Lecture 5: Linear Models for Classification

Tao LIN

SoE, Westlake University

March 24, 2025

Reading materials & Reference

Reading materials:

- Chapter 3, Stanford CS 229 Lecture Notes, https://cs229.stanford.edu/notes2022fall/main_notes.pdf
- Lecture 4, Stanford CS 231n, http://cs231n.stanford.edu/schedule.html

Reference:

- EPFL, CS-433 Machine Learning, https://github.com/epfml/ML_course

Table of Contents

Table of Contents

How do we trade-off the under-fitting and over-fitting?

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{1}$$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{1}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{2}$$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{1}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{2}$$

- f_S denotes the output.

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{1}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{2}$$

- f_S denotes the output.
- $\mathcal{A}$ denotes the learning algorithm.

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{1}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{2}$$

- f_S denotes the output.
- $\mathcal{A}$ denotes the learning algorithm.
- S denotes the input (dataset).

True Error, Empirical Error, and Training Error

- **True Error** (the **expected error** over all samples chosen according to $\mathcal{D}$):

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] . \tag{3}$$

We cannot estimate it due to the unknown $\mathcal{D}$.

True Error, Empirical Error, and Training Error

- **True Error** (the **expected error** over all samples chosen according to $\mathcal{D}$):

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] . \tag{3}$$

  We cannot estimate it due to the unknown $\mathcal{D}$.

- **Empirical Error** (what we can compute—the approximated true error via dataset S):

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) . \tag{4}$$

True Error, Empirical Error, and Training Error

- **True Error** (the **expected error** over all samples chosen according to $\mathcal{D}$):

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] . \tag{3}$$

  We cannot estimate it due to the unknown $\mathcal{D}$.

- **Empirical Error** (what we can compute—the approximated true error via dataset S):

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) . \tag{4}$$

- **Training Error** (what we are minimizing—when the model has been trained on the same data it is applied to):

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) . \tag{5}$$

True Error, Empirical Error, and Training Error

- **True Error** (the **expected error** over all samples chosen according to $\mathcal{D}$):

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] . \tag{3}$$

  We cannot estimate it due to the unknown $\mathcal{D}$.

- **Empirical Error** (what we can compute—the approximated true error via dataset S):

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) . \tag{4}$$

- **Training Error** (what we are minimizing—when the model has been trained on the same data it is applied to):

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) . \tag{5}$$

The reason that $L_S(f_S)$ might not be close $L_{\mathcal{D}}(f_S)$ is of course over-fitting.

Problem: validating the model on the same data subset we trained it on!

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{6}$$

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{6}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{6}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}
2. We **validate** it computing the error on the **test set** S_{test}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{7}$$

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{6}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}
2. We **validate** it computing the error on the **test set** S_{test}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{7}$$

- Since S_{train} and S_{test} are independent, we hope that $L_{S_{\text{test}}}(f_{S_{\text{train}}}) \approx L_{\mathcal{D}}(f_{S_{\text{train}}})$

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{6}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}
2. We **validate** it computing the error on the **test set** S_{test}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n,\mathbf{x}_n)\in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{7}$$

- Since S_{train} and S_{test} are independent, we hope that $L_{S_{\text{test}}}(f_{S_{\text{train}}}) \approx L_{\mathcal{D}}(f_{S_{\text{train}}})$
- Issues: we have fewer data both for the learning and validation tasks (trade-off)

Table of Contents

Generalization gap: How far is the test from the true error?

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{8}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{9}$$

- **Generalization Error:**

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right|. \tag{10}$$

Generalization gap: How far is the test from the true error?

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{8}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{9}$$

- **Generalization Error:**

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right| . \tag{10}$$

- **In Expectation:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{S_{\text{test}}\sim\mathcal{D}}\left[L_{S_{\text{test}}}(f)\right] , \tag{11}$$

Generalization gap: How far is the test from the true error?

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{8}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{9}$$

- **Generalization Error:**

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right|. \tag{10}$$

- **In Expectation:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{S_{\text{test}}\sim\mathcal{D}}\left[L_{S_{\text{test}}}(f)\right], \tag{11}$$

The variation of the $\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right|$ matters.

Theorem 1

Given a model f and a test set $S_{test} \sim \mathcal{D}$ i.i.d. (not used to learn f) and a loss $\ell(\cdot,\cdot) \in [a,b]$:

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{12}$$

Theorem 1

Given a model f and a test set $S_{test} \sim \mathcal{D}$ i.i.d. (not used to learn f) and a loss $\ell(\cdot,\cdot) \in [a,b]$:

$$\Pr\left[\left|L_{\mathcal{D}}(f) - L_{S_{test}}(f)\right| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\left|S_{test}\right|}}\right] \leq \delta \tag{12}$$

- The error decreases as $\mathcal{O}\left(1/\sqrt{S_{\text{test}}}\right)$ with the number test points.

Theorem 1

Given a model f and a test set $S_{test} \sim \mathcal{D}$ i.i.d. (not used to learn f) and a loss $\ell(\cdot,\cdot) \in [a,b]$:

$$\Pr\left[\left|L_{\mathcal{D}}(f) - L_{S_{test}}(f)\right| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\left|S_{test}\right|}}\right] \leq \delta \tag{12}$$

- The error decreases as $\mathcal{O}\left(1/\sqrt{S_{\text{test}}}\right)$ with the number test points.

⇒ The more data points we have, the more confident we are that the empirical loss we measure is close to the true loss.

Theorem 1

Given a model f and a test set $S_{test} \sim \mathcal{D}$ i.i.d. (not used to learn f) and a loss $\ell(\cdot,\cdot) \in [a,b]$:

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{12}$$

- The error decreases as $\mathcal{O}\left(1/\sqrt{S_{\text{test}}}\right)$ with the number test points.

⇒ The more data points we have, the more confident we are that the empirical loss we measure is close to the true loss.

Given a predictor f and a dataset S, we can control the expected risk:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{13}$$

How far is each of the K test errors $L_{S_{\text{test}}}(f_k)$ from the true $L_{\mathcal{D}}(f_k)$?

Theorem 2

We can bound the maximum deviation for all K candidates, by

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2|S_{test}|}}\right] \leq \delta \tag{14}$$

How far is each of the K test errors $L_{S_{\text{test}}}(f_k)$ from the true $L_{\mathcal{D}}(f_k)$?

Theorem 2

We can bound the maximum deviation for all K candidates, by

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{14}$$

- The error decreases as $\mathcal{O}(1/\sqrt{|S_{\text{test}}|})$ with the number test points.

How far is each of the K test errors $L_{S_{\text{test}}}(f_k)$ from the true $L_{\mathcal{D}}(f_k)$?

Theorem 2

We can bound the maximum deviation for all K candidates, by

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{14}$$

- The error decreases as $\mathcal{O}(1/\sqrt{|S_{\text{test}}|})$ with the number test points.
- When testing K hyper-parameters, the error only goes up by $\sqrt{\ln(K)}$.

How far is each of the K test errors $L_{S_{\text{test}}}(f_k)$ from the true $L_{\mathcal{D}}(f_k)$?

Theorem 2

We can bound the maximum deviation for all K candidates, by

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{14}$$

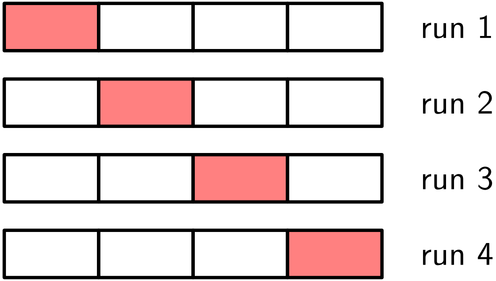
- The error decreases as $\mathcal{O}(1/\sqrt{|S_{\text{test}}|})$ with the number test points.
- When testing K hyper-parameters, the error only goes up by $\sqrt{\ln(K)}$.

$\Rightarrow$ So we can test many different models without incurring a large penalty.

Issues: Splitting the data once into two parts (one for training and one for testing) is not the most efficient way to use the data!

K-fold cross-validation:

1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing and use the remaining $K-1$ groups for training.
3. Average the K results

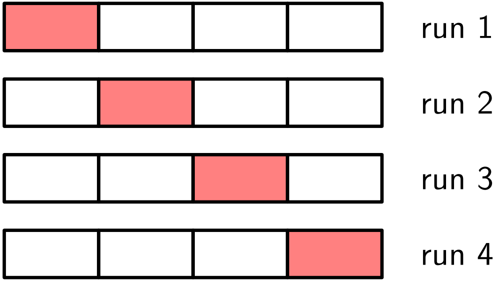


Benefits:

Issues: Splitting the data once into two parts (one for training and one for testing) is not the most efficient way to use the data!

K-fold cross-validation:

1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing and use the remaining $K - 1$ groups for training.
3. Average the K results



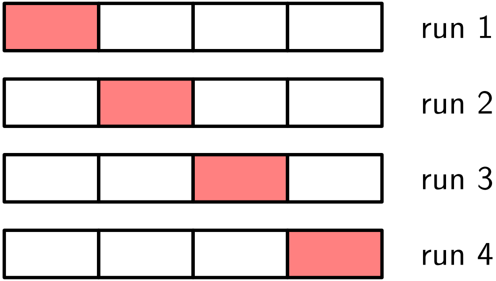
Benefits:

- We have used all data for training, and all data for testing, and used each data point the same number of times.

