Classification (part 1)

Logistic regression

Multiple linear regression

In a regression model, we're interested in predicting a quantitative variable (i.e. $y \in \mathbb{R}$):

$$\begin{aligned}\hat{y} &= h(\mathbf{x}; \hat{\boldsymbol{\theta}}) \\ &= \mathbf{x}^\top \boldsymbol{\theta} \triangleq \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p\end{aligned}$$

Classification

In classification, we are instead interested in predicting some **categorical** variable

- binary classification: $y \in \{0,1\}$ (or $y \in \{-1, +1\}$)
- multi–class: $y \in \{1,2,\ldots,K\}$ for $K \geq 2$
- multi–label: $\mathbf{y} = \left[y_1, y_2, \ldots, y_q\right]^\top$ with $y_j \in \{1,2,\ldots,K\}$ for $j = 1,\ldots,q$

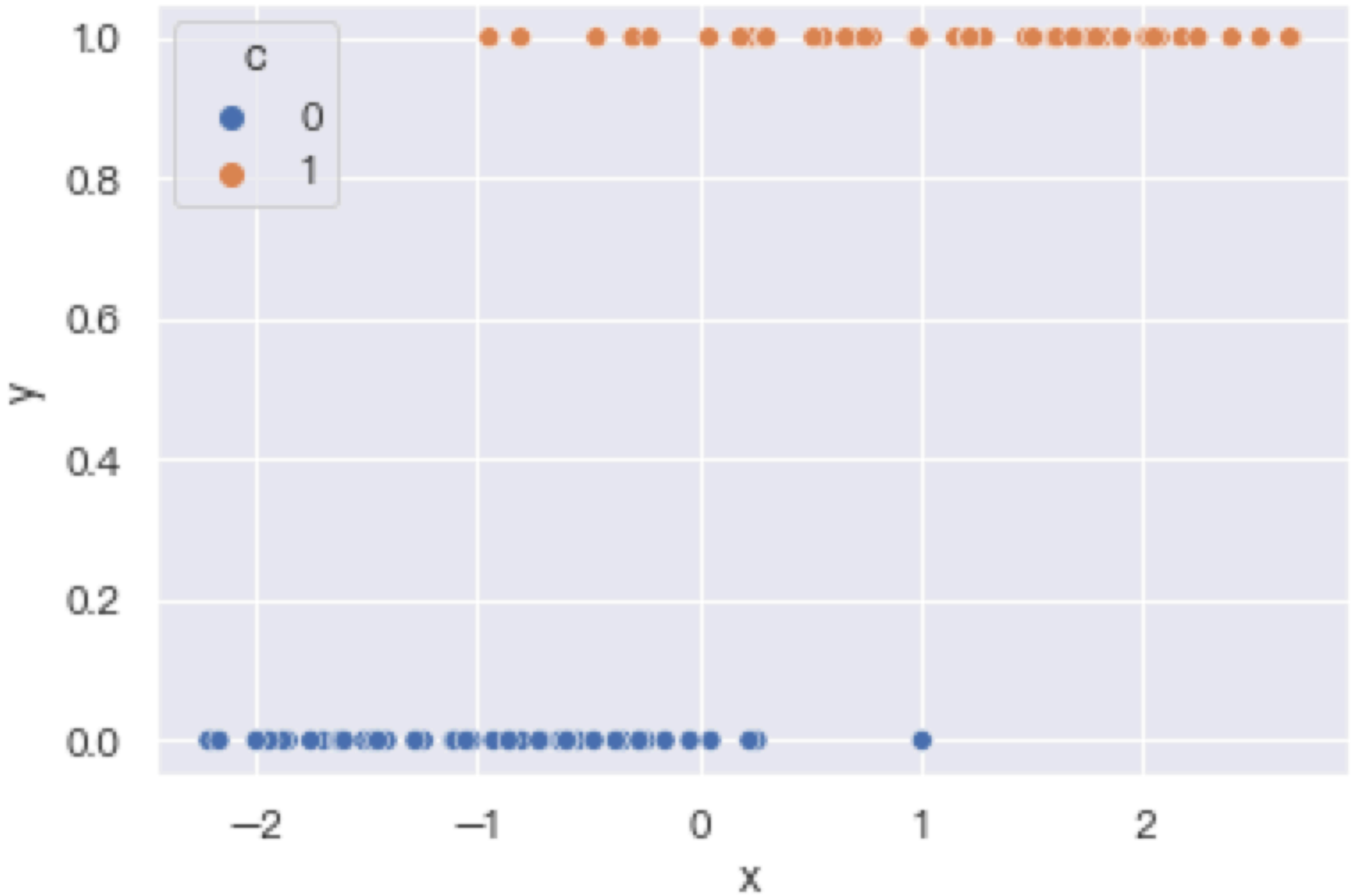
Agenda

- Logistic function
- Cross-entropy loss
- Classification by logistic regression
- Metrics for evaluating logistic regression models

Logistic function

Toy example

```
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=100, n_features=1,
                           n_informative=1, n_clusters_per_class=1,
                           n_redundant=0, flip_y=0)
dat = pd.DataFrame({'x':X[:,0], 'y':y, 'c':y})
sns.scatterplot(data=dat, x='x', y='y', hue='c')
plt.show()
```



Why not use MLR?

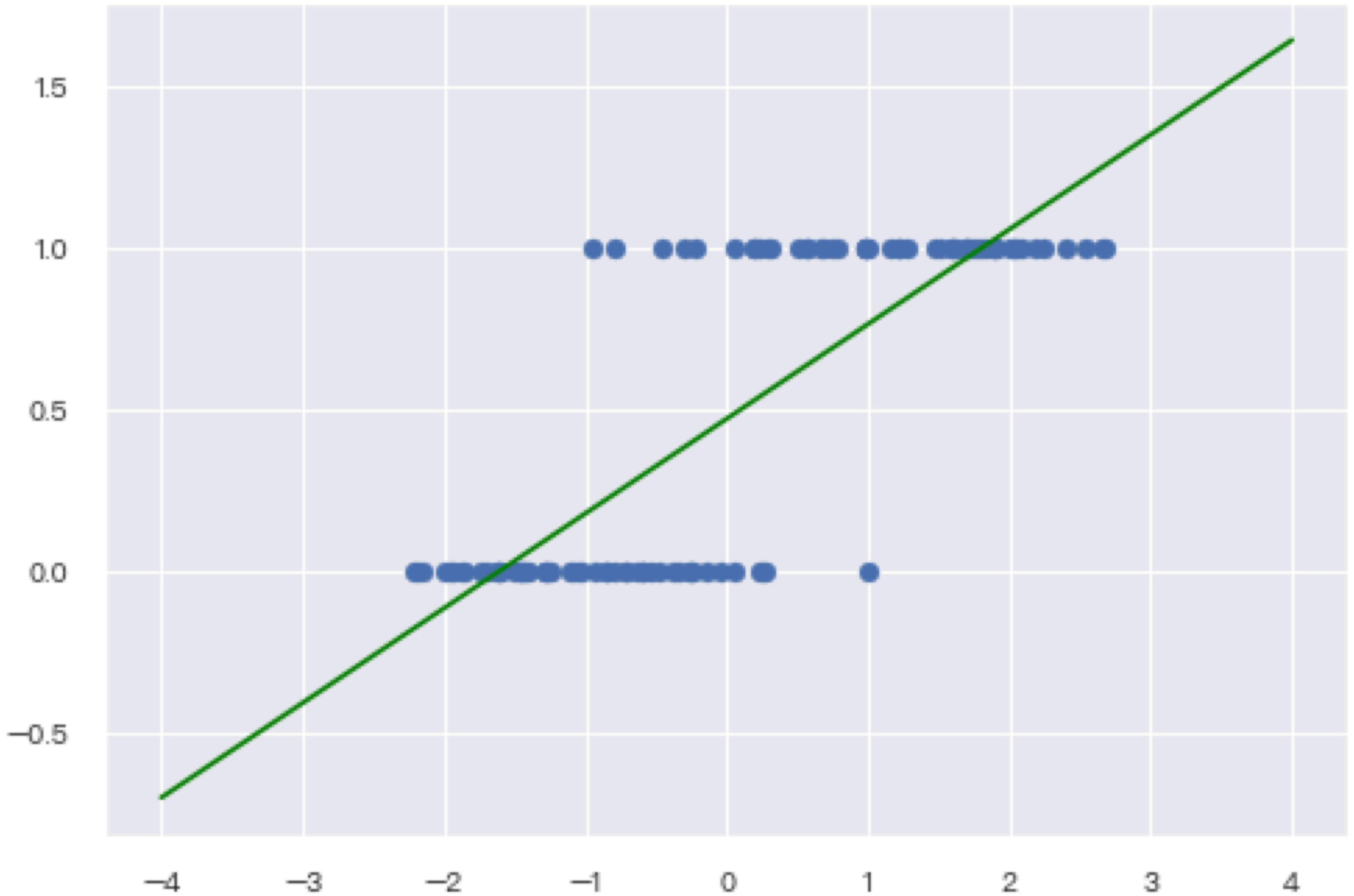
We already have a model that can predict any quantitative response. Why not use it here?

