Classification

Yuling Yan

University of Wisconsin–Madison, Fall 2024

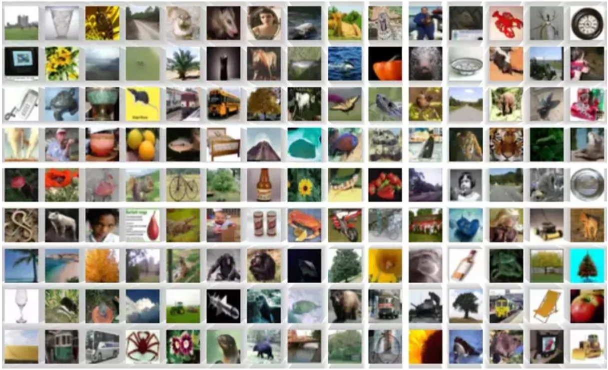
Classification problem

- Classification: assign a label (or category, class) to an observation based on its features
- $\mathcal{X}$: input space (e.g. $\mathbb{R}^d$); $\mathcal{Y}$: output space (e.g. $\{1, 2, \ldots, K\}$)
- $x \in \mathcal{X}$: feature vector, input, data point...
- $y \in \mathcal{Y}$: label, category, class...
- Classifier: a mapping $f : \mathcal{X} \to \mathcal{Y}$
- Goal: construct a classifier f that accurately predicts the label y given the features x

MNIST dataset

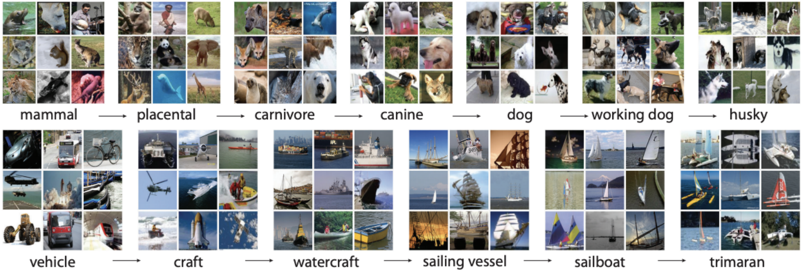
- Input: 28x28 gray scale (1 channel) images, i.e., $\mathcal{X} = \mathbb{R}^{28\times 28}$ or $\mathbb{R}^{784}$
- Output: digits 0 through 9 (i.e., $\mathcal{Y} = \{0, 1, \ldots, 9\}$)

CIFAR datasets



- Input: 32×32 RGB color (3 channels) images, i.e., $\mathcal{X} = \mathbb{R}^{32\times32\times3}$ or $\mathbb{R}^{3072}$
- Output: 10 classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks) or 100 classes

ImageNet dataset



- Input: varies, often high-resolution (often $224 \times 224 \times 3$)
- Output: 1000 different categories

Mathematical set-up

- Modeling assumption: the data (input-output pairs) come from an underlying data distribution ρ over $\mathcal{X} \times \mathcal{Y}$
- Training data: $(x_1, y_1), \ldots, (x_n, y_n) \overset{\text{i.i.d.}}{\sim} \rho$
- Error metric: for any given classifier f, its risk, defined as the average (expected) classification error on a new data is

$$R(f) := \mathbb{P}_{(X,Y)\sim\rho}(f(X) \neq Y)$$

- Supervised learning: build a classifier f based on training data, that makes the average classification error as small as possible

Questions

- Does there exists a "best" classifier?

  — this lecture

- Can we construct this "best" classifier with the information of ρ?

  — this lecture

- What can we do when we only have a finite number of training data?

  — next few weeks

Bayes optimal classifier: binary case

- Consider the binary case: $\mathcal{Y} = \{0, 1\}$
- Define the Bayes classifier: for any $x \in \mathcal{X}$,

$$f^\star(x) := \begin{cases} 1, & \text{if } \mathbb{P}(Y = 1 \mid X = x) \geq \mathbb{P}(Y = 0 \mid X = x), \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 2.1 (Bayes optimal classifier: binary case)

The Bayes classifier $f^\star$ minimizes the misclassification error, i.e.,

$$f^\star \in \operatorname*{arg\,min}_{f:\mathcal{X}\to\mathcal{Y}} \mathbb{P}_{(X,Y)\sim\rho}(f(X) \neq Y).$$

Proof of Theorem 2.1

We need to show that, for any classifier $f : \mathcal{X} \to \mathcal{Y}$,

$$R(f) = \mathbb{P}(f(X) \neq Y) \geq \mathbb{P}(f^\star(X) \neq Y) = R(f^\star)$$

By tower property,

$$\begin{aligned}
\mathbb{P}(f(X) \neq Y) &= \mathbb{E}\left[\mathbb{1}_{f(X)\neq Y}\right] \\
&= \mathbb{E}_X\left[\mathbb{E}\left[\mathbb{1}_{f(X)\neq Y} \mid X\right]\right] && \text{(tower property)} \\
&= \mathbb{E}_X\left[\mathbb{P}\left(f(X) \neq Y \mid X\right)\right] \\
&\geq \mathbb{E}_X\left[\mathbb{P}\left(f^\star(X) \neq Y \mid X\right)\right] && \text{(why?)} \\
&= \mathbb{E}_X\left[\mathbb{E}\left[\mathbb{1}_{f^\star(X)\neq Y} \mid X\right]\right] \\
&= \mathbb{E}\left[\mathbb{1}_{f^\star(X)\neq Y}\right] && \text{(tower property)} \\
&= \mathbb{P}(f^\star(X) \neq Y).
\end{aligned}$$

It suffices to check

$$\mathbb{P}\left(f(X) \neq Y \mid X\right) \geq \mathbb{P}\left(f^\star(X) \neq Y \mid X\right).$$

Proof of Theorem 2.1 (cont.)