Issues: Splitting the data once into two parts (one for training and one for testing) is not the most efficient way to use the data!

K-fold cross-validation:

1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing and use the remaining $K-1$ groups for training.
3. Average the K results



Benefits:

- We have used all data for training, and all data for testing, and used each data point the same number of times.
- Cross-validation returns an unbiased estimate of the generalization error and its variance.

Table of Contents

Bias-Variance Decomposition

$$
\begin{aligned}
\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] = {} & \text{Var}_{\epsilon \sim \mathcal{D}_\epsilon} [\epsilon] && \text{(noise variance)} \\
& + \left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}'}} \left[f_{S_{\text{train}'}}(\mathbf{x}_0) \right] \right)^2 && \text{(Bias)} \\
& + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[\left(\mathbb{E}_{S_{\text{train}'}} \left[f_{S_{\text{train}'}}(\mathbf{x}_0) \right] - f_{S_{\text{train}}}(\mathbf{x}_0) \right)^2 \right], && \text{(Variance)}
\end{aligned}
$$

which always lower-bounds the true error.

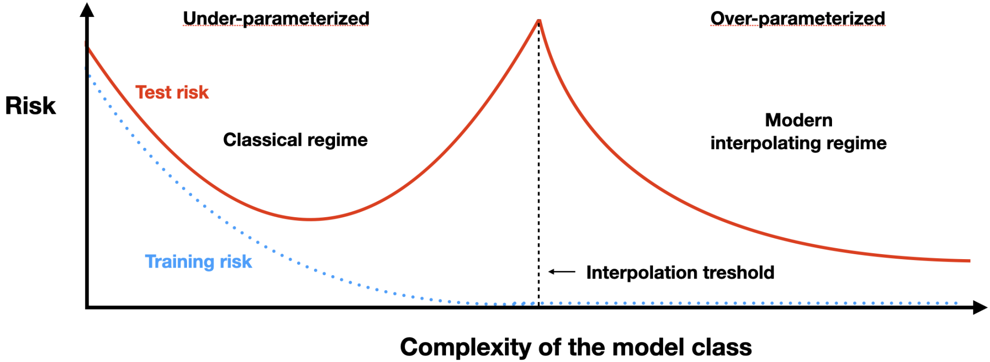
Bias-Variance Decomposition

$$
\begin{aligned}
\mathbb{E}_{S_{\text{train}} \in \mathcal{D}}\left[L(f_{S_{\text{train}}})\right] = {} & \text{Var}_{\epsilon \sim \mathcal{D}_\epsilon}[\epsilon] && \text{(noise variance)} \\
& + \left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right]\right)^2 && \text{(Bias)} \\
& + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[\left(\mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right], && \text{(Variance)}
\end{aligned}
$$

which always lower-bounds the true error.

$\Rightarrow$ To minimize the true error, we need to select a method that **simultaneously achieves low bias and low variance**.

Double descent curve in Deep Learning



Last lecture:

- Generalization Gap and Model Selection
- Bias-Variance Decomposition

Last lecture:

- Generalization Gap and Model Selection
- Bias-Variance Decomposition

This lecture:

- Logistic Regression
- Exponential Families and Generalized Linear Models

Table of Contents

Table of Contents

Definition of classification

We observe some data $\mathcal{S} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \underbrace{\mathcal{Y}}_{\text{Discrete set}}$

Definition of classification

We observe some data $\mathcal{S} = \{\mathbf{x}_n, y_n\}_{n=1}^N \in \mathcal{X} \times \underbrace{\mathcal{Y}}_{\text{Discrete set}}$

Goal: given a new observation $\mathbf{x}$, we want to predict its label y.

Definition of classification

We observe some data $\mathcal{S} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \underbrace{\mathcal{Y}}_{\text{Discrete set}}$

Goal: given a new observation $\mathbf{x}$, we want to predict its label y.

How: relates input to a categorical variable

Classifier

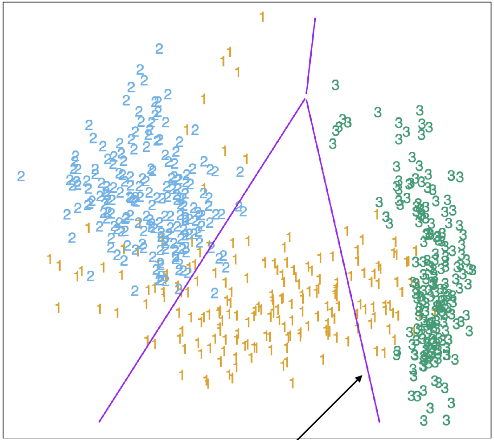
A classifier $f : \mathcal{X} \to \mathcal{Y}$

Classifier

A classifier $f : \mathcal{X} \to \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.

Classifier

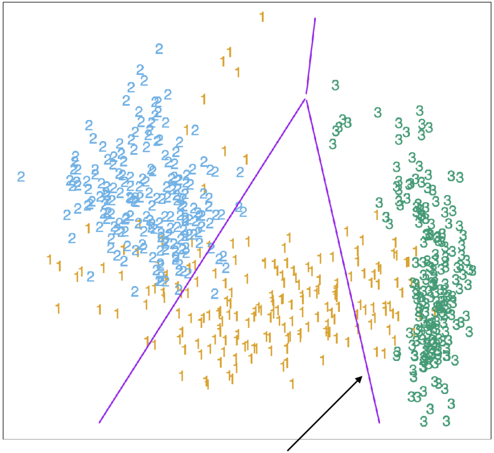
A classifier $f : \mathcal{X} \to \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



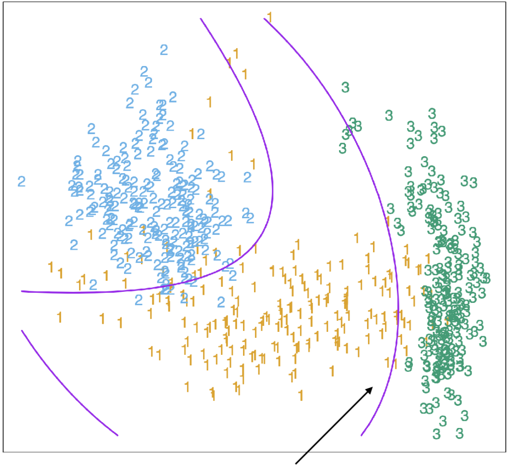
Linear Decision boundary

Classifier

A classifier $f : \mathcal{X} \to \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



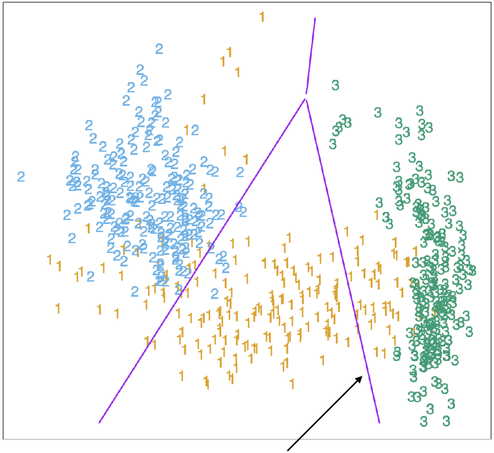
Linear Decision boundary



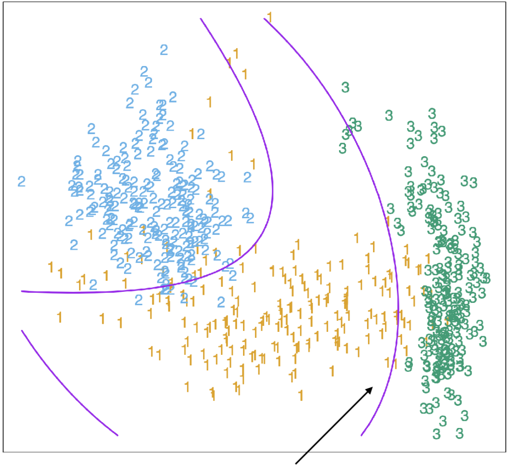
Nonlinear Decision boundary

Classifier

A classifier $f : \mathcal{X} \to \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



Linear Decision boundary



Nonlinear Decision boundary

Classification: a special case of regression?

Classification is a **regression problem** with discrete labels:

$$(\mathbf{x}, y) \in \mathcal{X} \times \{0, 1\} \subset \mathcal{X} \times \mathbb{R} \tag{15}$$

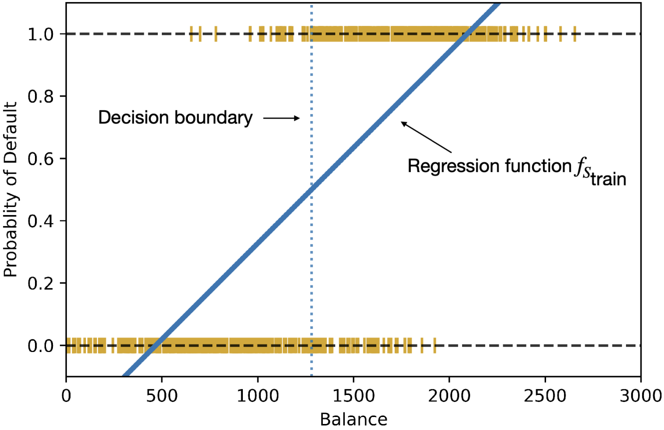
Classification: a special case of regression?

