness.

(a) To find optimal b (by the complementary slackness):

$$\begin{aligned} \alpha_n(1 - \xi_n - z_n(\boldsymbol{w}^\top\tilde{\boldsymbol{x}}_n + b)) = 0, \\ (C - \alpha_n)\xi_n = 0. \end{aligned}$$

i. For support vectors (with $\alpha_s > 0$):

$$b = z_s - z_s\xi_s - \boldsymbol{w}^\top\tilde{\boldsymbol{x}}_s.$$

ii. For **free support vectors** (with $0 < \alpha_s < C$):

$$\xi_s = 0 \Rightarrow b = z_s - \boldsymbol{w}^\top\tilde{\boldsymbol{x}}_s.$$

(b) Finally, we can solve unique b with **free support vectors** $(\boldsymbol{x}_s, y_s)$:

$$b = z_s - \sum_{\text{SV indices } n}\alpha_n z_n K_{\boldsymbol{\Phi}}(\boldsymbol{x}_n,\boldsymbol{x}_s).$$

(c) Further, let $\mathcal{I} = \{s \in \{1,\dots,N\} : 0 < \alpha_s < C\}$ be the set of all free support vectors,

$$b = \frac{1}{|\mathcal{I}|}\sum_{s\in\mathcal{I}}\left(z_s - \sum_{\text{SV indices } n}\alpha_n z_n K_{\boldsymbol{\Phi}}(\boldsymbol{x}_n,\boldsymbol{x}_s)\right).$$

(d) Or alternatively, for $\mathcal{I}^+ = \{s \in \mathcal{I} : z_s = +1\}$ and $\mathcal{I}^- = \{s \in \mathcal{I} : z_s = -1\}$,

$$b = -\frac{1}{2}\left(\min_{s\in\mathcal{I}^+}\sum_{\text{SV indices } n}\alpha_n z_n K_{\boldsymbol{\Phi}}(\boldsymbol{x}_n,\boldsymbol{x}_s) + \max_{s\in\mathcal{I}^-}\sum_{\text{SV indices } n}\alpha_n z_n K_{\boldsymbol{\Phi}}(\boldsymbol{x}_n,\boldsymbol{x}_s)\right)$$

18. Complementary slackness:

$$\begin{aligned} \alpha_n(1 - \xi_n - z_n(\boldsymbol{w}^\top\tilde{\boldsymbol{x}}_n + b)) = 0, \\ (C - \alpha_n)\xi_n = 0. \end{aligned}$$

(a) Non-SV ($\alpha_n = 0$): $\xi_n = 0$ — these are not support vectors lying on or outside the margin and are correctly specified.

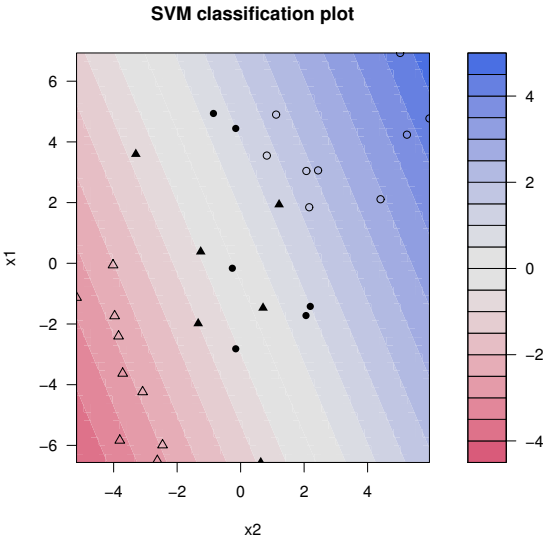
(b) SV ($0 < \alpha_n < C$): $\xi_n = 0$ — these are support vectors lying on the margin and are correctly specified.

(c) Bounded SV ($\alpha_n = C$): $\xi_n = 1 - z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b) \geq 0$ —these are support vectors and generally lie within the margin or are incorrectly specified. It is seldom on the margin for $\xi_n = 0$.

(d) If $\xi_n = 0$, it implies that the samples $(y_n, \tilde{\boldsymbol{x}}_n)$ are correctly specified and lies either on the margin (if $z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b) = 1$, and $\alpha_n = 0$) or away from the margin (if $z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b) > 1$ and $\alpha_n = 0$).

(e) If $\xi_n > 0$, it implies that the samples $(z_n, \tilde{\boldsymbol{x}}_n)$ may fall inside the margin or are misclassified, but they are still considered support vectors.

(f) α_n and ξ_n can be used for data analysis.



3.4.5 | Weighted SVM

1. For each vector $\tilde{\boldsymbol{x}}_n$, suppose that the classification error is weighted by κ_n when the sample point $(z_n, \tilde{\boldsymbol{x}}_n)$ is incorrectly classified.

2. This is equivalent to consider an extra amount of sample points, say κ_n (without loss of generality, we assume κ_n are positive integers), that are overlapping on $\tilde{\boldsymbol{x}}_n$.

3. In other words, we now have the pseudo samples that:

   (a) Pseudo sample size: $\tilde{N} = \sum_{n=1}^{N} \kappa_n$.

   (b) Pseudo responses: for $n = 1, \ldots, N$ and $j = 1, \ldots, \kappa_n$,

   $$z_{n,j} = z_n \in \{+1, -1\}.$$

   (c) Pseudo covariates: for $n = 1, \ldots, N$ and $j = 1, \ldots, \kappa_n$, $\tilde{\boldsymbol{x}}_{n,j} = \tilde{\boldsymbol{x}}_n$.

   (d) Linear classifier: for $n = 1, \ldots, N$ and $j = 1, \ldots, \kappa_n$, $h(\tilde{\boldsymbol{x}}_{n,j}) = \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_{n,j} + b$.

4. Denote the margin violation of cluster samples $z_{n,j}; j = 1, \ldots, \kappa_n$ by $\xi_{n,j}$ and $\xi_{n,j} = \xi_n$ for $j = 1, \ldots, \kappa_n$.

5. Let $\boldsymbol{\xi} = \{\xi_{n,j}\}$ be the $\tilde{N} \times 1$ margin violation vector.

6. The optimization problem of the soft-SVM is

$$\begin{aligned} \min_{b,\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{n=1}^{N} \sum_{j=1}^{\kappa_n} \xi_{n,j} \\ \text{subject to} \quad & z_{n,j}(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_{n,j} + b) \geq 1 - \xi_{n,j}, \\ & \xi_{n,j} \geq 0; \quad n = 1, \ldots, N, \; j = 1, \ldots, \kappa_n, \end{aligned}$$

   where C is the trade-off of large margin and margin violation.

