

Classification



Yuling Yan

University of Wisconsin-Madison, Fall 2024

Classification problem

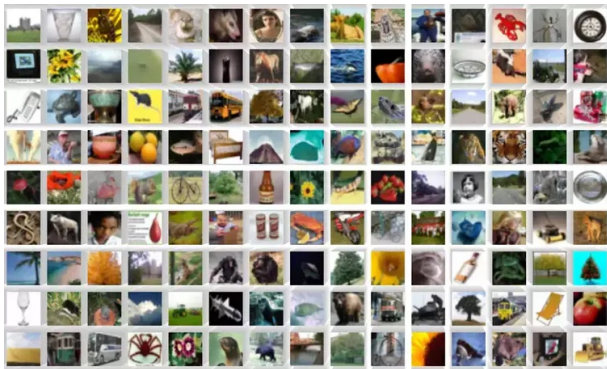
- Classification: assign a label (or category, class) to an observation based on its features
- \mathcal{X} : input space (e.g. \mathbb{R}^d); \mathcal{Y} : output space (e.g. $\{1, 2, \dots, K\}$)
- $x \in \mathcal{X}$: feature vector, input, data point...
- $y \in \mathcal{Y}$: label, category, class...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



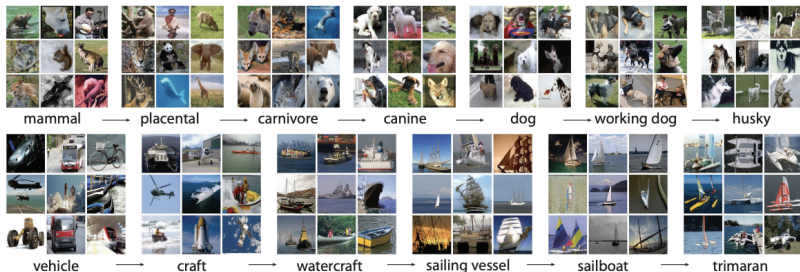
- Input: 28x28 gray scale (1 channel) images, i.e., $\mathcal{X} = \mathbb{R}^{28 \times 28}$ or \mathbb{R}^{784}
- Output: digits 0 through 9 (i.e., $\mathcal{Y} = \{0, 1, \dots, 9\}$)

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...

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 $f : \mathcal{X} \rightarrow \mathcal{Y}$ 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 \mathbf{x} , compute

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

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

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

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

- Estimate the covariance matrix

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

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

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

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

Maximum likelihood estimation

- Suppose we have i.i.d. data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$
- 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 $\min_{\beta \in \mathbb{R}^d} L(\beta)$:

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

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

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

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

for any minimizer β^*

- For strongly convex + smooth function L , GD satisfies

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

for some $\kappa \in (0, 1)$, where β^* is the unique minimizer

Stochastic gradient descent

Consider the following empirical risk minimization problem

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

where $\mathbf{x}_1, \dots, \mathbf{x}_n$ are training data points.

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

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

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

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

- **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 regression

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: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$
- **Linearly separable data:** \exists a separating hyperplane $\mathcal{H}_{\boldsymbol{\beta}, \beta_0}$ s.t.

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

Linearly separable data

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

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

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

$$y_i \cdot \mathbf{x}_i^\top \boldsymbol{\beta}_{\text{sep}} > 0 \quad (i = 1, \dots, n)$$

Linearly separable data

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

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

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

$$y_i \cdot \mathbf{x}_i^\top \boldsymbol{\beta}_{\text{sep}} > 0 \quad (i = 1, \dots, n)$$

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

$$y_i \cdot \mathbf{x}_i^\top \hat{\boldsymbol{\beta}} > 0 \quad (i = 1, \dots, n)$$

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

Perceptron Learning Algorithm

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

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

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

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

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

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

Perceptron Learning Algorithm

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

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

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

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

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

- 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 β , as they play different roles.
Consider the optimization problem

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

Optimal separating hyperplane

From now on, we “unmerge” β_0 from β , as they play different roles.
Consider the optimization problem

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

Implications:

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

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

Optimal separating hyperplane

From now on, we “unmerge” β_0 from β , as they play different roles.
Consider the optimization problem

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

Implications:

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

Optimal separating hyperplane

From now on, we “unmerge” β_0 from β , as they play different roles. Consider the optimization problem

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

Implications:

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

Reformulation as convex optimization

- **Original problem:**

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

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

Reformulation as convex optimization

- **Original problem:**

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

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

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

this is a convex optimization problem

Reformulation as convex optimization

- **Original problem:**

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

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

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

this is a convex optimization problem

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

SVMs for separable data

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

- SVM is a powerful method for binary classification
- finds a linear classifier with decision boundary $\{\mathbf{x} : \mathbf{x}^\top \hat{\beta} + \hat{\beta}_0 = 0\}$ 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:

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

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

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

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

- What is the benefit of introducing the Lagrangian function?

The Dual Problem

Key observation:

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

- relation (i) and (ii) always holds (why?)
- relation (ii) is often an equality (strong duality theory)
- The dual function $d(\boldsymbol{\lambda}) = \min_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda})$
- 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 x feasible in the primal and any $\lambda \geq 0$, we have:

$$d(\lambda) \leq f(x)$$

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

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

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

$$\exists x_0 \in \mathbb{R}^d \quad \text{s.t.} \quad g_i(x_0) < 0 \quad (i = 1, \dots, m).$$

- 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 (x, λ) is the optimal solution pair for the **primal**/**dual** problem

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

then

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

— This is a necessary condition!

Back to SVMs

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

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

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

- 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

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

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

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

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

Implications

For any optimal solution pair $(\beta^*, \beta_0^*, \alpha^*)$:

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

$$y_i(\beta^{*\top} \mathbf{x}_i + \beta_0^*) > 1 \implies \alpha_i = 0$$

$$\alpha_i > 0 \implies y_i(\beta^{*\top} \mathbf{x}_i + \beta_0^*) = 1$$

- **Recovering the primal solution:** after solving the dual problem (i.e., finding α_i^*), we can recover the primal solution (β^*, β_0^*) by

$$\beta^* = \sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i$$

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

— β^* is a linear combination of the support vectors

Accommodating non-separable data

SVM for linearly separable data:

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

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

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

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

Dual problem: non-separable data

- **Primal problem:**

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

- **Dual problem:**

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \quad (i = 1, \dots, n) \end{aligned}$$

- Homework: derive the **dual problem** from the **primal problem**

Dual problem: non-separable data

- **Primal problem:**

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

- **Dual problem:**

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \quad (i = 1, \dots, n) \end{aligned}$$

- Homework: derive the **dual problem** from the **primal problem**