Classification is a **regression problem** with discrete labels:

$$(\mathbf{x}, y) \in \mathcal{X} \times \{0, 1\} \subset \mathcal{X} \times \mathbb{R} \tag{15}$$

Could we use previously seen regression methods to solve it?

Is it a good idea to use some regression methods?



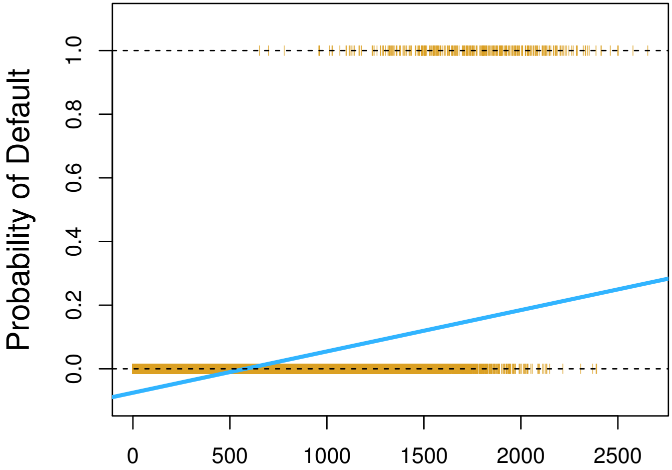
$y = 1$ **if "default"**

$y = 0$ **if "no default"**

We label the output as probability for sake of interpretation

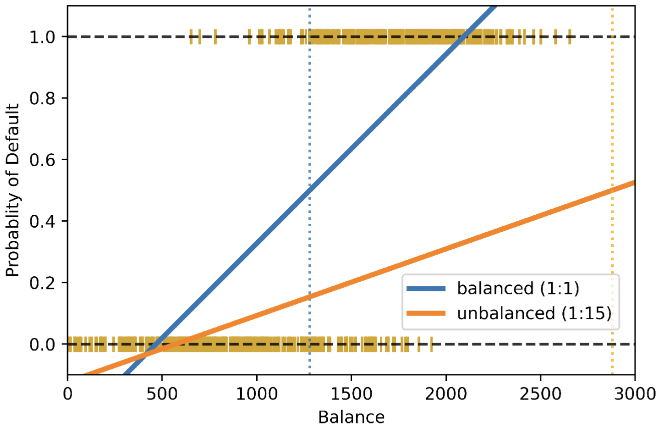
Classification is not just a special form of regression

- The predicted values are not probabilities (not in $[0, 1]$)



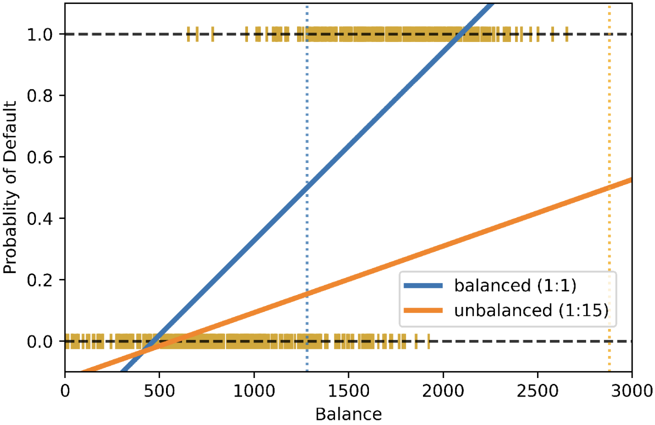
Classification is not just a special form of regression

- Sensitivity to unbalanced data



Classification is not just a special form of regression

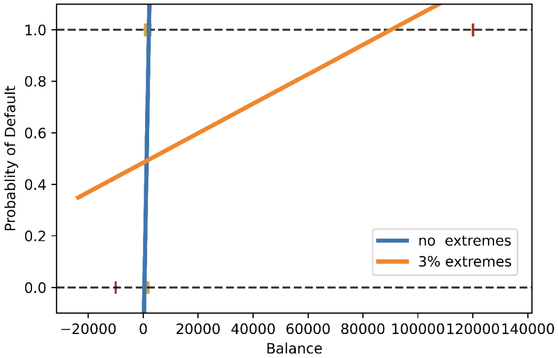
- Sensitivity to unbalanced data

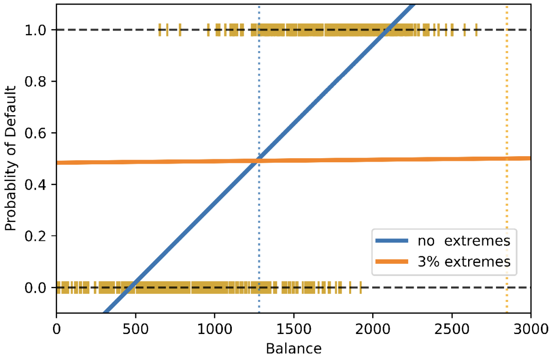


The position of the line depends crucially on how many points are in each class.

Classification is not just a special form of regression

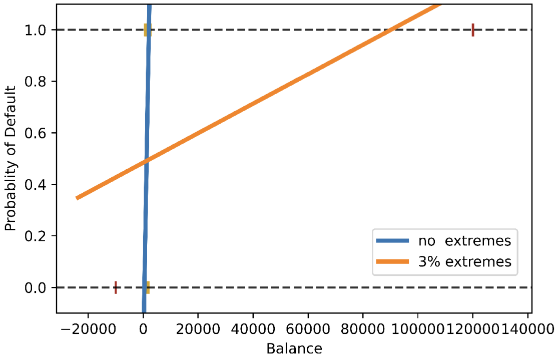
- Sensitivity to extreme values:

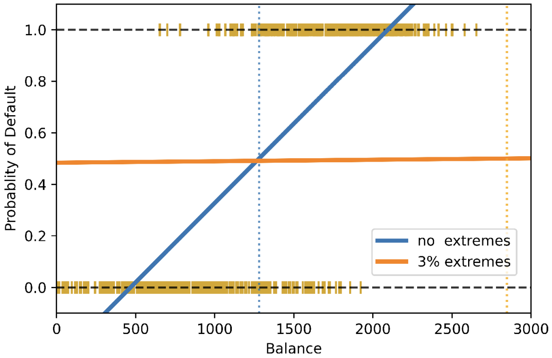




Classification is not just a special form of regression

- Sensitivity to extreme values:





The position of the line depends crucially on where the points lie.

How to perform classification?

- Many approaches have been developed

How to perform classification?

- Many approaches have been developed
- We won’t cover them in detail today

How to perform classification?

- Many approaches have been developed
- We won't cover them in detail today
- Instead, we will provide quick introductions

How to perform classification?

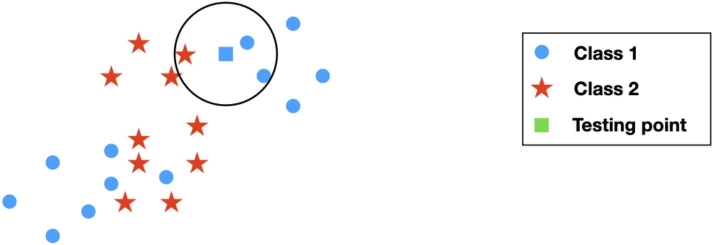
- Many approaches have been developed
- We won't cover them in detail today
- Instead, we will provide quick introductions

- Fundamental task of classification:

Divide the space into distinct decision regions

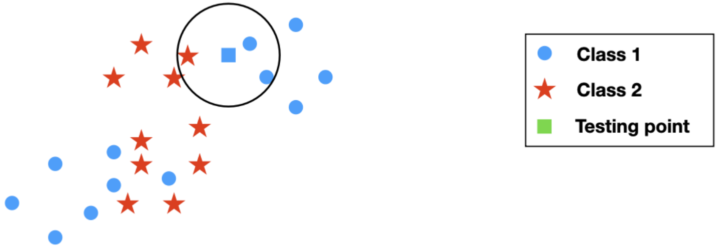
k-Nearest Neighbor

Assume that nearby points are likely to have similar labels



k-Nearest Neighbor

Assume that nearby points are likely to have similar labels



A new point $\mathbf{x}$ is classified based on the **majority vote of its k-nearest neighbors**.

k-Nearest Neighbor

Pros:

Cons:

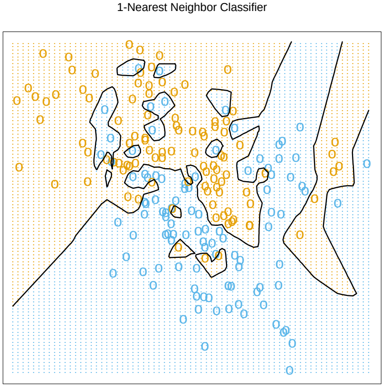


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by 1-nearest-neighbor classification.*

k-Nearest Neighbor

Pros:

- No **optimization** or training

Cons:

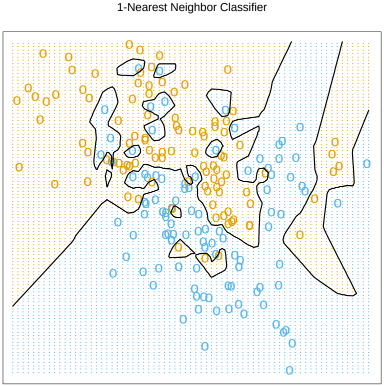


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (*BLUE $= 0$*,* ORANGE $= 1$*), and then predicted by 1-nearest-neighbor classification.*

k-Nearest Neighbor

Pros:

- No **optimization** or training
- **Easy** to implement

Cons:

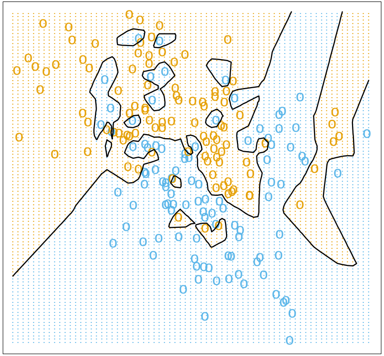


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then predicted by 1-nearest-neighbor classification.*

k-Nearest Neighbor

Pros:

- No **optimization** or training
- **Easy** to implement
- Works well in **low dimensions**, allowing for very complex decision boundaries

Cons:

1-Nearest Neighbor Classifier

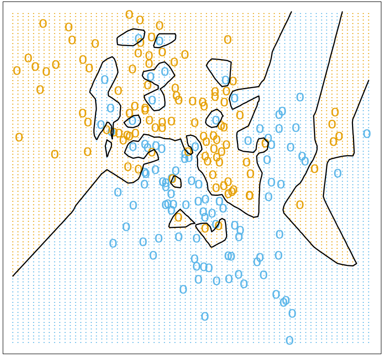


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

k-Nearest Neighbor

Pros:

- No **optimization** or training
- **Easy** to implement
- Works well in **low dimensions**, allowing for very complex decision boundaries

Cons:

- **Slow** at query time

1-Nearest Neighbor Classifier

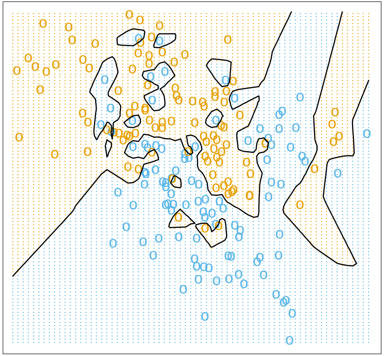


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then predicted by 1-nearest-neighbor classification.*

k-Nearest Neighbor

Pros:

- No **optimization** or training
- **Easy** to implement
- Works well in **low dimensions**, allowing for very complex decision boundaries

Cons:

- **Slow** at query time
- Not suitable for high-dimensional data

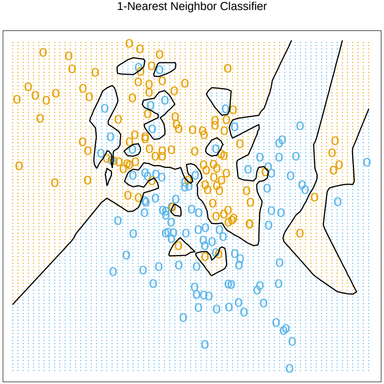


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by 1-nearest-neighbor classification.*

k-Nearest Neighbor

Pros:

- No **optimization** or training
- **Easy** to implement
- Works well in **low dimensions**, allowing for very complex decision boundaries

Cons:

- **Slow** at query time
- Not suitable for high-dimensional data
- Choosing the right local distance is crucial

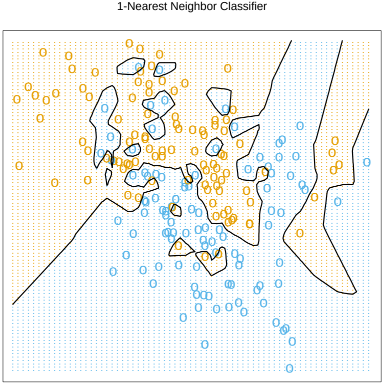


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by 1-nearest-neighbor classification.*

Linear Decision boundaries

Assume we restrict ourselves to linear decision boundaries (hyperplane):

Linear Decision boundaries

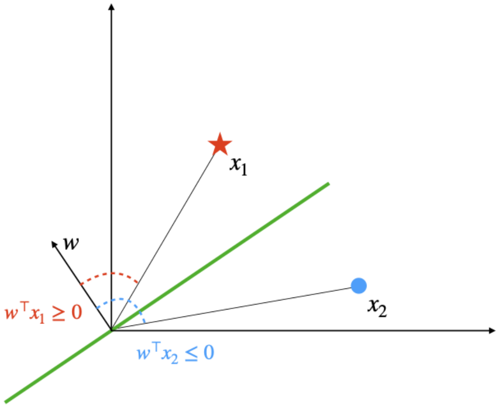
Assume we restrict ourselves to linear decision boundaries (hyperplane):

$$\Rightarrow \text{Prediction:} f(\mathbf{x}) = \text{sign}(\mathbf{x}^\top \mathbf{w})$$

Linear Decision boundaries

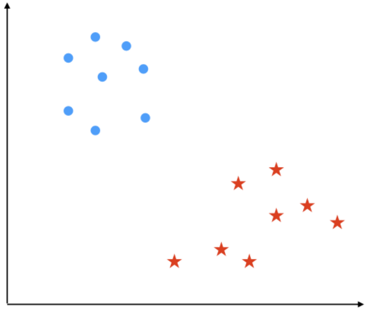
Assume we restrict ourselves to linear decision boundaries (hyperplane):

$$\Rightarrow \text{Prediction:} f(\mathbf{x}) = \text{sign}(\mathbf{x}^\top \mathbf{w})$$



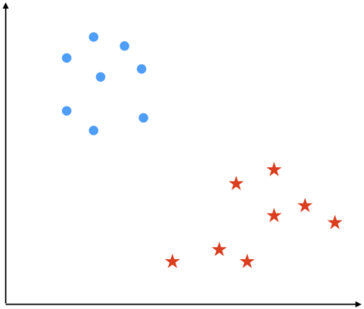
Separating hyperplane

Assume we restrict ourselves to linear decision boundaries (hyperplane):



Separating hyperplane

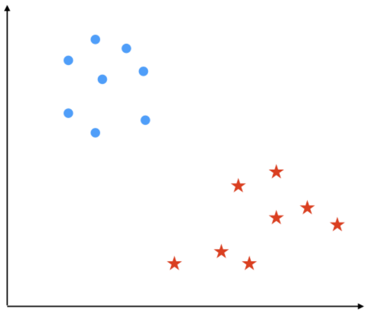
Assume we restrict ourselves to linear decision boundaries (hyperplane):



Assume the data are linearly separable, i.e., a separating hyperplane exists

Separating hyperplane

Assume we restrict ourselves to linear decision boundaries (hyperplane):

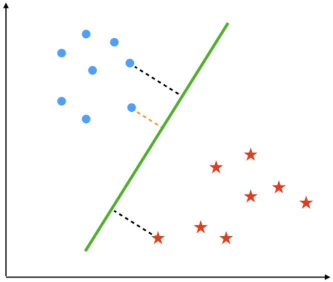


Assume the data are linearly separable, i.e., a separating hyperplane exists

> Which separating hyperplane would you pick?

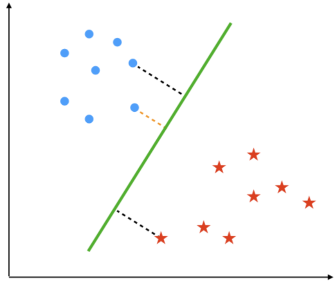
Margin

Key concept: The margin is the distance from the hyperplane to the closest point.



Margin

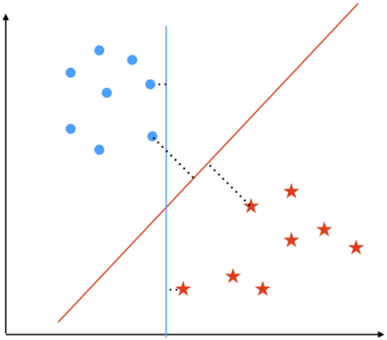
Key concept: The margin is the distance from the hyperplane to the closest point.



Take the one with the largest margin!