7. Since $z_{n,j} = z_n$, $\tilde{\boldsymbol{x}}_{n,j} = \tilde{\boldsymbol{x}}_n$ and $\xi_{n,j} = \xi_n$, the optimization problem can then be re-written as:

$$\begin{aligned} \min_{b,\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{n=1}^{N} \kappa_n \xi_n \\ \text{subject to} \quad & z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b) \geq 1 - \xi_n, \\ & \xi_n \geq 0; \quad n = 1, \ldots, N, \end{aligned}$$

8. We then have the dual problem:

$$\min_{b,\boldsymbol{w},\boldsymbol{\xi}} \left(\max_{\alpha_n \geq 0, \gamma_n \geq 0} \mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) \right),$$

where

$$\mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{n=1}^{N} \kappa_n \xi_n + \sum_{n=1}^{N} \gamma_n (-\xi_n) + \sum_{n=1}^{N} \alpha_n \big(1 - \xi_n - z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b)\big).$$

9. Under the strong duality, the dual problem is equivalent to

$$\begin{aligned} &\max_{\alpha_n \geq 0, \gamma_n \geq 0} \left(\min_{b,\boldsymbol{w},\boldsymbol{\xi}} \mathcal{L}(b, \boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) \right) \\ = &\max_{\alpha_n \geq 0, \gamma_n \geq 0} \left(\min_{b,\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \kappa_n \xi_n C + \sum_{n=1}^{N} \beta_n (-\xi_n) \right. \\ &\left. + \sum_{n=1}^{N} \alpha_n \big(1 - \xi_n - z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b)\big) \right) \\ = &\max_{\alpha_n \geq 0, \gamma_n \geq 0} \left(\min_{b,\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n \big(1 - z_n(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b)\big) \right) \end{aligned}$$

where note that γ_n and ξ_n are removed, and the last equality follows by

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = \kappa_n C - \gamma_n - \alpha_n = 0,$$

which gives $\gamma_n = \kappa_n C - \alpha_n \geq 0$ and $0 \leq \alpha_n \leq \kappa_n C$.

10. Note that

$$\frac{\partial \mathcal{L}}{\partial b} = 0, \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \boldsymbol{0}$$

implies that

$$\sum_{n=1}^{N} \alpha_n z_n = 0, \tag{14}$$

and

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n z_n \tilde{\boldsymbol{x}}_n. \tag{15}$$

11. Under the strong duality and by (14) and (15), we can further simplify the dual problem as:

$$\max_{0 \leq \alpha_n \leq \kappa_n C,\ \sum_{n=1}^{N} \alpha_n z_n = 0,\ \boldsymbol{w} = \sum_{n=1}^{N} \alpha_n z_n \tilde{\boldsymbol{x}}_n} \left\{ -\frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n z_n \tilde{\boldsymbol{x}}_n \right\|^2 + \sum_{n=1}^{N} \alpha_n \right\}.$$

12. We then have the standard soft-margin SVM dual problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m z_n z_m \tilde{\boldsymbol{x}}_n^\top \tilde{\boldsymbol{x}}_m - \sum_{n=1}^{N} \alpha_n, \\ \text{subject to} \quad & \sum_{n=1}^{N} z_n \alpha_n = 0,\ 0 \leq \alpha_n \leq \kappa_n C;\ n = 1, \ldots, N, \\ \text{implicity} \quad & \boldsymbol{w} = \sum_{n=1}^{N} \alpha_n z_n \tilde{\boldsymbol{x}}_n,\ \gamma_n = \kappa_n C - \alpha_n;\ n = 1, \ldots, N. \end{aligned} \tag{16}$$

13. Let $\mathcal{I} = \{n \in \{1, \ldots, N\} : 0 < \alpha_n \leq \kappa_n C\}$ denote the set of all support vectors.

14. We then have the classifier $h(\tilde{\boldsymbol{x}}_n) = \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + b$ with

$$\boldsymbol{w} = \sum_{n \in \mathcal{I}} \alpha_n z_n \tilde{\boldsymbol{x}}_n.$$

and b being obtained by free support vectors:

$$b = -\frac{1}{2}\left(\min_{n \in \mathcal{I}^+} \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n + \max_{n \in \mathcal{I}^-} \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n \right)$$

where $\mathcal{I}^+ = \{n \in \mathcal{I} : z_n = +1 \text{ and } 0 < \alpha_n < \kappa_n C\}$ and $\mathcal{I}^- = \{n \in \mathcal{I} : z_n = -1 \text{ and } 0 < \alpha_n < \kappa_n C\}$ are sets of free support vectors that lie on the margin.

15. In summary:

    (a) The optimization difference between standard soft-margin SVM and weighted soft-margin SVM lies in how the upper bounds of the dual variables (i.e., α_n) are determined.

    (b) In a standard soft-margin SVM, the upper bound of the dual variables is fixed and determined solely by the penalty parameter C.

    (c) However, in a weighted soft-margin SVM, the upper bounds of the dual variables are adjusted by multiplying the penalty parameter C with the predefined weights γ_n associated with each data point.

    (d) This adjustment allows for differential treatment of individual data points based on their importance or significance, as determined by the weights assigned to them.

    (e) In other words, The significance of a vector (i.e., $\tilde{\boldsymbol{x}}_n$) in constructing the classifier (i.e., $\boldsymbol{w}$) increases with higher weights (i.e., κ_n) assigned to it.

    (f) This reflects the importance attributed to data points with higher weights, allowing them to contribute more substantially to the classifier's construction.

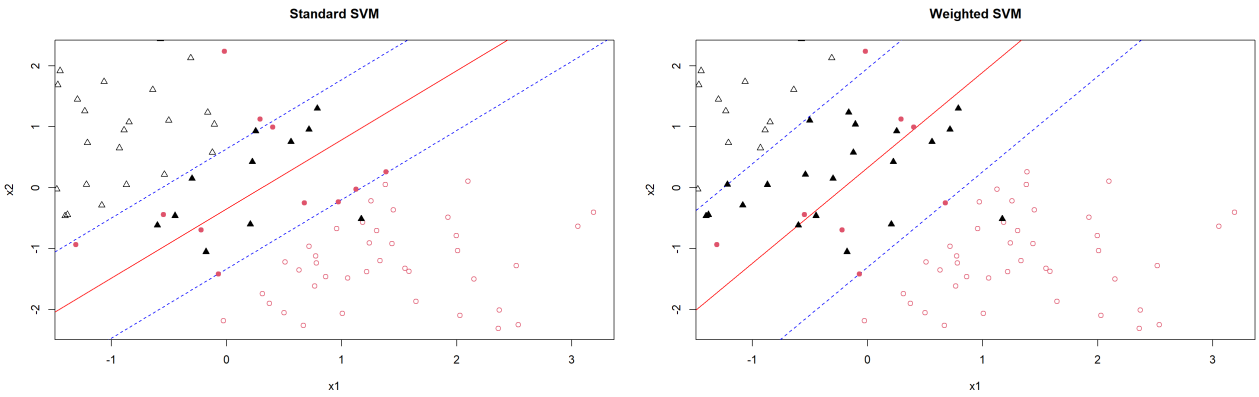
3.4.6 | Functions of (Weighted) Support Vector Machines in R/RStudio