Observe that

$$\mathbb{P}(f^\star(X) \neq Y \mid X) = \begin{cases} \mathbb{P}(Y = 0 \mid X) & \text{if } \mathbb{P}(Y = 1 \mid X) \geq \mathbb{P}(Y = 0 \mid X) \\ \mathbb{P}(Y = 1 \mid X) & \text{if } \mathbb{P}(Y = 1 \mid X) \geq \mathbb{P}(Y = 0 \mid X) \end{cases}$$
$$= \min\left\{\mathbb{P}(Y = 1 \mid X), \mathbb{P}(Y = 0 \mid X)\right\}$$

and

$$\mathbb{P}(f(X) \neq Y \mid X) = \begin{cases} \mathbb{P}(Y = 0 \mid X) & \text{if } f(X) = 1 \\ \mathbb{P}(Y = 1 \mid X) & \text{if } f(X) = 0 \end{cases}$$
$$\geq \min\left\{\mathbb{P}(Y = 1 \mid X), \mathbb{P}(Y = 0 \mid X)\right\}.$$

Therefore

$$\mathbb{P}(f^\star(X) \neq Y \mid X) \geq \mathbb{P}(f(X) \neq Y \mid X).$$

A few remarks

Bayes optimal classifier

$$f^\star(x) \coloneqq \begin{cases} 1, & \text{if } \mathbb{P}(Y=1 \mid X=x) \geq \mathbb{P}(Y=0 \mid X=x), \\ 0, & \text{otherwise.} \end{cases}$$

- Depends on the true underlying data distribution ρ
- The optimal classifier might not be unique
- When $\mathcal{X}$ is discrete, it is equivalent to

$$f^\star(x) \coloneqq \begin{cases} 1, & \text{if } \mathbb{P}(X=x, Y=1) \geq \mathbb{P}(X=x, Y=0), \\ 0, & \text{otherwise.} \end{cases}$$

Bayes risk: binary case

- Bayes risk:

$$R^\star := \mathbb{P}_{(X,Y)\sim\rho}(f^\star(X) \neq Y)$$

- The Bayes risk serves as a lower bound for the classification error that any practical classifier can achieve:

$$R^\star = \min_{f:\mathcal{X}\to\mathcal{Y}} \mathbb{P}_{(X,Y)\sim\rho}(f(X) \neq Y).$$

- It represents the inherent uncertainty in the classification problem due to overlapping distributions of the classes.
- Excess risk: $R(f) - R^\star$

Bayes optimal classifier: multiclass setting

- Consider the multiclass case: $\mathcal{Y} = \{1, \ldots, K\}$
- Define the Bayes classifier: for any $x \in \mathcal{X}$,

$$f^\star(x) := \arg\max_{y \in \mathcal{Y}} \mathbb{P}(Y = y \mid X = x)$$

Theorem 2.2 (Bayes optimal classifier: multiclass case)

The Bayes classifier $f^\star$ minimizes the misclassification error, i.e.,

$$f^\star \in \arg\min_{f : \mathcal{X} \to \mathcal{Y}} \mathbb{P}_{(X,Y) \sim \rho}(f(X) \neq Y).$$

Bayes optimal classifier: multiclass setting

- Consider the multiclass case: $\mathcal{Y} = \{1, \dots, K\}$
- Define the Bayes classifier: for any $x \in \mathcal{X}$,

$$f^\star(x) := \arg\max_{y \in \mathcal{Y}} \mathbb{P}(Y = y \mid X = x)$$

Theorem 2.2 (Bayes optimal classifier: multiclass case)

The Bayes classifier $f^\star$ minimizes the misclassification error, i.e.,

$$f^\star \in \arg\min_{f:\mathcal{X}\to\mathcal{Y}} \mathbb{P}_{(X,Y)\sim\rho}(f(X) \neq Y).$$

Proof: similar to Theorem 2.1, it suffices to check for any classifier f

$$\mathbb{P}\left(f(X) \neq Y \mid X\right) \geq \mathbb{P}\left(f^\star(X) \neq Y \mid X\right).$$

More general loss function?

- Consider more general loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- Define the risk for a classifier $f : \mathcal{X} \to \mathcal{Y}$ as

$$R_\ell(f) := \mathbb{E}_{(X,Y)\sim\rho}[\ell(f(X), Y)]$$

- Example: with 0-1 loss $\ell(y, y') = \mathbb{1}\{y \neq y'\}$, we recover the average classification error

$$R(f) = \mathbb{P}_{(X,Y)\sim\rho}(f(X) \neq Y)$$

- Goal: find f that minimizes the risk $R_\ell(f)$ (the Bayes classifier might not be optimal...)

Question: Can you think of settings where other types of loss functions are more appropriate than the 0-1 loss?

Example: traffic signs

- $\mathcal{Y} = \{\text{stop sign}, 50\text{ mph}, 40\text{ mph}\}$.
- Predicting 50 mph when it is actually a stop sign is worse than predicting 40 mph when it is actually 50mph.
- 0-1 loss is not suitable here...

Example: traffic signs

- $\mathcal{Y} = \{\text{stop sign}, 50 \text{ mph}, 40 \text{ mph}\}$.
- Predicting 50 mph when it is actually a stop sign is worse than predicting 40 mph when it is actually 50mph.
- 0-1 loss is not suitable here...

We will discuss classification with general loss later if time permits

Supervised learning

- Go back to 0-1 loss
- In practice, we don't know ρ. It is in general impossible to compute the Bayes classifier $f^\star$
- Goal: build a classifier $f : \mathcal{X} \to \mathcal{Y}$ based on training data $(x_1, y_1), \ldots, (x_n, y_n) \overset{\text{i.i.d.}}{\sim} \rho$
- Hope: achieve small excess risk $R(f) - R^\star$
- High-level framework:
  - Make some modeling assumptions on ρ
  - Design a good classifier f under this setup
  - For example, a good classifier may satisfy

$$R(f) - R^\star \leq h(n)$$

where $h(n)$ is a function of the sample size n describing the rate of convergence, e.g., $h(n) = O(1/n)$.