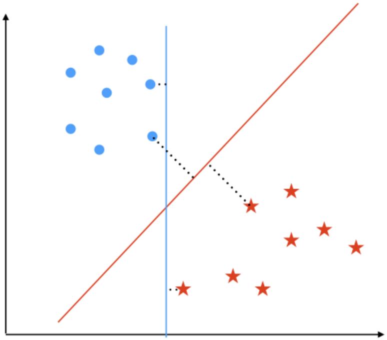
Max-margin separating hyperplane

Choose the hyperplane which maximizes the margin



Max-margin separating hyperplane

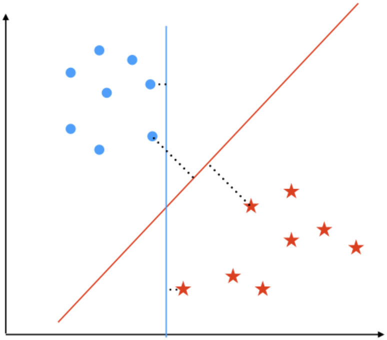
Choose the hyperplane which maximizes the margin



WHY?

Max-margin separating hyperplane

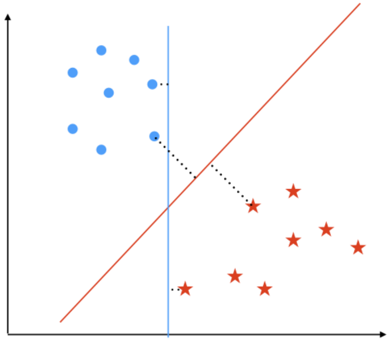
Choose the hyperplane which maximizes the margin



WHY? If we slightly change the training set, the number of misclassifications will stay low

Max-margin separating hyperplane

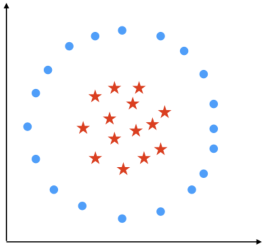
Choose the hyperplane which maximizes the margin



WHY? If we slightly change the training set, the number of misclassifications will stay low

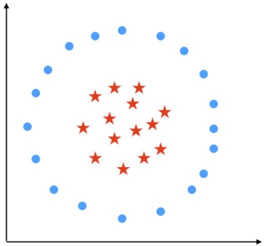
⇒ It will lead us to support vector machine (SVM) and logistic regression!

Nonlinear classifier



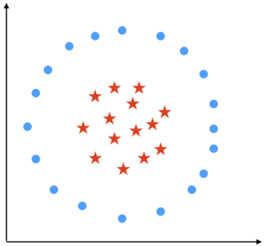
Nonlinear classifier

- Linear decision boundaries will not always work



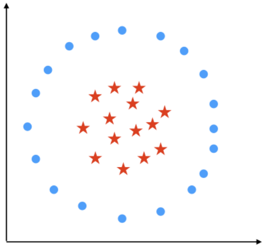
Nonlinear classifier

- Linear decision boundaries will not always work
- Feature augmentation (x, x^2, x^3, x^4)



Nonlinear classifier

- Linear decision boundaries will not always work
- Feature augmentation (x, x^2, x^3, x^4)
- Kernel method



Formalizing binary classification

Formalizing binary classification

- Setting: $(X, Y) \sim \mathcal{D}$ with ranges $\mathcal{X}, \mathcal{Y} = \{-1, 1\}$

Formalizing binary classification

- Setting: $(X, Y) \sim \mathcal{D}$ with ranges $\mathcal{X}, \mathcal{Y} = \{-1, 1\}$
- Loss function: $(0 - 1$ loss) $\ell(y, y') = 1_{y \neq y'}$

Formalizing binary classification

- Setting: $(X, Y) \sim \mathcal{D}$ with ranges $\mathcal{X}, \mathcal{Y} = \{-1, 1\}$
- Loss function: $(0-1$ loss) $\ell(y, y') = 1_{y \neq y'}$
- True risk for the classification:

$$L_{\mathcal{D}}(f) = \underbrace{\mathbb{E}_{\mathcal{D}}\left[1_{Y \neq f(X)}\right]}_{\text{classification error}} = \underbrace{\Pr_{\mathcal{D}}\left[Y \neq f(X)\right]}_{\text{probability of making an error}}$$

Formalizing binary classification

- Setting: $(X, Y) \sim \mathcal{D}$ with ranges $\mathcal{X}, \mathcal{Y} = \{-1, 1\}$
- Loss function: $(0 - 1$ loss$)$ $\ell(y, y') = 1_{y \neq y'}$
- True risk for the classification:

$$L_{\mathcal{D}}(f) = \underbrace{\mathbb{E}_{\mathcal{D}}\left[1_{Y \neq f(X)}\right]}_{\text{classification error}} = \underbrace{\Pr_{\mathcal{D}}\left[Y \neq f(X)\right]}_{\text{probability of making an error}}$$

- Goal: minimize $L_{\mathcal{D}}(f)$

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):
  If we want to maximize the probability of guessing the correct label,

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):
  If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the “correct” label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):
  If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{16}$$

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):
  If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{16}$$

This classifier is also called the *Bayes classifier*: $f^\star = \arg\min_f L_\mathcal{D}(f)$, where

$$f^\star(\mathbf{x}) \in \arg\max_{\{-1,1\}} \Pr(Y = y|X = \mathbf{x}) \tag{17}$$

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):
  If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{16}$$

  This classifier is also called the *Bayes classifier*: $f^\star = \arg\min_f L_\mathcal{D}(f)$, where

$$f^\star(\mathbf{x}) \in \arg\max_{\{-1,1\}} \Pr(Y = y | X = \mathbf{x}) \tag{17}$$

- In practice, we do not know the joint distribution $p(\mathbf{x}, y)$

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):
  If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{16}$$

  This classifier is also called the *Bayes classifier*: $f^\star = \arg\min_f L_{\mathcal{D}}(f)$, where

$$f^\star(\mathbf{x}) \in \arg\max_{\{-1,1\}} \Pr(Y = y | X = \mathbf{x}) \tag{17}$$

- In practice, we do not know the joint distribution $p(\mathbf{x}, y)$

  $\Rightarrow$ Bayes classifier is an unattainable gold standard.

Optimal classification for known generating model

What is the **optimal performance**, regardless of the finiteness of the training data?

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):
  If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{16}$$

  This classifier is also called the *Bayes classifier*: $f^\star = \arg\min_f L_{\mathcal{D}}(f)$, where

$$f^\star(\mathbf{x}) \in \arg\max_{\{-1,1\}} \Pr(Y = y|X = \mathbf{x}) \tag{17}$$

- In practice, we do not know the joint distribution $p(\mathbf{x}, y)$

  $\Rightarrow$ Bayes classifier is an unattainable gold standard.

  But we can use the data to learn the distribution (by assuming the data distribution).

Proof of the Bayes classifier

Claim 1: $\forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}^\star) \in \arg\min_{y \in \mathcal{Y}} \Pr(Y \neq y | X = \mathbf{x}) \implies f^\star \in \arg\min_{f:\mathcal{X}\to\mathcal{Y}} L_{\mathcal{D}}(f)$

$$
\begin{aligned}
L_{\mathcal{D}}(f) = \mathbb{E}_{X,Y}\left[1_{Y \neq f(X)}\right] &= \mathbb{E}_X\left[\mathbb{E}_{Y|X}\left[1_{Y\neq X}|X\right]\right] && (18)\\
&= \mathbb{E}_X\left[\Pr(Y \neq f(X)|X)\right] && (19)\\
&\geq \mathbb{E}_X\left[\min_{y\in\mathcal{Y}} \Pr(Y \neq y|X)\right] && (20)\\
&= \mathbb{E}_X\left[\Pr(Y \neq f^\star(X)|X)\right] = \mathbb{E}_{X,Y}\left[1_{Y\neq f^\star(X)}\right] = L_{\mathcal{D}}(f^\star) && (21)
\end{aligned}
$$

Claim 2: $f^\star(\mathbf{x}) \in \arg\min_{y\in\mathcal{Y}} \Pr(Y \neq y|X = \mathbf{x})$

$$
f^\star(\mathbf{x}) \in \arg\max_{y\in\mathcal{Y}} \Pr(Y = y|X = \mathbf{x}) = \arg\min_{y\in\mathcal{Y}} \Pr(Y \neq y|X = \mathbf{x}) \qquad (22)
$$

Two classes of classification algorithms

- **Non-parametric:**
- **Parametric:**

Two classes of classification algorithms

- **Non-parametric:** approximate the conditional distribution $\Pr(Y = y|X = \mathbf{x})$ via local averaging
- **Parametric:**

Two classes of classification algorithms

- **Non-parametric:** approximate the conditional distribution $\Pr(Y = y|X = \mathbf{x})$ via local averaging $\Rightarrow$ Follow nearest neighbors' decisions (KNN).
- **Parametric:**

Two classes of classification algorithms

- **Non-parametric:** approximate the conditional distribution $\Pr(Y = y|X = \mathbf{x})$ via local averaging $\Rightarrow$ Follow nearest neighbors' decisions (KNN).
- **Parametric:** approximate true distribution $\mathcal{D}$ via training data

Two classes of classification algorithms

- **Non-parametric:** approximate the conditional distribution $\Pr(Y = y|X = \mathbf{x})$ via local averaging
  $\Rightarrow$ Follow nearest neighbors' decisions (KNN).