1. Code in R:

```
# Load required library
library(WeightSVM)
# Generate example data
set.seed(123)
x1 = c(rnorm(50, mean = -1), rnorm(50, mean = 1))
x2 = c(rnorm(50, mean = 1), rnorm(50, mean = -1))
y = c(rep(-1, 50), rep(1, 50))
data = data.frame(x1 = x1, x2 = x2, y = factor(y))
# Fit standard soft-margin SVM
model = wsvm(x = data[, c("x1", "x2")], y = data$y, weight = rep(1, 100),
kernel="linear",cost=10,scale=FALSE)
cf = coef(model)
# plot data and separating hyperplane
plot(x2~x1,data=data[which(data[,3]==1),],col=y,pch=1,main="Standard SVM")
points(x2~x1,data=data[which(data[,3]==-1),],col=y,pch=2)
abline(-cf[1]/cf[3],-cf[2]/cf[3],col="red")
abline(-(cf[1] + 1)/cf[3], -cf[2]/cf[3], col = "blue",lty="dashed")
abline(-(cf[1] - 1)/cf[3], -cf[2]/cf[3], col = "blue",lty="dashed")
sv_data = data[rownames(model$SV),]
points(x2~x1,data=sv_data[which(sv_data[,3]==-1),],col=y,pch=17)
points(x2~x1,data=sv_data[which(sv_data[,3]==1),],col=y,pch=16)
# Define different weights for the positive and negative samples
```

```
weights = ifelse(y == 1, 3, 1)
# Fit weighted soft-margin SVM
modelw = wsvm(x = data[, c("x1", "x2")], y = data$y, weight = weights ,
kernel="linear",cost=10,scale=FALSE)
cf = coef(modelw)
# plot data and separating hyperplane
plot(x2~x1,data=data[which(data[,3]==1),],col=y,pch=1,main="Weighted SVM")
points(x2~x1,data=data[which(data[,3]==-1),],col=y,pch=2)
abline(-cf[1]/cf[3],-cf[2]/cf[3],col="red")
abline(-(cf[1] + 1)/cf[3], -cf[2]/cf[3], col = "blue",lty="dashed")
abline(-(cf[1] - 1)/cf[3], -cf[2]/cf[3], col = "blue",lty="dashed")
sv_data = data[rownames(modelw$SV),]
points(x2~x1,data=sv_data[which(sv_data[,3]==-1),],col=y,pch=17)
points(x2~x1,data=sv_data[which(sv_data[,3]==1),],col=y,pch=16)
```

2. **Loading Required Libraries**: The WeightSVM library provides functions for weighted SVM.
3. **Generating Example Data**: We create a synthetic dataset with two features (x1 and x2) and a binary response variable y. There are 50 samples each for class -1 and class $+1$.
4. **Fitting Standard Soft-Margin SVM**:
   (a) We fit a standard soft-margin SVM using the wsvm function from the **WeightSVM package**.
   (b) We specify the input features x, the response variable y, and set the weights to be equal (**rep(1, 100)**).
   (c) The kernel is set to **linear** and **the cost parameter (C) is set to 10**.
   (d) **scale = FALSE** ensures that the dataset is not standardized.
   (e) The coefficients of the SVM model are obtained using **coef(model)**.
5. **Plotting Data and Separating Hyperplane for Standard SVM**:
   (a) We plot the data points with different colors and shapes based on their class labels.
   (b) The separating hyperplane and the margin are plotted using the coefficients obtained from the model.
   (c) Support vectors are identified and plotted with different shapes.
6. **Defining Different Weights for Weighted Soft-Margin SVM**: We define different weights for the positive and negative samples. In this case, positive samples are weighted 3 times more than negative samples.
7. **Fitting Weighted Soft-Margin SVM**: We fit a weighted soft-margin SVM using the same procedure as the standard SVM, but with the specified weights.
8. **Plotting Data and Separating Hyperplane for Weighted SVM**:
   (a) Similar to the standard SVM, we plot the data points, separating hyperplane, margin, and support vectors for the weighted SVM.
   (b) The code demonstrates the effect of different weights on the decision boundary and margin of SVMs using the same dataset.
   (c) The decision boundary may change based on the weights assigned to different samples, impacting how the SVM model classifies new data points.
9. The performance of the two SVMs are visualized in the followings:



4 | THRESHOLD REGRESSION MODELS

4.1 | Threshold Boundary Regression (TBR) Models

1. Consider a dataset $\mathcal{D} = \{(y_n, \boldsymbol{x}_n)\}_{n=1}^N$, where $y_n \in \mathbb{R}$ is the continuous response and $\boldsymbol{x}_n \in \mathbb{R}^d$ is a d-dimensional covariate vector.

2. Traditional linear regression assumes that all observations follow a single linear model.

3. However, in many real-world applications, the relationship between the response and predictors may vary across different regions of the covariate space.

4. Threshold regression models provide a flexible framework to capture such **regime-switching behavior**, where the data-generating mechanism changes when a certain threshold variable crosses a specific value.

   (a) A common form of the threshold regression model is defined as:

$$y_n = \begin{cases} \boldsymbol{\beta}_h^\top \boldsymbol{x}_n + \epsilon_n, & \text{if } q_n > \alpha \\ \boldsymbol{\beta}_g^\top \boldsymbol{x}_n + \epsilon_n, & \text{if } q_n \le \alpha \end{cases}$$

   i. q_n is the threshold variable (often a component of $\boldsymbol{x}_n$ or an external variable);
   ii. α is the unknown threshold;
   iii. $\boldsymbol{\beta}_h$ and $\boldsymbol{\beta}_g$ are regression coefficients for the two regimes;
   iv. The error term ε_n is typically assumed to be independently and identically distributed with mean zero and finite variance.

   (b) The conventional threshold regression splits the sample based on the threshold condition and fits separate linear regressions in each region.