Linear Methods for Classification

Linear classifiers

- Linear classifiers: decision boundaries are linear hyperplanes
  - Hyperplane $\mathcal{H}_{\boldsymbol{\beta},\beta_0} = \{\boldsymbol{x} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle + \beta_0 = 0\}$
  - Half planes cut by $\mathcal{H}_{\boldsymbol{\beta},\beta_0}$:

$$\mathcal{H}^+_{\boldsymbol{\beta},\beta_0} = \{\boldsymbol{x} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle + \beta_0 \geq 0\},$$
$$\mathcal{H}^-_{\boldsymbol{\beta},\beta_0} = \{\boldsymbol{x} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle + \beta_0 < 0\}.$$

  - Example: in the binary case, the linear classifier has the form

$$f(\boldsymbol{x}) = \mathbb{1}\{\boldsymbol{x} \in \mathcal{H}^+_{\boldsymbol{\beta},\beta_0}\}$$

- Three approaches to learn a linear classifier from the data:
  - Linear discriminant analysis (LDA)
  - Logistic regression
  - Perceptrons and Support vector machines (SVMs)

Linear discriminant analysis (LDA)

- Model set-up: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{1, \ldots, K\}$. For $k = 1, \ldots, K$,

$$\mathbb{P}(Y = k) = \omega_k, \qquad X \mid Y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

  where $\omega_k \geq 0$, $\sum_{k=1}^{K} \omega_k = 1$, $\boldsymbol{\mu}_k \in \mathbb{R}^d$, $\boldsymbol{\Sigma} \in \mathbb{S}_+^d$

- The Bayes classifier under this setup: for any $\boldsymbol{x}$, compute

$$\delta_k(\boldsymbol{x}) := \underbrace{\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log \omega_k}_{\propto \log \mathbb{P}(Y=k \mid X=\boldsymbol{x}) + \text{constant}}.$$

  Let $f^\star(\boldsymbol{x}) = \arg\max_{1 \leq k \leq K} \delta_k(\boldsymbol{x})$.

- Issue: model parameters are unknown...

Plug-in approach

- Plug-in approach: replace the unknown parameters with reliable estimates
- Suppose we have i.i.d. data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \overset{\text{i.i.d.}}{\sim} \rho$
- For each $1 \leq k \leq K$, let $n_k = \sum_{i=1}^{n} \mathbb{1}\{y_i = k\}$ and

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i: y_i = k} \boldsymbol{x}_i, \qquad \widehat{\omega}_k = \frac{n_k}{n}$$

- Estimate the covariance matrix

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{N - K} \sum_{k=1}^{K} \sum_{i: y_i = k} \big(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k\big)\big(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k\big)^\top$$

- Replace $\boldsymbol{\mu}_k$, ω_k, $\boldsymbol{\Sigma}$ with $\widehat{\boldsymbol{\mu}}_k$, $\widehat{\omega}_k$, $\widehat{\boldsymbol{\Sigma}}$

$$\widehat{\delta}_k(\boldsymbol{x}) := \underbrace{\boldsymbol{x}^\top \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_k - \frac{1}{2} \widehat{\boldsymbol{\mu}}_k^\top \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_k + \log \widehat{\omega}_k}_{\text{linear in } \boldsymbol{x}}.$$

Generalization

- Consider a more general set-up: for $k = 1, \dots, K$, assume

$$\mathbb{P}(Y = k) = \omega_k, \qquad X \mid Y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  where $\omega_k \geq 0$, $\sum_{k=1}^K \omega_k = 1$, $\mu_k \in \mathbb{R}^d$, $\boldsymbol{\Sigma}_k \in \mathbb{S}^d_+$
- This setup will lead to the so-called quadratic discriminant analysis (QDA)
- Homework: derive QDA
  - What is the Bayes classifier under this setup?
  - How to derive a practical (data-driven) classifier?
  - Is this still a linear classifier?

Logistic regression

- Model set-up: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{0, 1, \ldots, K\}$. Let

$$\mathbb{P}(Y = k \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{\beta}_k^\top \boldsymbol{x} + \beta_{0,k})}{1 + \sum_{k'=1}^{K} \exp(\boldsymbol{\beta}_{k'}^\top \boldsymbol{x} + \beta_{0,k'})}, \quad (1 \le k \le K),$$

$$\mathbb{P}(Y = 0 \mid \boldsymbol{x}) = \frac{1}{1 + \sum_{k'=1}^{K} \exp(\boldsymbol{\beta}_{k'}^\top \boldsymbol{x} + \beta_{0,k})},$$

where the parameters $\boldsymbol{\beta}_k \in \mathbb{R}^d$, $\beta_{0,k} \in \mathbb{R}$ for $k = 1, \ldots, K$

Logistic regression

- Model set-up: $\mathcal{X} = \mathbb{R}^d \times \{1\}$, $\mathcal{Y} = \{0, 1, \ldots, K\}$. Let

$$\mathbb{P}(Y = k \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{\beta}_k^\top \boldsymbol{x})}{1 + \sum_{k'=1}^K \exp(\boldsymbol{\beta}_{k'}^\top \boldsymbol{x})}, \qquad (k = 1, \ldots, K),$$

$$\mathbb{P}(Y = 0 \mid \boldsymbol{x}) = \frac{1}{1 + \sum_{k'=1}^K \exp(\boldsymbol{\beta}_{k'}^\top \boldsymbol{x})},$$

  where the parameters $\boldsymbol{\beta}_k \in \mathbb{R}^{d+1}$ for $k = 1, \ldots, K$

- Bayes classifier:

$$f(\boldsymbol{x}) = \begin{cases} \text{argmax}_{1 \le k \le K} \boldsymbol{\beta}_k^\top \boldsymbol{x}, & \text{if } \max_{1 \le k \le K} \boldsymbol{\beta}_k^\top \boldsymbol{x} > 0, \\ 0, & \text{otherwise.} \end{cases}$$