- **Parametric:** approximate true distribution $\mathcal{D}$ via training data
  $\Rightarrow$ Minimize the empirical risk on training data (ERM)

Recap

- Classification:
  - Mapping inputs to discrete outputs (categorical)
  - Not a special form of regression!
- Ways to perform classification:
  - **Non-parametric**: K-Nearest-Neighbors
  - **Parametric**: learning X-to-Y mapping via ERM
  - More complex decision boundaries? Non-linear classifiers

Table of Contents

Motivation for Logistic Regression

Rather than modeling the output Y directly,

Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

- The predicted value is not in $[0, 1]$.

Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

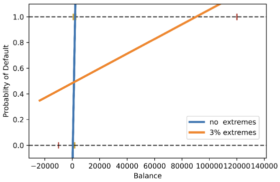
- The predicted value is not in $[0, 1]$.
- Very large ($y \gg 1$) or very small ($y \ll 0$) values of the prediction will contribute to the error if we use the squared loss.

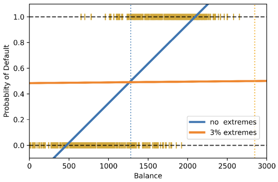
Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

- The predicted value is not in $[0, 1]$.
- Very large ($y \gg 1$) or very small ($y \ll 0$) values of the prediction will contribute to the error if we use the squared loss.



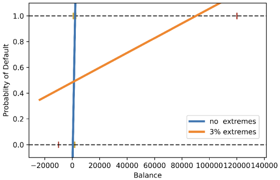


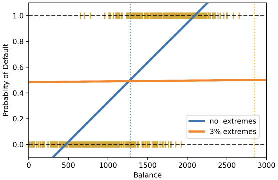
Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

- The predicted value is not in $[0, 1]$.
- Very large ($y \gg 1$) or very small ($y \ll 0$) values of the prediction will contribute to the error if we use the squared loss.

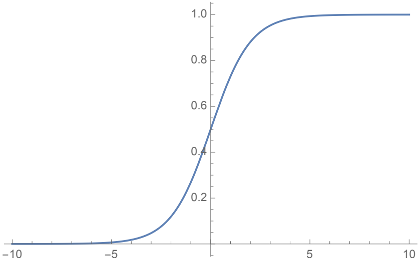




Solution: Transforming the predictions that take values in $(-\infty, \infty)$ into $[0, 1]$.

The logistic function

Consider first of all the case of two classes.

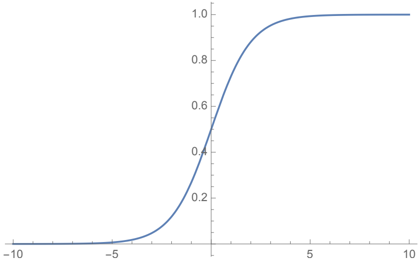


The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{23}$$

(24)

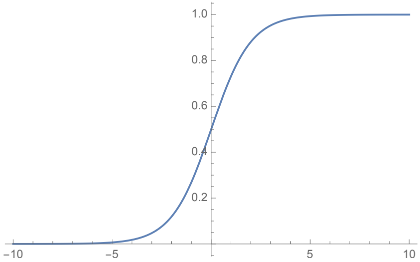


The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (23)$$

$$= \frac{1}{1 + \exp(-\eta)} \quad (24)$$

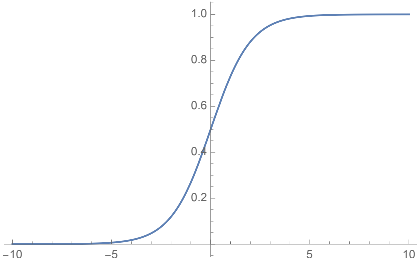


The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{23}$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{24}$$



 Logistic Regression

The logistic function

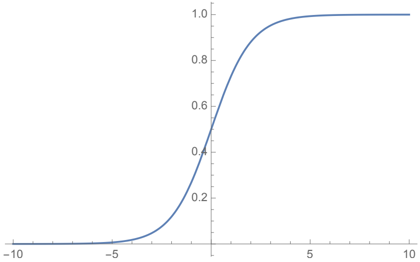
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{23}$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{24}$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{25}$$



The logistic function

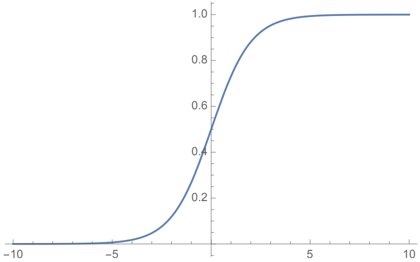
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (23)$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (24)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (25)$$



The logistic function

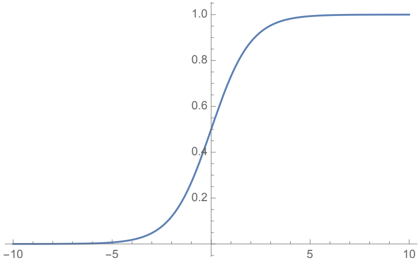
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (23)$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (24)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (25)$$



Properties of the logistic function (Exercise):

The logistic function

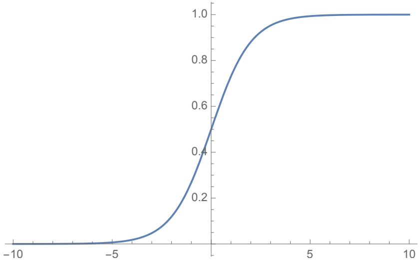
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{23}$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{24}$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \tag{25}$$



Properties of the logistic function (Exercise):

- $1 - \sigma(\eta) = \sigma(-\eta)$

The logistic function

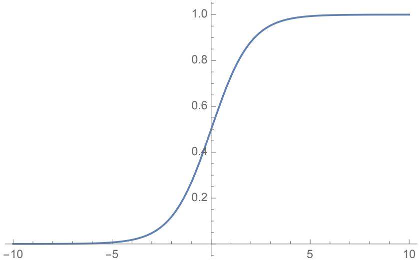
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{23}$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{24}$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \tag{25}$$



Properties of the logistic function (Exercise):

- $1 - \sigma(\eta) = \sigma(-\eta)$
- $\sigma'(\eta) = \sigma(\eta)\,(1 - \sigma(\eta))$

The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

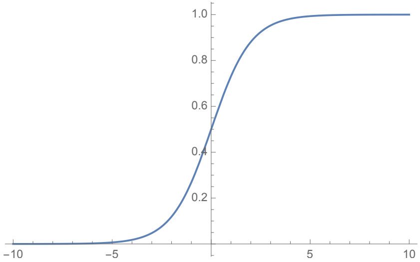
$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (23)$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (24)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (25)$$

(Exercise) For the case of $K > 2$ classes:



Properties of the logistic function (Exercise):

- $1 - \sigma(\eta) = \sigma(-\eta)$
- $\sigma'(\eta) = \sigma(\eta)\,(1 - \sigma(\eta))$

The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{23}$$

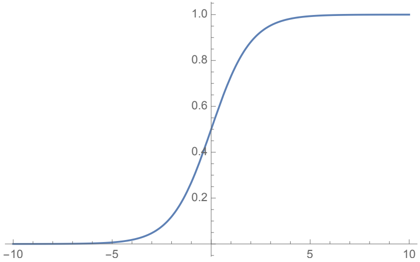
$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{24}$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \tag{25}$$

(Exercise) For the case of $K > 2$ classes:

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \tag{26}$$



Properties of the logistic function (Exercise):

- $1 - \sigma(\eta) = \sigma(-\eta)$
- $\sigma'(\eta) = \sigma(\eta)\,(1 - \sigma(\eta))$

The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{23}$$

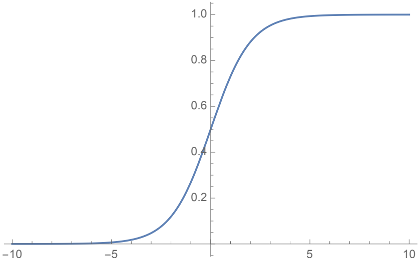
$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{24}$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \tag{25}$$

(Exercise) For the case of $K > 2$ classes:

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(\eta_k)}{\sum_j \exp(\eta_j)} \tag{26}$$



Properties of the logistic function (Exercise):

- $1 - \sigma(\eta) = \sigma(-\eta)$
- $\sigma'(\eta) = \sigma(\eta)\,(1 - \sigma(\eta))$

Logistic Regression

Given a "new" feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] \tag{27}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] \quad , \tag{28}$$

Logistic Regression

Given a "new" feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] \; = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{27}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] \qquad\qquad , \tag{28}$$

Logistic Regression

Given a "new" feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{27}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \ , \tag{28}$$

Logistic Regression

Given a "new" feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] \; = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{27}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] \; = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \; , \tag{28}$$

where we predict a real value (a probability) and not a label.

Logistic Regression

Given a “new” feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{27}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right)\ , \tag{28}$$

where we predict a real value (a probability) and not a label.