   (c) When $\boldsymbol{\beta}_h \neq \boldsymbol{\beta}_g$, the model allows for structural changes in the regression relationship at the threshold point α, making it particularly useful in economics, medicine, and environmental studies, where such discontinuities are common.

5. A key limitation of classical threshold regression is its reliance on a single threshold variable.

   (a) If q_n is uninformative, it may lead to incorrect sample partitioning and biased results.

   (b) To enhance flexibility, one should consider constructing linear or nonlinear classification rules based on multiple covariates for data-driven thresholding.

6. **Threshold boundary regression (TBR) models**:

$$y_n = \begin{cases} \boldsymbol{\beta}_h^\top \boldsymbol{x}_n + \epsilon_n, & \text{if } \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n > 0 \\ \boldsymbol{\beta}_g^\top \boldsymbol{x}_n + \epsilon_n, & \text{if } \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n \le 0 \end{cases}$$

   (a) $\tilde{\boldsymbol{x}}_n = \boldsymbol{\Phi}(\boldsymbol{x}_n)$ and $\boldsymbol{\Phi} : \{1\} \times \mathbb{R}^d \to \mathbb{R}^{\tilde{d}+1}$ is the transformation function with $\tilde{d} \ge d$;

(b) $\boldsymbol{\beta}_h \in \mathbb{R}^{d+1}$ and $\boldsymbol{\beta}_g \in \mathbb{R}^{d+1}$: regression parameter vectors;

(c) $\boldsymbol{w}$: threshold parameter vector and

$$\boldsymbol{w} \in \Theta_w = \{\boldsymbol{w} \in \mathbb{R}^{\tilde{d}+1} \,:\, \|\boldsymbol{w}\| = 1\}.$$

7. Loss function for TBR model:

$$\mathrm{E}_{\mathrm{in}}(\boldsymbol{w}; \boldsymbol{\beta}_h, \boldsymbol{\beta}_g) = \frac{1}{N} \sum_{n \in \mathcal{H}(\boldsymbol{w})} (y_n - \boldsymbol{\beta}_h^\top \boldsymbol{x}_n)^2 + \frac{1}{N} \sum_{n \in \mathcal{G}(\boldsymbol{w})} (y_n - \boldsymbol{\beta}_g^\top \boldsymbol{x}_n)^2.$$

(a) $\mathcal{H}(\boldsymbol{w}) = \{n \,:\, \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n > 0,\ n = 1, \ldots, N\}$: index set;

(b) $\mathcal{G}(\boldsymbol{w}) = \{n \,:\, \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n \le 0,\ n = 1, \ldots, N\}$: another index set;

(c) $\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n = 0$: the classification threshold boundary.

8. Based on $\boldsymbol{w}$, $\boldsymbol{\beta}_h$ and $\boldsymbol{\beta}_g$ can be estimated by least-squares (LS) method:

$$\begin{aligned} \hat{\boldsymbol{\beta}}_h(\boldsymbol{w}) &= \left(\sum_{n \in \mathcal{H}(\boldsymbol{w})} \boldsymbol{x}_n \boldsymbol{x}_n^\top \right)^{-1} \left(\sum_{n \in \mathcal{H}(\boldsymbol{w})} \boldsymbol{x}_n y_n \right), \\ \hat{\boldsymbol{\beta}}_g(\boldsymbol{w}) &= \left(\sum_{n \in \mathcal{G}(\boldsymbol{w})} \boldsymbol{x}_n \boldsymbol{x}_n^\top \right)^{-1} \left(\sum_{n \in \mathcal{G}(\boldsymbol{w})} \boldsymbol{x}_n y_n \right). \end{aligned} \tag{17}$$

9. Based on $\boldsymbol{\beta}_h$ and $\boldsymbol{\beta}_g$, how to obtain $\boldsymbol{w}$?

(a) Consider two indices sets:

$$\begin{aligned} \tilde{\mathcal{H}}(\boldsymbol{\beta}_h, \boldsymbol{\theta}_g) &= \{n \,:\, (y_n - \boldsymbol{\beta}_h^\top \boldsymbol{x}_n)^2 < (y_n - \boldsymbol{\beta}_g^\top \boldsymbol{x}_n)^2,\ n = 1, \ldots, N\}, \\ \tilde{\mathcal{G}}(\boldsymbol{\beta}_h, \boldsymbol{\theta}_g) &= \{n \,:\, (y_n - \boldsymbol{\beta}_h^\top \boldsymbol{x}_n)^2 \ge (y_n - \boldsymbol{\beta}_g^\top \boldsymbol{x}_n)^2,\ n = 1, \ldots, N\}. \end{aligned} \tag{18}$$

(b) $\tilde{\mathcal{H}}$ collects samples that are more align with the regression model corresponding to $\boldsymbol{\beta}_h$;

(c) $\tilde{\mathcal{G}}$ collects samples that are more align with the regression model corresponding to $\boldsymbol{\beta}_g$;

(d) If the two subsets $\tilde{\mathcal{H}}$ and $\tilde{\mathcal{G}}$ are linearly separable by a classifier, say $\boldsymbol{w}_0^\top \tilde{\boldsymbol{x}}_n = 0$, then $\boldsymbol{w}_0$ is the minimizer of $\mathrm{E}_{\mathrm{in}}(\boldsymbol{w}; \boldsymbol{\beta}_h, \boldsymbol{\beta}_g)$.

(e) In other words, the optimization problem of $\mathrm{E}_{\mathrm{in}}(\boldsymbol{w}; \boldsymbol{\beta}_h, \boldsymbol{\beta}_g)$ is equivalent to optimizing the following loss function:

$$\begin{aligned} \tilde{\mathrm{E}}_{\mathrm{in}}(\boldsymbol{w}; \boldsymbol{\beta}_h, \boldsymbol{\beta}_g) = &\frac{1}{N} \sum_{n \in \tilde{\mathcal{H}}(\boldsymbol{\beta}_h, \boldsymbol{\beta}_g) \setminus \mathcal{H}(\boldsymbol{w})} \left| (y_n - \boldsymbol{\beta}_h^\top \boldsymbol{x}_n)^2 - (y_n - \boldsymbol{\beta}_g^\top \boldsymbol{x}_n)^2 \right| \\ &+ \frac{1}{N} \sum_{n \in \tilde{\mathcal{G}}(\boldsymbol{\beta}_h, \boldsymbol{\beta}_g) \setminus \mathcal{G}(\boldsymbol{w})} \left| (y_n - \boldsymbol{\beta}_h^\top \boldsymbol{x}_n)^2 - (y_n - \boldsymbol{\beta}_g^\top \boldsymbol{x}_n)^2 \right|. \end{aligned} \tag{19}$$

10. $\tilde{\mathrm{E}}_{\mathrm{in}}(\boldsymbol{w}; \boldsymbol{\beta}_h, \boldsymbol{\beta}_g)$ is a weighted zero-one loss function:

(a) $\tilde{\mathrm{E}}_{\mathrm{in}}(\boldsymbol{w}; \boldsymbol{\beta}_h, \boldsymbol{\beta}_g)$ is non-differentiable;

(b) The optimization of $\tilde{\mathrm{E}}_{\mathrm{in}}(\boldsymbol{w}; \boldsymbol{\beta}_h, \boldsymbol{\beta}_g)$ is NP-hard;

(c) A surrogate algorithm such as **Weighted SVM** can be applied to minimizing $\tilde{\mathrm{E}}_{\mathrm{in}}(\boldsymbol{w}; \boldsymbol{\beta}_h, \boldsymbol{\beta}_g)$.