- The output can be outside of the range $[0,1]$. What does $\hat{y} = 2$ mean?
- The response y does not follow normal distribution nor contain measurement error
- Sensitive to outliers

3930

```python
import numpy as np
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X, y)
xeval = np.linspace(-4,4,201).reshape(201,1)
ypred = reg.predict(xeval)
```

```python
plt.style.use('seaborn-notebook')
plt.scatter(X, y)
plt.plot(xeval, ypred, c='green')
plt.show()
```



Bernoulli model

- Since the true Y is either 0 or 1, we need to make sure $h(\mathbf{X};\boldsymbol{\theta}) = 0$ or 1.

- Our idea is to model the conditional probability $p(\mathbf{x}) = P(Y = 1 \mid \mathbf{X} = \mathbf{x})$ and determine the output of $h(\mathbf{X};\boldsymbol{\theta})$ by the value of $p(\mathbf{x})$.

Logistic function

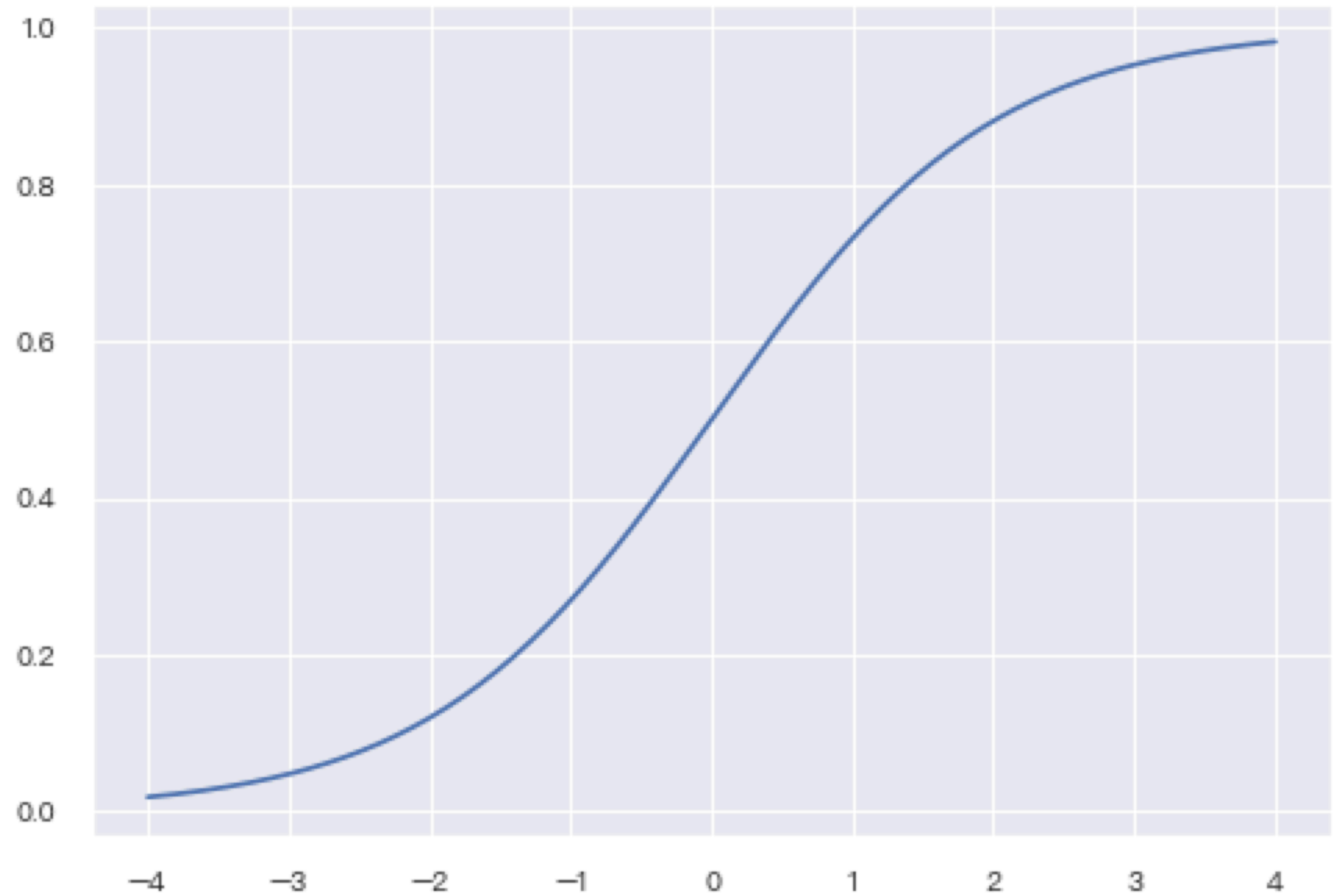
- Note that $p(\mathbf{x})$ is a probability, and thus $0 \leq p(\mathbf{x}) \leq 1$.
- One way to achieve this constraint is trough the logistic function:

$$\sigma(t) = \frac{1}{1 + e^{-t}}.$$