Label prediction: quantize the probability

$$\text{if } p(1|\mathbf{x}) \geq 1/2 \Rightarrow \text{predict the class } 1 \tag{29}$$

$$\text{if } p(1|\mathbf{x}) < 1/2 \Rightarrow \text{predict the class } 0 \tag{30}$$

Logistic Regression

Given a “new” feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{27}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \, , \tag{28}$$

where we predict a real value (a probability) and not a label.

Label prediction: quantize the probability

$$\text{if } p(1|\mathbf{x}) \geq 1/2 \Rightarrow \text{predict the class } 1 \tag{29}$$

$$\text{if } p(1|\mathbf{x}) < 1/2 \Rightarrow \text{predict the class } 0 \tag{30}$$

Interpretation:

Logistic Regression

Given a "new" feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr[Y = 1|\mathbf{X} = \mathbf{x}] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{27}$$

$$p(0|\mathbf{x}) := \Pr[Y = 0|\mathbf{X} = \mathbf{x}] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) , \tag{28}$$

where we predict a real value (a probability) and not a label.

Label prediction: quantize the probability

$$\text{if } p(1|\mathbf{x}) \geq 1/2 \Rightarrow \text{predict the class } 1 \tag{29}$$

$$\text{if } p(1|\mathbf{x}) < 1/2 \Rightarrow \text{predict the class } 0 \tag{30}$$

Interpretation:

- Very large $\mathbf{x}^\top \mathbf{w} + w_0$ corresponds to $p(1|\mathbf{x})$ very close to 0 or 1 (high confidence).

Logistic Regression

Given a “new” feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{27}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) , \tag{28}$$

where we predict a real value (a probability) and not a label.

Label prediction: quantize the probability

$$\text{if } p(1|\mathbf{x}) \geq 1/2 \Rightarrow \text{predict the class } 1 \tag{29}$$

$$\text{if } p(1|\mathbf{x}) < 1/2 \Rightarrow \text{predict the class } 0 \tag{30}$$

Interpretation:

- Very large $\mathbf{x}^\top \mathbf{w} + w_0$ corresponds to $p(1|\mathbf{x})$ very close to 0 or 1 (high confidence).
- Small $\left|\mathbf{x}^\top \mathbf{w} + w_0\right|$ corresponds to $p(1|\mathbf{x})$ very close to 0.5 (low confidence).

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ (fixed but unknown $\mathcal{D}$).

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ (fixed but unknown $\mathcal{D}$).
The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{31}$$

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ (fixed but unknown $\mathcal{D}$).
The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^{N} | \mathbf{w}) \tag{31}$$

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).
The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N | \mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \,, \tag{31}$$

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ (fixed but unknown $\mathcal{D}$).
The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^{N} | \mathbf{w}) = \prod_{n=1}^{N} p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \,, \tag{31}$$

or equivalently,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} [-\log \mathcal{L}(\mathbf{w})] = \arg\min \sum_{n=1}^{N} -\log \left(p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \right) \,. \tag{32}$$

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).
The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N|\mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\}|\mathbf{w}) \,, \tag{31}$$

or equivalently,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}}[-\log \mathcal{L}(\mathbf{w})] = \arg\min \sum_{n=1}^N -\log\left(p(\{\mathbf{x}_n, y_n\}|\mathbf{w})\right) \,. \tag{32}$$

This estimator is **consistent** (under mild condition):

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).
The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N | \mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \,, \tag{31}$$

or equivalently,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} [-\log \mathcal{L}(\mathbf{w})] = \arg\min \sum_{n=1}^N - \log \left(p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \right) \,. \tag{32}$$

This estimator is **consistent** (under mild condition):
$\Longrightarrow$ if the data are generated according to the model,

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).
The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N|\mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\}|\mathbf{w})\,, \tag{31}$$

or equivalently,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}}[-\log \mathcal{L}(\mathbf{w})] = \arg\min \sum_{n=1}^N -\log\left(p(\{\mathbf{x}_n, y_n\}|\mathbf{w})\right)\,. \tag{32}$$

This estimator is **consistent** (under mild condition):
$\Longrightarrow$ if the data are generated according to the model,

the MLE converges to the true parameter when $N \to \infty$.

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w})\,, \tag{33}$$

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \,, \tag{33}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \, , \tag{33}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

For Logistic Regression, we have:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n)$$

(35)

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \,, \tag{33}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

For Logistic Regression, we have:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n) = \prod_{n:y_n=1} p(y_n = 1|\mathbf{x}_n) \prod_{n:y_n=0} p(y_n = 0|\mathbf{x}_n) \tag{34}$$

$$\tag{35}$$

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \,, \tag{33}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

For Logistic Regression, we have:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n) = \prod_{n:y_n=1} p(y_n = 1|\mathbf{x}_n) \prod_{n:y_n=0} p(y_n = 0|\mathbf{x}_n) \tag{34}$$

$$= \prod_{n=1}^{N} \sigma(\mathbf{x}_n^\top \mathbf{w})^{y_n} \left[1 - \sigma(\mathbf{x}_n^\top \mathbf{w})\right]^{1-y_n} \tag{35}$$

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \,, \tag{33}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

For Logistic Regression, we have:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n) = \prod_{n:y_n=1} p(y_n = 1|\mathbf{x}_n) \prod_{n:y_n=0} p(y_n = 0|\mathbf{x}_n) \tag{34}$$

$$= \prod_{n=1}^{N} \sigma(\mathbf{x}_n^\top \mathbf{w})^{y_n} \left[1 - \sigma(\mathbf{x}_n^\top \mathbf{w})\right]^{1-y_n} \tag{35}$$

As a result,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} \left(- \log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{36}$$

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{37}$$

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{37}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) \tag{38}$$

$$\tag{39}$$

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{37}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) \tag{38}$$

$$= \frac{1}{N} \mathbf{X}^\top \left[\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y} \right] , \tag{39}$$

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{37}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) \tag{38}$$

$$= \frac{1}{N} \mathbf{X}^\top \left[\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y} \right] , \tag{39}$$

where $\mathbf{X} \in \mathbb{R}^{N \times d}$.

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{37}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) \tag{38}$$

$$= \frac{1}{N} \mathbf{X}^\top \left[\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y} \right] , \tag{39}$$

where $\mathbf{X} \in \mathbb{R}^{N \times d}$.

$\Rightarrow$ It has no closed-form solution to $\nabla \mathcal{L}(\mathbf{w}) = 0$.

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{37}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) \tag{38}$$

$$= \frac{1}{N} \mathbf{X}^\top \left[\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y} \right], \tag{39}$$

where $\mathbf{X} \in \mathbb{R}^{N \times d}$.

$\Rightarrow$ It has no closed-form solution to $\nabla \mathcal{L}(\mathbf{w}) = 0$.

$\Rightarrow$ Good news: the cost function L is convex.

Convexity of the loss function L

- Claim: The function

$$\mathcal{L}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} -y_n\mathbf{x}_n^\top\mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top\mathbf{w}}) \tag{40}$$

is convex in the weight vector $\mathbf{w}$

Convexity of the loss function L

- Claim: The function

$$\mathcal{L}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} -y_n\mathbf{x}_n^\top\mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top\mathbf{w}}) \tag{40}$$

  is convex in the weight vector $\mathbf{w}$

- Proof: $\mathcal{L}$ is obtained through simple convexity preserving operations:

Convexity of the loss function L

- Claim: The function

$$\mathcal{L}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \tag{40}$$

  is convex in the weight vector $\mathbf{w}$

- Proof: $\mathcal{L}$ is obtained through simple convexity preserving operations:
  1. Positive combinations of convex functions is convex

Convexity of the loss function L

- Claim: The function

$$\mathcal{L}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \tag{40}$$

  is convex in the weight vector $\mathbf{w}$

- Proof: $\mathcal{L}$ is obtained through simple convexity preserving operations:
  1. Positive combinations of convex functions is convex
  2. Composition of a convex and a linear functions is convex

Convexity of the loss function L

- Claim: The function

$$\mathcal{L}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \tag{40}$$

  is convex in the weight vector $\mathbf{w}$

- Proof: $\mathcal{L}$ is obtained through simple convexity preserving operations:
  1. Positive combinations of convex functions is convex
  2. Composition of a convex and a linear functions is convex
  3. A linear function is both convex and concave

Convexity of the loss function L

- Claim: The function

$$\mathcal{L}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} -y_n\mathbf{x}_n^\top\mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top\mathbf{w}}) \tag{40}$$

  is convex in the weight vector $\mathbf{w}$

- Proof: $\mathcal{L}$ is obtained through simple convexity preserving operations:
  1. Positive combinations of convex functions is convex
  2. Composition of a convex and a linear functions is convex
  3. A linear function is both convex and concave
  4. $\eta \to \log(1 + e^{\eta})$ is convex (exercise)

Proof of the convexity of $\mathcal{L}$

- Proof of "$\eta \to \log(1 + e^{\eta})$ is convex"