- Estimate $\boldsymbol{\beta}_k$'s: maximum likelihood estimation (MLE)

Maximum likelihood estimation

- Suppose we have i.i.d. data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
- The negative log-likelihood function

$$\ell(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{k=1}^{K}\sum_{i:y_i=k}\boldsymbol{x}_i^\top\boldsymbol{\beta}_k + \frac{1}{n}\sum_{i=1}^{n}\log\left[1 + \sum_{k'=1}^{K}\exp(\boldsymbol{x}_i^\top\boldsymbol{\beta}_{k'})\right]$$

- Maximum likelihood estimation (MLE)

$$\widehat{\boldsymbol{\beta}} := \arg\min_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta})$$

- Convex optimization: solve by e.g., gradient descent

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t - \eta\nabla\ell(\boldsymbol{\beta}^t) \qquad (t = 0, 1, \ldots)$$

A brief introduction to gradient descent

Gradient descent (GD) for solving $\min_{\boldsymbol{\beta}\in\mathbb{R}^d} L(\boldsymbol{\beta})$:

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t - \eta \nabla L(\boldsymbol{\beta}^t) \qquad (t = 0, 1, \ldots)$$

When η is properly small, GD satisfy the following properties:

- For smooth function L, GD is a descent algorithm: $L(\boldsymbol{\beta}^{t+1}) \le L(\boldsymbol{\beta}^t)$
- For convex + smooth function L, GD satisfies

  $$L(\boldsymbol{\beta}^t) - L(\boldsymbol{\beta}^\star) \le O\left(\frac{\|\boldsymbol{\beta}^0 - \boldsymbol{\beta}^\star\|_2^2}{t}\right) \qquad (t = 0, 1, \ldots)$$

  for any minimizer $\boldsymbol{\beta}^\star$
- For strongly convex + smooth function L, GD satisfies

  $$\|\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^\star\|_2 \le (1-\kappa)^t \|\boldsymbol{\beta}^0 - \boldsymbol{\beta}^\star\|_2 \qquad (t = 0, 1, \ldots)$$

  for some $\kappa \in (0, 1)$, where $\boldsymbol{\beta}^\star$ is the unique minimizer

Stochastic gradient descent

Consider the following empirical risk minimization problem

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} L(\boldsymbol{\beta}) := \frac{1}{n} \sum_{i=1}^{n} g(\boldsymbol{\beta}; \boldsymbol{x}_i),$$

where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are training data points.

- **Stochastic gradient descent:** for $t = 0, 1, \ldots,$

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t - \eta \nabla g(\boldsymbol{\beta}^t; \boldsymbol{x}_{i_t}) \quad \text{where} \quad \boldsymbol{x}_{i_t} \overset{\text{ind.}}{\sim} \text{Unif}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$$

- **Gradient descent:** for $t = 0, 1, \ldots,$

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t - \eta \nabla L(\boldsymbol{\beta}^t) = \boldsymbol{\beta}^t - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla g(\boldsymbol{\beta}; \boldsymbol{x}_i)$$

- **Advantage of SGD:** much faster updates, especially for large datasets, but still enjoys nice properties (sometimes even better than GD!)

Gradient descent methods

Example: GD / SGD for logistic regresion

Take-away: (stochastic) gradient descent is the default method for solving unconstrained optimization problem

— simple and effective!

Recommended reading materials: Lecture 1 and 10 of the course

Large-Scale Optimization for Data Science

by Prof. Yuxin Chen (UPenn); Lecture on GD and SGD

Perceptrons and SVMs

Linearly separable data

- Consider binary classification: $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, -1\}$
- Training data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
- **Linearly separable data:** $\exists$ a separating hyperplane $\mathcal{H}_{\boldsymbol{\beta},\beta_0}$ s.t.

$$y_i \cdot (\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) > 0 \qquad (i = 1, \ldots, n)$$

Linearly separable data

- Consider binary classification: $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, -1\}$
- Training data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
- **Linearly separable data:** $\exists$ a separating hyperplane $\mathcal{H}_{\boldsymbol{\beta},\beta_0}$ s.t.

$$y_i \cdot (\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) > 0 \qquad (i = 1, \ldots, n)$$

- by merging β_0 into $\boldsymbol{\beta}$ and adding 1 to $\boldsymbol{x}_i$'s, this assumption becomes: $\exists\, \boldsymbol{\beta}_{\mathsf{sep}} \in \mathbb{R}^{d+1}$

$$y_i \cdot \boldsymbol{x}_i^\top \boldsymbol{\beta}_{\mathsf{sep}} > 0 \qquad (i = 1, \ldots, n)$$

Linearly separable data

- Consider binary classification: $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, -1\}$
- Training data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
- **Linearly separable data:** $\exists$ a separating hyperplane $\mathcal{H}_{\boldsymbol{\beta}, \beta_0}$ s.t.

$$y_i \cdot (\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) > 0 \qquad (i = 1, \ldots, n)$$

- by merging β_0 into $\boldsymbol{\beta}$ and adding 1 to $\boldsymbol{x}_i$'s, this assumption becomes: $\exists\, \boldsymbol{\beta}_{\mathsf{sep}} \in \mathbb{R}^{d+1}$

$$y_i \cdot \boldsymbol{x}_i^\top \boldsymbol{\beta}_{\mathsf{sep}} > 0 \qquad (i = 1, \ldots, n)$$

- **Goal:** search a separating hyperplane indexed by $\widehat{\boldsymbol{\beta}}$

$$y_i \cdot \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}} > 0 \qquad (i = 1, \ldots, n)$$

(note that $\boldsymbol{\beta}_{\mathsf{sep}}$ is not known a priori)