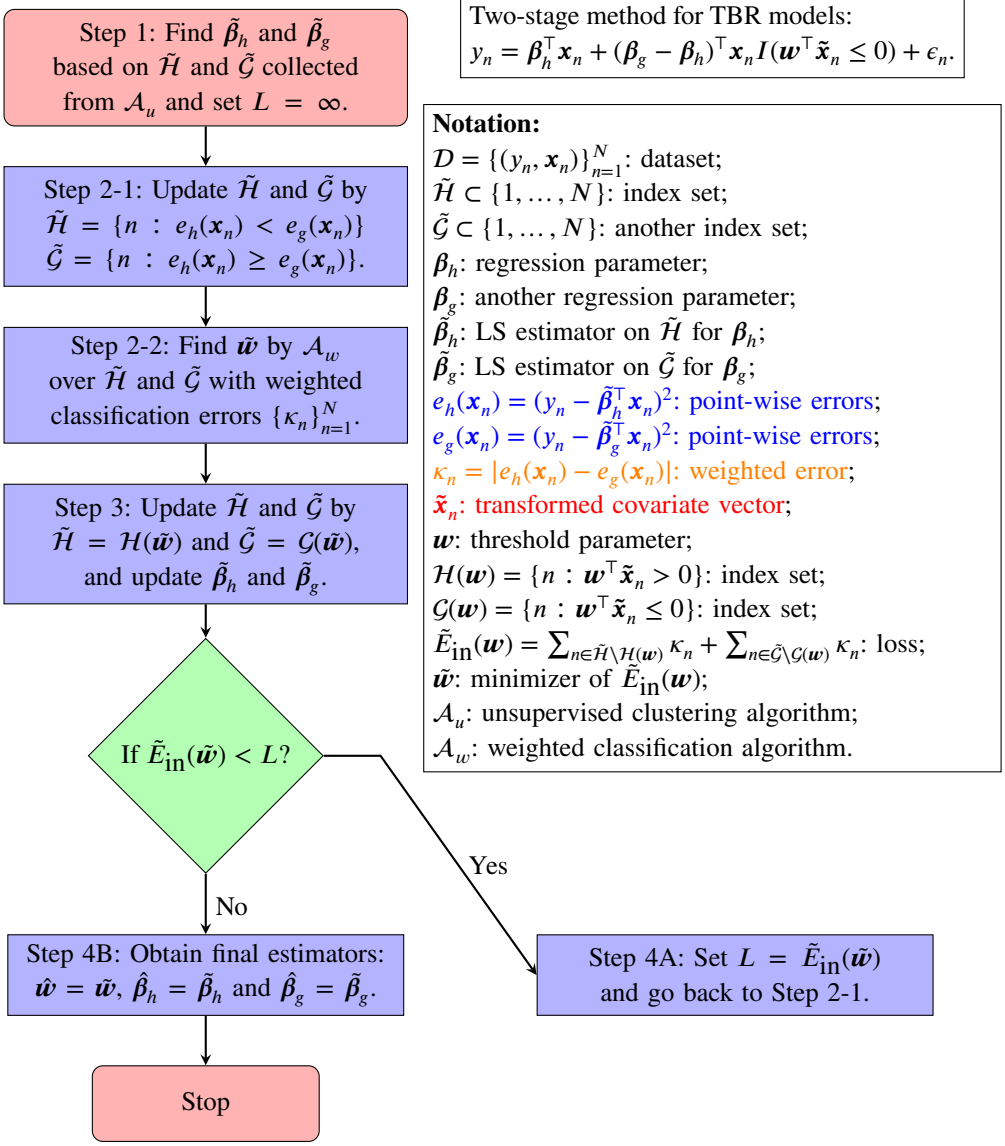
11. A two-stage iterative algorithm:

(a) We initialize by applying an unsupervised learning method (such as K-means) to obtain two clusters, say $\tilde{\mathcal{H}}$ and $\tilde{\mathcal{G}}$ to obtain regression parameter estimators, say $\tilde{\boldsymbol{\beta}}_h$ and $\tilde{\boldsymbol{\beta}}_g$.

(b) Based on $\tilde{\boldsymbol{\beta}}_h$ and $\tilde{\boldsymbol{\beta}}_g$, we obtain the minimizer of (19), say $\tilde{\boldsymbol{w}}$, via a supervised learning method (such as weighted SVM).

(c) Based on $\tilde{\boldsymbol{w}}$, we update $\tilde{\boldsymbol{\beta}}_h$ and $\tilde{\boldsymbol{\beta}}_g$ by (17).

(d) The comprehensive schematic diagram of the two-stage procedure, accompanied by detailed notations for all elements to estimating regression parameters $\boldsymbol{\beta}_h$ and $\boldsymbol{\beta}_g$ and threshold parameter $\boldsymbol{w}$ of TBR models:



4.2 | Threshold Boundary Logistic Regression (TBLR) Models

1. For binary dataset $\mathcal{D} = \{(y_n, \boldsymbol{x}_n)\}_{n=1,\ldots,N}$ with $y_n \in \{0, 1\}$, we consider the TBLR models:

$$P_{\boldsymbol{\beta}_h, \boldsymbol{\beta}_g, \boldsymbol{w}}(y_n = 1 | \boldsymbol{x}_n) = \begin{cases} \theta(\boldsymbol{\beta}_h^\top \boldsymbol{x}_n), & \text{if } \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n > 0, \\ \theta(\boldsymbol{\beta}_g^\top \boldsymbol{x}_n), & \text{if } \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n \leq 0 \end{cases} \tag{20}$$

   (a) $\boldsymbol{\beta}_h \in \mathbb{R}^{d+1}$ and $\boldsymbol{\beta}_g \in \mathbb{R}^{d+1}$: regression parameter vectors;

   (b) $\boldsymbol{w}$: threshold parameter vector and

$$\boldsymbol{w} \in \Theta_w = \{\boldsymbol{w} \in \mathbb{R}^{\tilde{d}+1} : \|\boldsymbol{w}\| = 1\}.$$

   (c) $\theta(s)$: the logistic function and recall $\theta(s) = (1 + \exp(-s))^{-1}$;

   (d) $\tilde{\boldsymbol{x}}$: the transformed covariate vector.