Proof of the convexity of $\mathcal{L}$

- Proof of "$\eta \to \log(1+e^\eta)$ is convex"

$$h(\eta) = \log(1+e^\eta) \tag{41}$$

$$h'(\eta) = \frac{e^\eta}{1+e^\eta} = \sigma(\eta) \tag{42}$$

$$h''(\eta) = \sigma'(\eta) = \frac{e^\eta}{(1+e^\eta)^2} \geq 0 \tag{43}$$

Proof of the convexity of $\mathcal{L}$

- Proof of "$\eta \to \log(1 + e^{\eta})$ is convex"

$$h(\eta) = \log(1 + e^{\eta}) \tag{41}$$

$$h'(\eta) = \frac{e^{\eta}}{1 + e^{\eta}} = \sigma(\eta) \tag{42}$$

$$h''(\eta) = \sigma'(\eta) = \frac{e^{\eta}}{(1 + e^{\eta})^2} \geq 0 \tag{43}$$

- With the property of "composition of a convex and a linear functions is convex":

Proof of the convexity of $\mathcal{L}$

- Proof of "$\eta \to \log(1 + e^\eta)$ is convex"

$$h(\eta) = \log(1 + e^\eta) \tag{41}$$

$$h'(\eta) = \frac{e^\eta}{1 + e^\eta} = \sigma(\eta) \tag{42}$$

$$h''(\eta) = \sigma'(\eta) = \frac{e^\eta}{(1 + e^\eta)^2} \geq 0 \tag{43}$$

- With the property of "composition of a convex and a linear functions is convex":

$\Longrightarrow \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}})$ is convex

Proof of the convexity of $\mathcal{L}$

- Proof of "$\eta \to \log(1+e^{\eta})$ is convex"

$$h(\eta) = \log(1+e^{\eta}) \tag{41}$$

$$h'(\eta) = \frac{e^{\eta}}{1+e^{\eta}} = \sigma(\eta) \tag{42}$$

$$h''(\eta) = \sigma'(\eta) = \frac{e^{\eta}}{(1+e^{\eta})^2} \geq 0 \tag{43}$$

- With the property of "composition of a convex and a linear functions is convex":

$$\Longrightarrow \log(1+e^{\mathbf{x}_n^\top \mathbf{w}}) \text{ is convex}$$

- With the property of "A linear function is both convex and concave"

Proof of the convexity of $\mathcal{L}$

- Proof of "$\eta \to \log(1+e^{\eta})$ is convex"

$$h(\eta) = \log(1+e^{\eta}) \tag{41}$$

$$h'(\eta) = \frac{e^{\eta}}{1+e^{\eta}} = \sigma(\eta) \tag{42}$$

$$h''(\eta) = \sigma'(\eta) = \frac{e^{\eta}}{(1+e^{\eta})^2} \geq 0 \tag{43}$$

- With the property of "composition of a convex and a linear functions is convex":

$\Longrightarrow \log(1+e^{\mathbf{x}_n^\top \mathbf{w}})$ is convex

- With the property of "A linear function is both convex and concave"

$\Longrightarrow -y_n \mathbf{x}_n^\top \mathbf{w}$ is convex

Proof of the convexity of $\mathcal{L}$

- Proof of "$\eta \to \log(1+e^{\eta})$ is convex"

$$h(\eta) = \log(1+e^{\eta}) \tag{41}$$

$$h'(\eta) = \frac{e^{\eta}}{1+e^{\eta}} = \sigma(\eta) \tag{42}$$

$$h''(\eta) = \sigma'(\eta) = \frac{e^{\eta}}{(1+e^{\eta})^2} \geq 0 \tag{43}$$

- With the property of "composition of a convex and a linear functions is convex":

$$\Longrightarrow \log(1+e^{\mathbf{x}_n^\top \mathbf{w}}) \text{ is convex}$$

- With the property of "A linear function is both convex and concave"

$$\Longrightarrow -y_n \mathbf{x}_n^\top \mathbf{w} \text{ is convex}$$

- Positive combinations of convex functions is convex

Proof of the convexity of $\mathcal{L}$

- Proof of "$\eta \to \log(1+e^\eta)$ is convex"

$$h(\eta) = \log(1+e^\eta) \tag{41}$$

$$h'(\eta) = \frac{e^\eta}{1+e^\eta} = \sigma(\eta) \tag{42}$$

$$h''(\eta) = \sigma'(\eta) = \frac{e^\eta}{(1+e^\eta)^2} \geq 0 \tag{43}$$

- With the property of "composition of a convex and a linear functions is convex":

$$\implies \log(1+e^{\mathbf{x}_n^\top \mathbf{w}}) \text{ is convex}$$

- With the property of "A linear function is both convex and concave"

$$\implies -y_n \mathbf{x}_n^\top \mathbf{w} \text{ is convex}$$

- Positive combinations of convex functions is convex

$$\mathcal{L}(\mathbf{w}) = \tfrac{1}{N} \textstyle\sum_{n=1}^N -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1+e^{\mathbf{x}_n^\top \mathbf{w}}) \text{ is convex}$$

Second proof: Hessian of $\mathcal{L}$ is psd

The Hessian $\nabla^2\mathcal{L}$ is the **matrix** whose entries are the **second derivatives** $\frac{\partial^2}{\partial w_i \partial w_j}\mathcal{L}(\mathbf{w})$.

Second proof: Hessian of $\mathcal{L}$ is psd

The Hessian $\nabla^2 \mathcal{L}$ is the **matrix** whose entries are the **second derivatives** $\frac{\partial^2}{\partial w_i \partial w_j} \mathcal{L}(\mathbf{w})$.

$$
\begin{aligned}
\nabla^2 \mathcal{L}(\mathbf{w}) &= \nabla \left[\nabla \mathcal{L}(\mathbf{w})\right]^\top \\
&= \nabla \left[\frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) \right]^\top \\
&= \frac{1}{N} \sum_{n=1}^{N} \nabla \sigma(\mathbf{x}_n^\top \mathbf{w}) \mathbf{x}_n^\top = \frac{1}{N} \sum_{n=1}^{N} \sigma(\mathbf{x}_n^\top \mathbf{w}) \left(1 - \sigma(\mathbf{x}_n^\top \mathbf{w})\right) \mathbf{x}_n \mathbf{x}_n^\top
\end{aligned}
$$

Second proof: Hessian of $\mathcal{L}$ is psd

The Hessian $\nabla^2\mathcal{L}$ is the **matrix** whose entries are the **second derivatives** $\frac{\partial^2}{\partial w_i \partial w_j}\mathcal{L}(\mathbf{w})$.

$$
\begin{aligned}
\nabla^2\mathcal{L}(\mathbf{w}) &= \nabla\left[\nabla\mathcal{L}(\mathbf{w})\right]^\top \\
&= \nabla\left[\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\left(\sigma(\mathbf{x}_n^\top\mathbf{w}) - y_n\right)\right]^\top \\
&= \frac{1}{N}\sum_{n=1}^{N}\nabla\sigma(\mathbf{x}_n^\top\mathbf{w})\mathbf{x}_n^\top = \frac{1}{N}\sum_{n=1}^{N}\sigma(\mathbf{x}_n^\top\mathbf{w})\left(1-\sigma(\mathbf{x}_n^\top\mathbf{w})\right)\mathbf{x}_n\mathbf{x}_n^\top
\end{aligned}
$$

It can be written under the matrix form:

$$
\nabla^2\mathcal{L}(\mathbf{w}) = \frac{1}{N}\mathbf{X}^\top\mathbf{S}\mathbf{X}\,, \qquad \text{where } \mathbf{S} = \operatorname{diag}\left[\sigma(\mathbf{x}_n^\top\mathbf{w})\left(1-\sigma(\mathbf{x}_n^\top\mathbf{w})\right)\right] \succeq 0 \tag{44}
$$

Second proof: Hessian of $\mathcal{L}$ is psd

The Hessian $\nabla^2\mathcal{L}$ is the **matrix** whose entries are the **second derivatives** $\frac{\partial^2}{\partial w_i \partial w_j}\mathcal{L}(\mathbf{w})$.

$$\begin{aligned}
\nabla^2\mathcal{L}(\mathbf{w}) &= \nabla\left[\nabla\mathcal{L}(\mathbf{w})\right]^\top \\
&= \nabla\left[\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\left(\sigma(\mathbf{x}_n^\top\mathbf{w}) - y_n\right)\right]^\top \\
&= \frac{1}{N}\sum_{n=1}^{N}\nabla\sigma(\mathbf{x}_n^\top\mathbf{w})\mathbf{x}_n^\top = \frac{1}{N}\sum_{n=1}^{N}\sigma(\mathbf{x}_n^\top\mathbf{w})\left(1-\sigma(\mathbf{x}_n^\top\mathbf{w})\right)\mathbf{x}_n\mathbf{x}_n^\top
\end{aligned}$$

It can be written under the matrix form:

$$\nabla^2\mathcal{L}(\mathbf{w}) = \frac{1}{N}\mathbf{X}^\top\mathbf{S}\mathbf{X}, \qquad \text{where } \mathbf{S} = \operatorname{diag}\left[\sigma(\mathbf{x}_n^\top\mathbf{w})\left(1-\sigma(\mathbf{x}_n^\top\mathbf{w})\right)\right] \succeq 0 \tag{44}$$

$\Longrightarrow \mathcal{L}$ is convex since $\nabla^2\mathcal{L}(\mathbf{w}) \succeq 0$.