Perceptron Learning Algorithm

- For every $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$, define the set $\mathcal{M}_{\boldsymbol{\beta}} := \underbrace{\{i : y_i \cdot \boldsymbol{x}_i^\top \boldsymbol{\beta} \leq 0\}}_{\text{misclassified points}}$
- Target: minimize the perceptron loss

$$\sigma(\boldsymbol{\beta}) := -\sum_{i \in \mathcal{M}_{\boldsymbol{\beta}}} y_i \cdot \boldsymbol{x}_i^\top \boldsymbol{\beta} \propto \sum_{i \in \mathcal{M}_{\boldsymbol{\beta}}} \text{dist}(\boldsymbol{x}_i, \mathcal{H}_{\boldsymbol{\beta}})$$

  where $\mathcal{H}_{\boldsymbol{\beta}} = \{\boldsymbol{x} : \boldsymbol{x}^\top \boldsymbol{\beta} = 0\}$

- Algorithm: initialize with $\boldsymbol{\beta}^0 \in \mathbb{R}^{d+1}$, for $t = 0, 1, \ldots,$ update

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t + \eta y_i \boldsymbol{x}_i, \quad \text{for a random } i \in \mathcal{M}_{\boldsymbol{\beta}^t}$$

  where $\eta > 0$ is the step size; in fact, we can take $\eta = 1$ here...

Perceptron Learning Algorithm

- For every $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$, define the set $\mathcal{M}_{\boldsymbol{\beta}} := \underbrace{\{i : y_i \cdot \boldsymbol{x}_i^\top \boldsymbol{\beta} \le 0\}}_{\text{misclassified points}}$
- Target: minimize the perceptron loss

$$\sigma(\boldsymbol{\beta}) := -\sum_{i \in \mathcal{M}_{\boldsymbol{\beta}}} y_i \cdot \boldsymbol{x}_i^\top \boldsymbol{\beta} \propto \sum_{i \in \mathcal{M}_{\boldsymbol{\beta}}} \text{dist}(\boldsymbol{x}_i, \mathcal{H}_{\boldsymbol{\beta}})$$

  where $\mathcal{H}_{\boldsymbol{\beta}} = \{\boldsymbol{x} : \boldsymbol{x}^\top \boldsymbol{\beta} = 0\}$
- Algorithm: initialize with $\boldsymbol{\beta}^0 \in \mathbb{R}^{d+1}$, for $t = 0, 1, \ldots,$ update

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t + y_i \boldsymbol{x}_i, \quad \text{for a random } i \in \mathcal{M}_{\boldsymbol{\beta}^t}$$

- Interpretation: SGD with step size 1 (kind of...)

Convergence theory

Theorem 2.3

When the data is linearly separable, the perceptron learning algorithm converges to a separating hyperplane in a finite number of steps.

Convergence theory

Theorem 2.3

When the data is linearly separable, the perceptron learning algorithm converges to a separating hyperplane in a finite number of steps.

Limitations:

- solutions not unique: might converge to an unstable hyperplane

Convergence theory

Theorem 2.3

When the data is linearly separable, the perceptron learning algorithm converges to a separating hyperplane in a finite number of steps.

Limitations:

- solutions not unique: might converge to an unstable hyperplane

  — resort to "optimal separating hyperplane"

Convergence theory

Theorem 2.3

When the data is linearly separable, the perceptron learning algorithm converges to a separating hyperplane in a finite number of steps.

Limitations:

- solutions not unique: might converge to an unstable hyperplane
  — resort to "optimal separating hyperplane"
- only works linearly separable data. If the classes cannot be separated by a hyperplane, the algorithm will not converge

Convergence theory

Theorem 2.3

When the data is linearly separable, the perceptron learning algorithm converges to a separating hyperplane in a finite number of steps.

Limitations:

- solutions not unique: might converge to an unstable hyperplane
  — resort to "optimal separating hyperplane"
- only works linearly separable data. If the classes cannot be separated by a hyperplane, the algorithm will not converge
- the "finite" number of steps can be very large

Optimal separating hyperplane

From now on, we "unmerge" β_0 from $\boldsymbol{\beta}$, as they play different roles. Consider the optimization problem

$$\max_{\|\boldsymbol{\beta}\|_2=1,\beta_0,M} \quad M \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq M \quad (i = 1, \ldots, n)$$

Optimal separating hyperplane

From now on, we "unmerge" β_0 from $\boldsymbol{\beta}$, as they play different roles. Consider the optimization problem

$$\max_{\|\boldsymbol{\beta}\|_2=1,\beta_0,M} \quad M \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq M \quad (i = 1, \ldots, n)$$

Implications:

- the distance between $\boldsymbol{x}$ and the hyperplane $\mathcal{H}_{\boldsymbol{\beta},\beta_0}$ is

$$\text{dist}(\boldsymbol{x}, \mathcal{H}_{\boldsymbol{\beta},\beta_0}) = \frac{|\boldsymbol{\beta}^\top \boldsymbol{x} + \beta_0|}{\|\boldsymbol{\beta}\|_2} \overset{\text{if } \|\beta\|_2=1}{=\!=\!=} |\boldsymbol{\beta}^\top \boldsymbol{x} + \beta_0|$$

Optimal separating hyperplane

From now on, we "unmerge" β_0 from $\boldsymbol{\beta}$, as they play different roles.
Consider the optimization problem

$$\max_{\|\boldsymbol{\beta}\|_2=1, \beta_0, M} \quad M \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq M \quad (i = 1, \ldots, n)$$

Implications:

- the distance between $\boldsymbol{x}$ and the hyperplane $\mathcal{H}_{\boldsymbol{\beta},\beta_0}$ is $|\boldsymbol{\beta}^\top \boldsymbol{x} + \beta_0|$
- offers a unique solution that maximizes the *margin* M
- **Margin:** the distance between $\mathcal{H}_{\boldsymbol{\beta},\beta_0}$ and the $\underbrace{\text{closest data points}}_{\text{support vectors}}$ from each class

Optimal separating hyperplane

From now on, we "unmerge" β_0 from $\boldsymbol{\beta}$, as they play different roles. Consider the optimization problem

$$\max_{\|\boldsymbol{\beta}\|_2=1,\beta_0,M} \quad M \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq M \quad (i = 1, \ldots, n)$$

Implications:

- the distance between $\boldsymbol{x}$ and the hyperplane $\mathcal{H}_{\boldsymbol{\beta},\beta_0}$ is $|\boldsymbol{\beta}^\top \boldsymbol{x} + \beta_0|$
- offers a unique solution that maximizes the *margin* M
- **Margin:** the distance between $\mathcal{H}_{\boldsymbol{\beta},\beta_0}$ and the $\underbrace{\text{closest data points}}_{\text{support vectors}}$ from each class
- **Intuition:** a large margin on the training data will lead to good separation on the test data.