2. Loss function of TBLR models:

$$\begin{aligned} \mathrm{E}_{\mathrm{in}}^{(b)}(\boldsymbol{w};\boldsymbol{\beta}_h,\boldsymbol{\beta}_g) = \frac{2}{N}\sum_{n\in\mathcal{H}(\theta_b)}\Big(-y_n\boldsymbol{\beta}_h^{\top}\boldsymbol{x}_n + \log\big(1+\exp(\boldsymbol{\beta}_h^{\top}\boldsymbol{x}_n)\big)\Big) \\ + \frac{2}{N}\sum_{n\in\mathcal{G}(\theta_b)}\Big(-y_n\boldsymbol{\beta}_g^{\top}\boldsymbol{x}_n + \log\big(1+\exp(\boldsymbol{\beta}_g^{\top}\boldsymbol{x}_n)\big)\Big), \end{aligned} \tag{21}$$

   (a) $\mathcal{H}(\boldsymbol{w}) = \{n : \boldsymbol{w}^{\top}\tilde{\boldsymbol{x}}_n > 0,\ n = 1,\ldots,N\}$: index set;

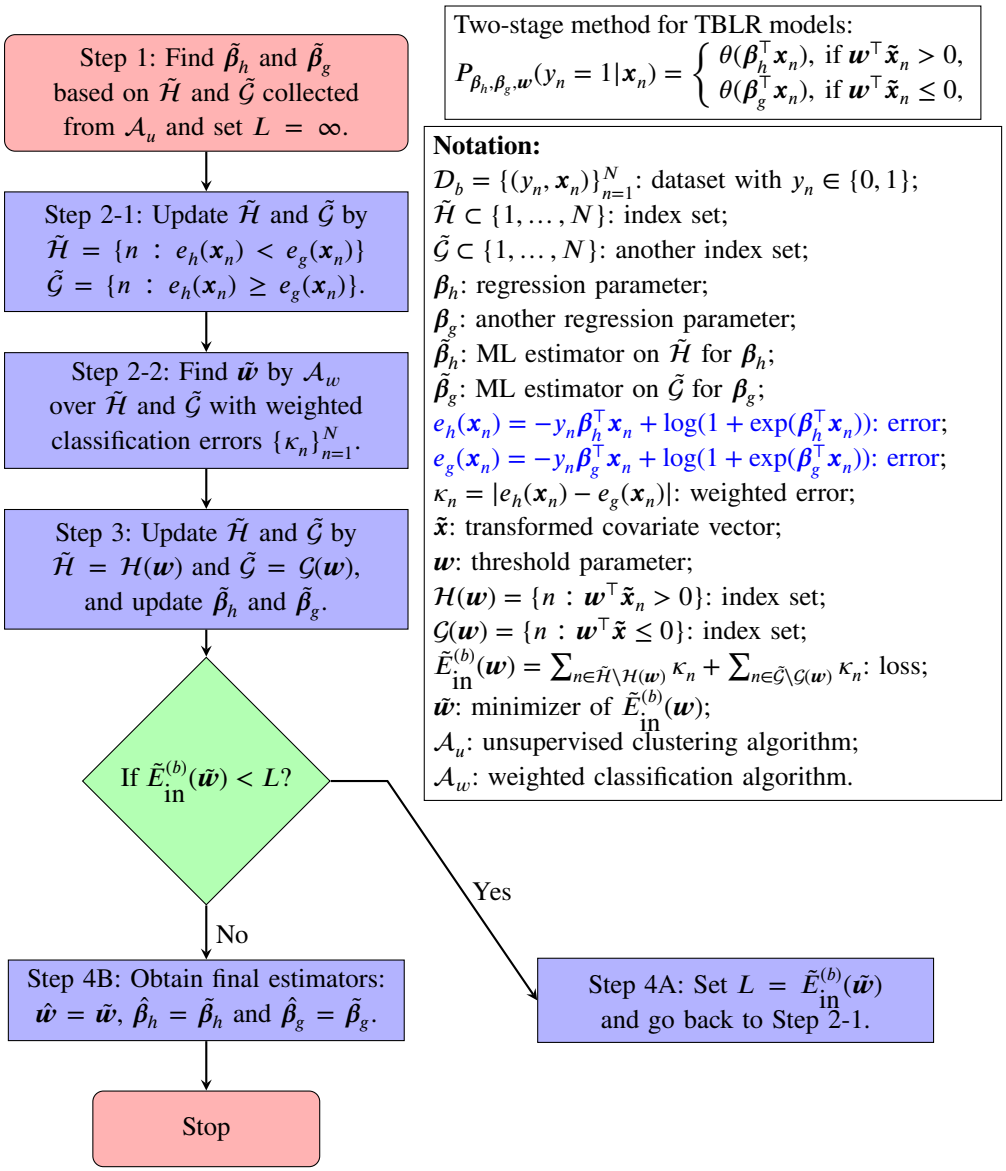
   (b) $\mathcal{G}(\boldsymbol{w}) = \{n : \boldsymbol{w}^{\top}\tilde{\boldsymbol{x}}_n \le 0,\ n = 1,\ldots,N\}$: another index set;

   (c) $\boldsymbol{w}^{\top}\tilde{\boldsymbol{x}}_n = 0$: the classification threshold boundary.

