

Classification



Yuling Yan

University of Wisconsin-Madison, Fall 2024

Classification problem

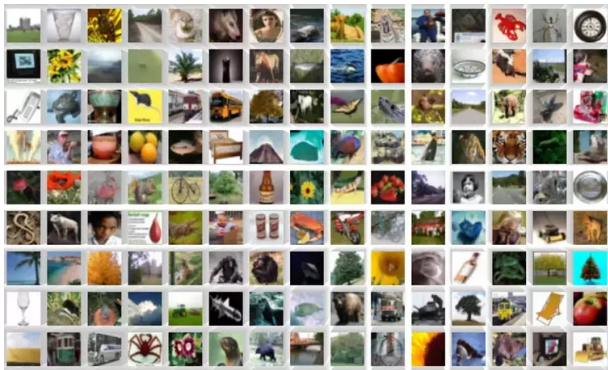
- Classification: assign a label (or category, class) to an observation based on its features
- \mathcal{X} : input space (e.g. \mathbb{R}^d); \mathcal{Y} : output space (e.g. $\{1, 2, \dots, K\}$)
- $x \in \mathcal{X}$: feature vector, input, data point...
- $y \in \mathcal{Y}$: label, category, class...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

MNIST dataset



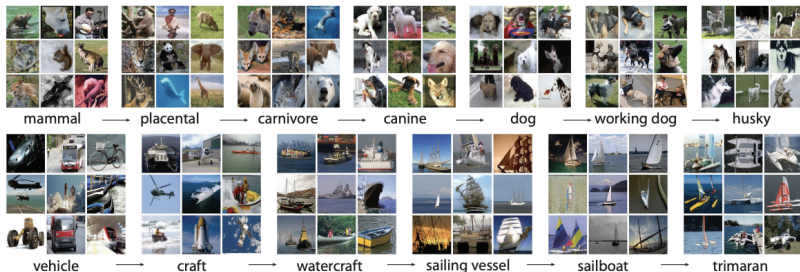
- Input: 28x28 gray scale (1 channel) images, i.e., $\mathcal{X} = \mathbb{R}^{28 \times 28}$ or \mathbb{R}^{784}
- Output: digits 0 through 9 (i.e., $\mathcal{Y} = \{0, 1, \dots, 9\}$)

CIFAR datasets



- Input: 32×32 RGB color (3 channels) images, i.e., $\mathcal{X} = \mathbb{R}^{32 \times 32 \times 3}$ 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: $(x_1, y_1), \dots, (x_n, y_n) \stackrel{\text{i.i.d.}}{\sim} \rho$
- 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 $x \in \mathcal{X}$,

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

Theorem 2.1 (Bayes optimal classifier: binary case)

The Bayes classifier f^ minimizes the misclassification error, i.e.,*

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

Proof of Theorem 2.1

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

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

By tower property,

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

It suffices to check

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

Proof of Theorem 2.1 (cont.)

Observe that

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

and

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

Therefore

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

A few remarks

Bayes optimal classifier

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

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

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

Bayes risk: binary case

- Bayes risk:

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

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

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

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

Bayes optimal classifier: multiclass setting

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

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

Theorem 2.2 (Bayes optimal classifier: multiclass case)

The Bayes classifier f^ minimizes the misclassification error, i.e.,*

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

Bayes optimal classifier: multiclass setting

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

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

Theorem 2.2 (Bayes optimal classifier: multiclass case)

The Bayes classifier f^ minimizes the misclassification error, i.e.,*

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

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

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

More general loss function?

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

Supervised learning

- In practice, we don't know ρ . It is in general impossible to compute the Bayes classifier f^\star
- Goal: build a classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ based on training data $(x_1, y_1), \dots, (x_n, y_n) \stackrel{\text{i.i.d.}}{\sim} \rho$
- 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 $\mathcal{H}_{\beta, \beta_0} = \{\mathbf{x} \in \mathbb{R}^d : \langle \beta, \mathbf{x} \rangle + \beta_0 = 0\}$
 - Half planes cut by $\mathcal{H}_{\beta, \beta_0}$:

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

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

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

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

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

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

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

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

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

- 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 $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$
- For each $1 \leq k \leq K$, let $n_k = \sum_{i=1}^n \mathbb{1}\{y_i = k\}$ and

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

- Estimate the covariance matrix

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

- Replace $\boldsymbol{\mu}_k, \omega_k, \boldsymbol{\Sigma}$ with $\hat{\boldsymbol{\mu}}_k, \hat{\omega}_k, \hat{\boldsymbol{\Sigma}}$

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

Generalization

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

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

where $\omega_k \geq 0$, $\sum_{k=1}^K \omega_k = 1$, $\mu_k \in \mathbb{R}^d$, $\Sigma_k \in \mathbb{S}_+^d$

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

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

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

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

Logistic regression

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

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

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

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

- Bayes classifier: $f(\mathbf{x}) = \arg \max_{0 \leq k \leq K} \mathbb{P}(Y = k \mid \mathbf{x})$
- Estimate $\boldsymbol{\beta}_k$'s: maximum likelihood estimation (MLE)

Maximum likelihood estimation

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

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

- Maximum likelihood estimation (MLE)

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

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

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

A brief introduction to gradient descent

Gradient descent (GD) for solving $\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta \nabla f(\mathbf{x}^t) \quad (t = 0, 1, \dots)$$

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

- For smooth function f , GD is a descent algorithm: $f(\mathbf{x}^{t+1}) \leq f(\mathbf{x}^t)$
- For convex + smooth function f , GD satisfies

$$f(\mathbf{x}^t) - f(\mathbf{x}^*) \leq O\left(\frac{\|\mathbf{x}^0 - \mathbf{x}^*\|_2^2}{t}\right) \quad (t = 0, 1, \dots)$$

- For strongly convex + smooth function f , GD satisfies

$$\|\mathbf{x}^{t+1} - \mathbf{x}^*\|_2 \leq (1 - \kappa)^t \|\mathbf{x}^0 - \mathbf{x}^*\|_2 \quad (t = 0, 1, \dots)$$

for some $\kappa \in (0, 1)$, where \mathbf{x}^* is the unique minimizer

Gradient descent

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

— simple and effective!

Recommended reading material: Lecture 1 of the course

Large-Scale Optimization for Data Science

by Prof. Yuxin Chen (UPenn); [Link](#)

Perceptrons

- For a hyperplane $\mathcal{H}_{\beta, \beta_0}$, let

$$f_{\beta, \beta_0}(x) := \begin{cases} 1, & \text{if } \langle \beta, \mathbf{x} \rangle + \beta_0 \geq 0, \\ -1, & \text{if } \langle \beta, \mathbf{x} \rangle + \beta_0 < 0. \end{cases}$$

- Define the set

$$\mathcal{M}_{\beta, \beta_0} := \{i : f_{\beta, \beta_0}(\mathbf{x}_i) \neq y_i\}$$

- Perceptron loss:

$$\sigma(\beta, \beta_0) := \sum_{i \in \mathcal{M}_{\beta, \beta_0}} \text{dist}(\mathbf{x}_i, \mathcal{H}_{\beta, \beta_0})$$

- Perceptron classifier: $f_{\hat{\beta}, \hat{\beta}_0}(x)$ where

$$(\hat{\beta}, \hat{\beta}_0) := \arg \min \sigma(\beta, \beta_0).$$