Reformulation as convex optimization

- **Original problem:**

$$\max_{\|\boldsymbol{\beta}\|_2=1,\beta_0,M} \quad M \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq M \quad (i = 1, \ldots, n)$$

- **Issue:** this is not a convex optimization problem...

Reformulation as convex optimization

- **Original problem:**

$$\max_{\|\boldsymbol{\beta}\|_2=1,\beta_0,M} \quad M \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq M \quad (i = 1, \ldots, n)$$

- **Issue:** this is not a convex optimization problem...

- **Reformulation:**

$$\min_{\boldsymbol{\beta},\beta_0} \quad \|\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 \quad (i = 1, \ldots, n)$$

  this is a convex optimization problem

Reformulation as convex optimization

- **Original problem:**

$$\max_{\|\boldsymbol{\beta}\|_2=1,\beta_0,M} \quad M \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq M \quad (i = 1, \ldots, n)$$

- **Issue:** this is not a convex optimization problem...
- **Reformulation:**

$$\min_{\boldsymbol{\beta},\beta_0} \quad \|\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 \quad (i = 1, \ldots, n)$$

  this is a convex optimization problem

- This is known as the support vector machine (SVM)

SVMs for separable data

$$\min_{\boldsymbol{\beta},\beta_0} \quad \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 \quad (i = 1, \ldots, n)$$

- SVM is a powerful method for binary classification
- finds a linear classifier with decision boundary $\{\boldsymbol{x} : \boldsymbol{x}^\top \widehat{\boldsymbol{\beta}} + \widehat{\beta}_0 = 0\}$ to separate two classes with the maximum margin
- This is only feasible for *linearly separated data*

  — *can be generalized to accommodate non-separable data*
- What can we say about SVM?

  — *resort to duality theory!*

Convex optimization and duality theory

Primal problem and Lagrangian function

- Consider a convex optimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) \quad \text{s.t.} \quad g_i(\boldsymbol{x}) \le 0 \quad (i = 1, \ldots, m).$$

  where $f(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ are convex functions

- This is called the **primal problem**
- To handle the constraints, we introduce **Lagrange multipliers** λ_i
- The Lagrangian function is:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x})$$

- What is the benefit of introducing the Lagrangian function?

The Dual Problem

Key observation:

$$\underbrace{\min_{\boldsymbol{x}:g(\boldsymbol{x})\leq 0} f(\boldsymbol{x})}_{\text{primal problem}} \overset{\text{(i)}}{=} \min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}\geq 0} L(\boldsymbol{x},\boldsymbol{\lambda}) \overset{\text{(ii)}}{\geq} \max_{\boldsymbol{\lambda}\geq 0} \underbrace{\min_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda})}_{=:d(\boldsymbol{\lambda})} = \underbrace{\max_{\boldsymbol{\lambda}\geq 0} d(\boldsymbol{\lambda})}_{\text{dual problem}}$$

- relation (i) and (ii) always holds (why?)
- relation (ii) is often an equality (strong duality theory)
- The dual function $d(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda})$
- The **dual problem** is to maximize the dual function $d(\boldsymbol{\lambda})$:

$$\max_{\boldsymbol{\lambda}\geq 0} d(\boldsymbol{\lambda})$$

Strong and Weak Duality

Weak Duality: For any $\boldsymbol{x}$ feasible in the primal and any $\boldsymbol{\lambda} \geq 0$, we have:

$$d(\boldsymbol{\lambda}) \leq f(\boldsymbol{x})$$

Strong Duality: If the problem satisfies certain conditions (e.g., Slater's condition), then:

$$\min_{\boldsymbol{x}: g(\boldsymbol{x}) \leq 0} f(\boldsymbol{x}) = \max_{\boldsymbol{\lambda} \geq 0} d(\boldsymbol{\lambda})$$

- Slater's condition: the feasible region has an interior point, i.e.,

$$\exists\, \boldsymbol{x}_0 \in \mathbb{R}^d \quad \text{s.t.} \quad g_i(\boldsymbol{x}_0) < 0 \quad (i = 1, \ldots, m).$$

- In convex optimization, strong duality often holds, meaning the primal and dual problems have the same optimal value.

KKT Conditions

The **Karush-Kuhn-Tucker (KKT)** conditions: if strong duality holds, and $(\boldsymbol{x}, \boldsymbol{\lambda})$ is the optimal solution pair for the primal/dual problem

$$\underbrace{\min_{\boldsymbol{x}: g(\boldsymbol{x}) \leq 0} f(\boldsymbol{x})}_{\text{primal problem}} = \underbrace{\max_{\boldsymbol{\lambda} \geq 0} d(\boldsymbol{\lambda})}_{\text{dual problem}},$$

then

- **Primal feasibility:** $g_i(\boldsymbol{x}) \leq 0$
- **Dual feasibility:** $\lambda_i \geq 0$
- **Complementary slackness:** $\lambda_i g_i(\boldsymbol{x}) = 0$
- **Stationarity:** $\nabla f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\boldsymbol{x}) = 0$

— This is a necessary condition!