3. Based on $\boldsymbol{w}$, the ML estimators of regression parameter vectors $\boldsymbol{\beta}_h$ and $\boldsymbol{\beta}_g$ can be obtained by GD or SGD methods:

$$\begin{aligned} \tilde{\boldsymbol{\beta}}_h(\boldsymbol{w}) &= \underset{\boldsymbol{\beta}_h\in\mathbb{R}^{d+1}}{\arg\min}\sum_{n\in\mathcal{H}(\boldsymbol{w})}\Big(-y_n\boldsymbol{\beta}_h^{\top}\boldsymbol{x}_n + \log\big(1+\exp(\boldsymbol{\beta}_h^{\top}\boldsymbol{x}_n)\big)\Big), \\ \tilde{\boldsymbol{\beta}}_g(\boldsymbol{w}) &= \underset{\boldsymbol{\beta}_g\in\mathbb{R}^{d+1}}{\arg\min}\sum_{n\in\mathcal{G}(\boldsymbol{w})}\Big(-y_n\boldsymbol{\beta}_g^{\top}\boldsymbol{x}_n + \log\big(1+\exp(\boldsymbol{\beta}_g^{\top}\boldsymbol{x}_n)\big)\Big). \end{aligned} \tag{22}$$

4. A binary version of the two-stage iterative algorithm:



Two-stage method for TBLR models:

$$P_{\boldsymbol{\beta}_h,\boldsymbol{\beta}_g,\boldsymbol{w}}(y_n = 1|\boldsymbol{x}_n) = \begin{cases} \theta(\boldsymbol{\beta}_h^{\top}\boldsymbol{x}_n), & \text{if } \boldsymbol{w}^{\top}\tilde{\boldsymbol{x}}_n > 0, \\ \theta(\boldsymbol{\beta}_g^{\top}\boldsymbol{x}_n), & \text{if } \boldsymbol{w}^{\top}\tilde{\boldsymbol{x}}_n \le 0, \end{cases}$$

Notation:

$\mathcal{D}_b = \{(y_n, \boldsymbol{x}_n)\}_{n=1}^N$: dataset with $y_n \in \{0,1\}$;
$\tilde{\mathcal{H}} \subset \{1,\ldots,N\}$: index set;
$\tilde{\mathcal{G}} \subset \{1,\ldots,N\}$: another index set;
$\boldsymbol{\beta}_h$: regression parameter;
$\boldsymbol{\beta}_g$: another regression parameter;
$\tilde{\boldsymbol{\beta}}_h$: ML estimator on $\tilde{\mathcal{H}}$ for $\boldsymbol{\beta}_h$;
$\tilde{\boldsymbol{\beta}}_g$: ML estimator on $\tilde{\mathcal{G}}$ for $\boldsymbol{\beta}_g$;
$e_h(\boldsymbol{x}_n) = -y_n\boldsymbol{\beta}_h^{\top}\boldsymbol{x}_n + \log(1+\exp(\boldsymbol{\beta}_h^{\top}\boldsymbol{x}_n))$: error;
$e_g(\boldsymbol{x}_n) = -y_n\boldsymbol{\beta}_g^{\top}\boldsymbol{x}_n + \log(1+\exp(\boldsymbol{\beta}_g^{\top}\boldsymbol{x}_n))$: error;
$\kappa_n = |e_h(\boldsymbol{x}_n) - e_g(\boldsymbol{x}_n)|$: weighted error;
$\tilde{\boldsymbol{x}}$: transformed covariate vector;
$\boldsymbol{w}$: threshold parameter;
$\mathcal{H}(\boldsymbol{w}) = \{n : \boldsymbol{w}^{\top}\tilde{\boldsymbol{x}}_n > 0\}$: index set;
$\mathcal{G}(\boldsymbol{w}) = \{n : \boldsymbol{w}^{\top}\tilde{\boldsymbol{x}} \le 0\}$: index set;
$\tilde{E}_{\mathrm{in}}^{(b)}(\boldsymbol{w}) = \sum_{n\in\tilde{\mathcal{H}}\setminus\mathcal{H}(\boldsymbol{w})}\kappa_n + \sum_{n\in\tilde{\mathcal{G}}\setminus\mathcal{G}(\boldsymbol{w})}\kappa_n$: loss;
$\tilde{\boldsymbol{w}}$: minimizer of $\tilde{E}_{\mathrm{in}}^{(b)}(\boldsymbol{w})$;
$\mathcal{A}_u$: unsupervised clustering algorithm;
$\mathcal{A}_w$: weighted classification algorithm.

4.3 | Numerical Studies

1. Simulation (binary case):

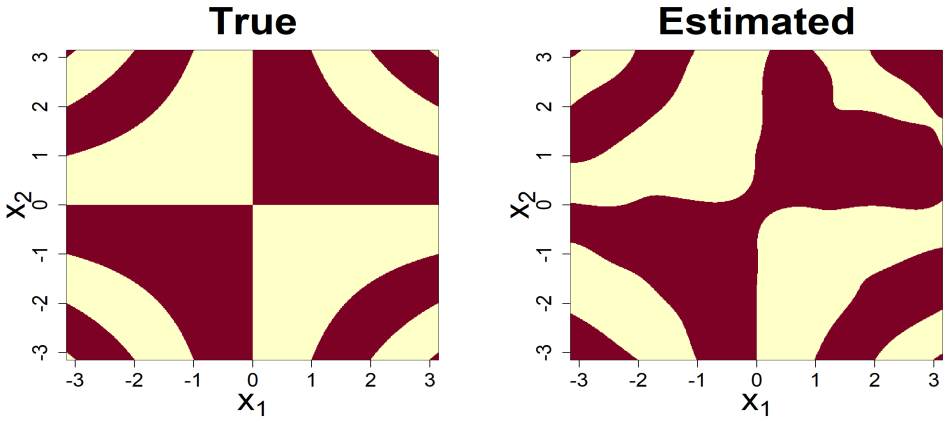
   (a) For the dataset $\mathcal{D}_b = \{(y_n, \boldsymbol{x}_n)\}$ with $\boldsymbol{x}_n = (1, x_{n,1}, x_{n,2})$ (i.e., $d = 2$ in this case);

   (b) $x_{n,1}$ and $x_{n,2}$ generated from Uniform$(-\pi, \pi)$.

   (c) For the TBLR model:

$$P(y_n = 1|\boldsymbol{x}_n) = \begin{cases} \theta\big((0,-1,-2)(1, x_{n,1}, x_{n,2})^\top\big), & \text{if } \sin(x_{n,1} \cdot x_{n,2}) > 0, \\ \theta\big((-3,-2,1)(1, x_{n,1}, x_{n,2})^\top\big), & \text{if } \sin(x_{n,1} \cdot x_{n,2}) \le 0, \end{cases}$$

   (d) Classification images for $N = 20000$, where (I) is the true classification image from the sign of $\sin(x_{n,1} \cdot x_{n,2})$ and (II) is the estimated classification image using weighted SVM as $\mathcal{A}_w$ with $C = 20$ and the radial kernel function:



   (a) (b)

   (e) The R codes for fitting TBLR models are available at https://github.com/antijhow/TBLR-WSVM.

2. Application (continuous case):

   (a) (Day) Bike Sharing Data from https://www.kaggle.com/datasets/contactprad/bike-share-daily-data.