Back to SVMs

$$\min_{\boldsymbol{\beta},\beta_0} \quad \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 \quad (i = 1, \ldots, n)$$

- The **dual problem** for SVM is (why?):

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \quad \text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0,\ \alpha_i \geq 0$$

- It is straightforward to check that Slater's condition holds

  — *primal and dual problems are equivalent!*

- The dual problem is a quadratic programming problem, which is easier to compute with standard software (e.g. CVX)

Checking KKT conditions

$$\text{(P)} \quad \min_{\boldsymbol{\beta},\beta_0} \quad \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 \quad (i = 1, \ldots, n)$$

$$\text{(D)} \quad \max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \quad \text{s.t.} \quad \sum_{i=1}^n \alpha_i y_i = 0,\ \alpha_i \geq 0$$

The Karush-Kuhn-Tucker (KKT) conditions for optimality:

- **Primal feasibility:** $y_i(\boldsymbol{\beta}^\top \boldsymbol{x}_i + \beta_0) \geq 1$
- **Dual feasibility:** $\alpha_i \geq 0$
- **Complementary slackness:** $\alpha_i[y_i(\boldsymbol{\beta}^\top \boldsymbol{x}_i + \beta_0) - 1] = 0$
- **Stationarity:** $\boldsymbol{\beta} = \sum_{i=1}^n \alpha_i y_i \boldsymbol{x}_i$

Implications

For any optimal solution pair $(\boldsymbol{\beta}^\star, \beta_0^\star, \alpha^\star)$:

- **Support vectors:** data points $\boldsymbol{x}_i$ with $\alpha_i > 0$

$$y_i(\boldsymbol{\beta}^{\star\top}\boldsymbol{x}_i + \beta_0^\star) > 1 \quad \Longrightarrow \quad \alpha_i = 0$$
$$\alpha_i > 0 \quad \Longrightarrow \quad y_i(\boldsymbol{\beta}^{\star\top}\boldsymbol{x}_i + \beta_0^\star) = 1$$

- **Recovering the primal solution:** after solving the dual problem (i.e., finding $\alpha_i^\star$), we can recover the primal solution $(\boldsymbol{\beta}^\star, \beta_0^\star)$ by

$$\boldsymbol{\beta}^\star = \sum_{i=1}^n \alpha_i^\star y_i \boldsymbol{x}_i$$

  and $\beta_0^\star = y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i$ for any support vector $\boldsymbol{x}_i$

— *$\boldsymbol{\beta}^\star$ is a linear combination of the support vectors*

Accommodating non-separable data

SVM for linearly separable data:

$$\min_{\boldsymbol{\beta},\beta_0} \quad \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad y_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 \quad (i = 1, \ldots, n)$$

- For non-separable data, we introduce slack variables $\xi_i \geq 0$ to allow violations of the margin:

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0,\xi} \; \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t.} \quad y_i(\boldsymbol{\beta}^\top \boldsymbol{x}_i + \beta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (i = 1, \ldots, n) \end{aligned}$$

- $C > 0$ is the “cost” parameter
- the separable case corresponds to $C = \infty$

Dual problem: non-separable data

- **Primal problem:**

$$\min_{\boldsymbol{\beta}, \beta_0, \xi} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad y_i(\boldsymbol{\beta}^\top \boldsymbol{x}_i + \beta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (i = 1, \ldots, n)$$

- **Dual problem:**

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \quad (i = 1, \ldots, n)$$

- Homework: derive the dual problem from the primal problem

Dual problem: non-separable data

- **Primal problem:**

$$\min_{\boldsymbol{\beta},\beta_0,\xi} \; \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(\boldsymbol{\beta}^\top \boldsymbol{x}_i + \beta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (i = 1, \ldots, n)$$

- **Dual problem:**

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \quad (i = 1, \ldots, n)$$

- Homework: derive the dual problem from the primal problem

Kernel density classifier and naive Bayes classifier

Recap: Bayes optimal classifier

Bayes optimal classifier: for any $x \in \mathcal{X}$, output

$$f^\star(x) := \arg\max_{y \in \mathcal{Y}} \mathbb{P}(Y = y \mid X = x)$$

- Issue: depends on unknown data distribution ρ

Recap: Bayes optimal classifier

Bayes optimal classifier: for any $x \in \mathcal{X}$, output

$$f^\star(x) := \arg\max_{y \in \mathcal{Y}} \mathbb{P}(Y = y \mid X = x)$$

- Issue: depends on unknown data distribution ρ
- Bayes formula:

$$\mathbb{P}(Y = y \mid X = x) = \frac{\mathbb{P}(X = x \mid Y = y)\,\mathbb{P}(Y = y)}{\sum_{y' \in \mathcal{Y}} \mathbb{P}(X = x \mid Y = y')\,\mathbb{P}(Y = y')}$$

— *Is it possible to estimate these quantities?*

Recap: Bayes optimal classifier

Bayes optimal classifier: for any $x \in \mathcal{X}$, output

$$f^\star(x) := \arg\max_{y \in \mathcal{Y}} \mathbb{P}(Y = y \mid X = x)$$

- Issue: depends on unknown data distribution ρ
- Bayes formula:

$$\mathbb{P}(Y = y \mid X = x) = \frac{\widehat{\mathbb{P}}(X = x \mid Y = y)\,\widehat{\mathbb{P}}(Y = y)}{\sum_{y' \in \mathcal{Y}} \widehat{\mathbb{P}}(X = x \mid Y = y')\,\widehat{\mathbb{P}}(Y = y')}$$

— *Is it possible to estimate these quantities?*

- Plug-in method:
  - marginal probabilities $\mathbb{P}(Y = y)$ are easy to estimate (use frequency)
  - key difficulty: estimate conditional densities $\mathbb{P}(X = x \mid Y = y)$

Detour: density estimation

Setup: density estimation

- Target: an unknown density function f
- What we have: i.i.d. data $X_1, \ldots, X_n \sim f$
- Goal: construct a good density estimation $\widehat{f}(\cdot)$

Setup: density estimation

- Target: an unknown density function f
- What we have: i.i.d. data $X_1, \ldots, X_n \sim f$
- Goal: construct a good density estimation $\widehat{f}(\cdot)$
- Criteria: mean integrated squared error (MISE)

$$\text{MISE}(\widehat{f}) = \mathbb{E}\left[\int \left(\widehat{f}(x) - f(x)\right)^2 dx \right]$$

Setup: density estimation

- Target: an unknown density function f
- What we have: i.i.d. data $X_1, \dots, X_n \sim f$
- Goal: construct a good density estimation $\widehat{f}(\cdot)$
- Criteria: mean integrated squared error (MISE)

$$\mathsf{MISE}(\widehat{f}) = \mathbb{E}\Big[\int \big(\widehat{f}(x) - f(x)\big)^2 dx\Big]$$

- Density estimation: find $\widehat{f}$ with as small MISE as possible
  - Histogram method
  - Kernel density estimation

Bias-variance tradeoff

Mean integrated squared error (MISE):

$$\mathrm{MISE}(\widehat{f}) = \mathbb{E}\Big[\int \big(\widehat{f}(x) - f(x)\big)^2 dx\Big]$$

- **Bias:** Measures how far the estimated density is from the true density on average.

$$b(x) := \mathbb{E}[\widehat{f}(x)] - f(x)$$

- **Variance:** Measures how much $\hat{p}(x)$ fluctuates around its mean:

$$v(x) := \mathsf{var}(\widehat{f}(x)) = \mathbb{E}[(\widehat{f}(x) - \mathbb{E}[\widehat{f}(x)])^2]$$

Theorem 2.4

$$\mathrm{MISE}(\widehat{f}) = \int b^2(x)\mathrm{d}x + \int v(x)\mathrm{d}x$$

Histograms

- Consider 1D setting, and assume that $f(\cdot)$ is supported on $[0,1]$

  — *we can always rescale the data to* $[0,1]$

- Histogram method: estimate the density by partitioning the interval and counting the frequency of data points in each partition

- The data is divided into m bins of equal width $h = 1/m$ (bandwidth)

$$B_1 = \left[0, \frac{1}{m}\right), \quad B_2 = \left[\frac{1}{m}, \frac{2}{m}\right), \quad \dots \quad , B_m = \left[\frac{m-1}{m}, 1\right]$$

- Each bin is assigned a probability proportional to the number of observations falling into that bin:

$$\widehat{f}(x) := \begin{cases} \widehat{p}_1/h, & x \in B_1, \\ \vdots & \vdots \\ \widehat{p}_m/h, & x \in B_m, \end{cases} \quad \text{where} \quad \widehat{p}_j = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i \in B_j\}.$$

Optimal bandwidth

Theorem 2.5 (informal)

Under some regularity conditions, we have

$$\mathrm{MISE}(\widehat{f}) \approx \frac{h^2}{12} \int f'(u)^2 \mathrm{d}u + \frac{1}{nh}$$

- *The optimal bandwidth choice is*

$$h^\star = \frac{1}{n^{1/3}} \left(\frac{6}{\int f'(u)^2 \mathrm{d}u} \right)^{1/3}$$

- *With this choice of $h^\star$, we have*

$$\mathrm{MISE}(\widehat{f}) \approx \frac{C}{n^{2/3}} \quad \textit{where} \quad C = \left(\frac{3}{4}\right)^{2/3} \left(\int f'(u)^2 \mathrm{d}u \right)^{1/3}.$$

Cross-validation

- Issue: the optimal bandwidth $h^\star$ depends on the unknown density f
- Idea: estimate the risk under each bandwidth selection h

$$L(h) \coloneqq \int \left(\widehat{f}(x) - f(x)\right)^2 \mathrm{d}x = \underbrace{\int \widehat{f}^2(x)\mathrm{d}x - 2\int \widehat{f}(x) f(x)\mathrm{d}x}_{=:J(h)} + \int f^2(x)\mathrm{d}x$$

- Cross-validation estimate of the risk:

$$\widehat{J}(h) \coloneqq \int \widehat{f}^2(x)\mathrm{d}x - \frac{2}{n}\sum_{i=1}^{n} \widehat{f}_{(-i)}(X_i)$$

- It can be shown that $\widehat{J}(h) \approx \mathbb{E}[J(x)]$
- **Cross validation:** select h that minimizes $\widehat{J}(h)$

Cross-validation

- Issue: the optimal bandwidth $h^\star$ depends on the unknown density f
- Idea: estimate the risk under each bandwidth selection h

$$L(h) := \int \left(\widehat{f}(x) - f(x)\right)^2 \mathrm{d}x = \underbrace{\int \widehat{f}^2(x)\mathrm{d}x - 2\int \widehat{f}(x) f(x)\mathrm{d}x}_{=:J(h)} + \int f^2(x)\mathrm{d}x$$

- Cross-validation estimate of the risk:

$$\widehat{J}(h) := \int \widehat{f}^2(x)\mathrm{d}x - \frac{2}{n}\sum_{i=1}^{n} \widehat{f}_{(-i)}(X_i)$$

- It can be shown that $\widehat{J}(h) \approx \mathbb{E}[J(x)]$
- **Cross validation:** select h that minimizes $\widehat{J}(h)$
- HW: prove the formula below that allows efficient computation of $\widehat{J}(h)$:

$$\widehat{J}(h) = \frac{2}{(n-1)h} - \frac{n+1}{n-1}\sum_{j=1}^{m} \widehat{p}_j^2$$

Limitation of the histogram method

- Histograms are discontinuous (not a continuous density)
- The convergence rate $O(n^{-2/3})$ is not ideal
- Complicated in higher dimension (number of bins will be exponential in dimension)
- A better solution: kernel density estimation