   (b) The dataset $\mathcal{D} = \{(y_n, \boldsymbol{x}_n)\}_{n=1,\ldots,N}$ is related to the two-year historical log corresponding to years 2011 and 2012 from Capital Bikeshare system, Washington D.C., USA.

      i. $N = 731$ days and $d = 6$ covariates;

      ii. y_n: normalized cnt value (count of total rental bikes including both casual and registered);

      iii. $x_{n,1}$: workingday (binary), if day is neither weekend nor holiday is 1, otherwise is 0;

      iv. $x_{n,2}$: weathersit (Ordinal Categorical), 1—4 with higher values corresponding to increasingly adverse weather conditions;

      v. $x_{n,3}$: temp, normalized temperature in Celsius;

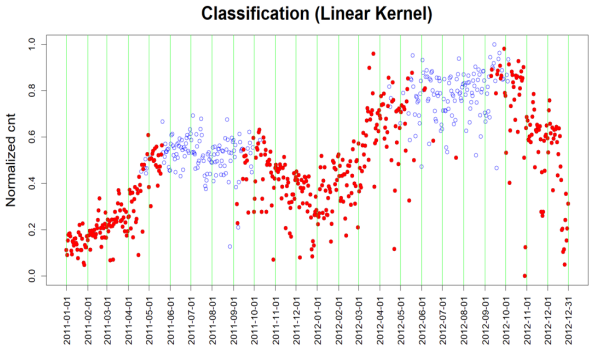
      vi. $x_{n,4}$: atemp, normalized feeling temperature in Celsius;

      vii. $x_{n,5}$: hum, normalized humidity;

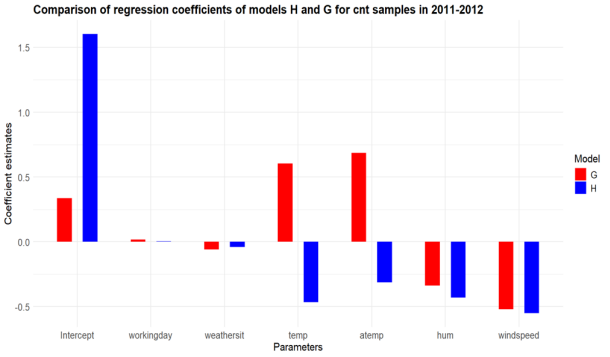
      viii. $x_{n,6}$: windspeed, normalized wind speed.

   (c) Fitting TBR model with linear kernel of WSVM and $C = 50$:

$$y_n = \begin{cases} \boldsymbol{\beta}_h^\top \boldsymbol{x}_n + \epsilon_n, & \text{if } \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n > 0 \\ \boldsymbol{\beta}_g^\top \boldsymbol{x}_n + \epsilon_n, & \text{if } \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n \le 0 \end{cases}$$



(a)



(b)

i. Two distinct groups via TBR model: **Cooler months (Sep–May)** vs. **Warmer months (Jun–Aug)**, revealing a temperature-driven behavioral shift in rental patterns.

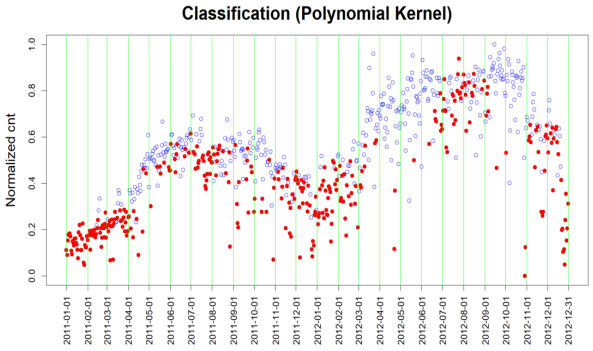
ii. **Opposite Effects of Temperature Variables**:

   A. In cooler months, temperature (temp) and apparent temperature (atemp) have positive effects on rental counts.

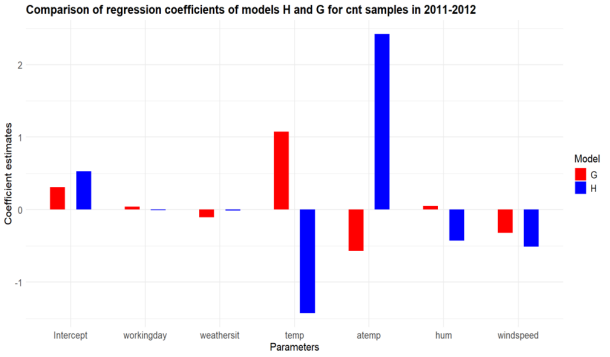
   B. In warmer months, the same variables exhibit negative effects, suggesting heat suppresses demand during hot seasons.

iii. **Negative Effects of Humidity and Wind Speed**: Both humidity and windspeed show negative coefficients, suggesting that more humid or windier conditions deter bike usage, regardless of season.

(d) Fitting TBR model for polynomial kernel of WSVM with $C = 50$ and the polynomial degree $q = 3$:

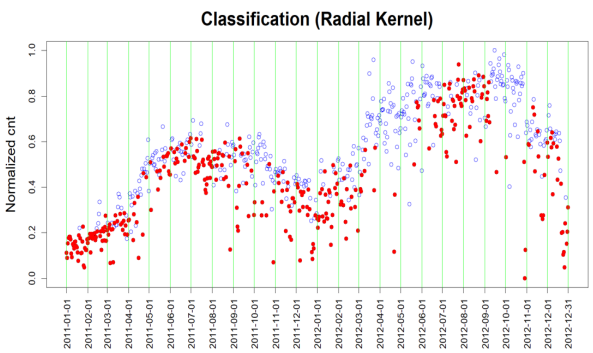


(a)

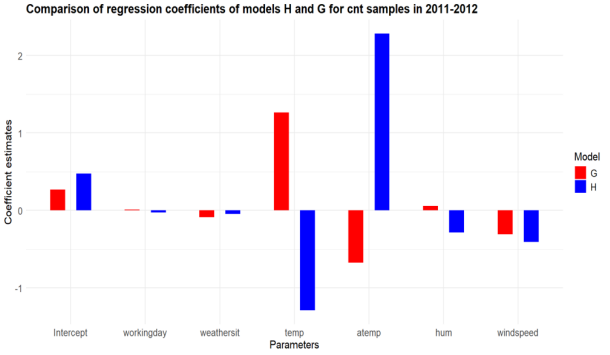


(b)

(e) Fitting TBR model for radial basis kernel of WSVM with $C = 50$:



(a)



(b)

(f) The R codes for fitting TBR models are available at https://github.com/antijhow/TBR-WSVM.