- Here t is a transformation of the features $\mathbf{x}$; e.g., $t = \mathbf{x}^\top \boldsymbol{\theta}$ or an output of a deep neural network.
- The function $\sigma(t)$ is also called **sigmoid** function.

```python
def sigmoid(t):
    return 1/(1+np.exp(-t))
```

```python
plt.plot(xeval,sigmoid(xeval))
plt.show()
```



Properties of the logistic function

- Definition:

$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t}$$

- Range: $0 < \sigma(t) < 1$

- Reflection and symmetry:

$$1 - \sigma(t) = \frac{1}{1+e^t} = \frac{e^{-t}}{1+e^{-t}} = \sigma(-t)$$

- Derivative:

$$\frac{d}{dt}\sigma(t) = \sigma(t)\big(1 - \sigma(t)\big) = \sigma(t)\sigma(-t)$$

quotient rule

- Inverse:

$$t = \sigma^{-1}(p) = \log\left(\frac{p}{1-p}\right)$$

Cross-entropy loss

Logistic regression with squared loss

We might estimate $\boldsymbol{\theta}$ with squared-error loss, which yields the following empirical risk:

$$R\left(\boldsymbol{\theta}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma\left(\mathbf{x}_i^\top\boldsymbol{\theta}\right)\right)^2,$$

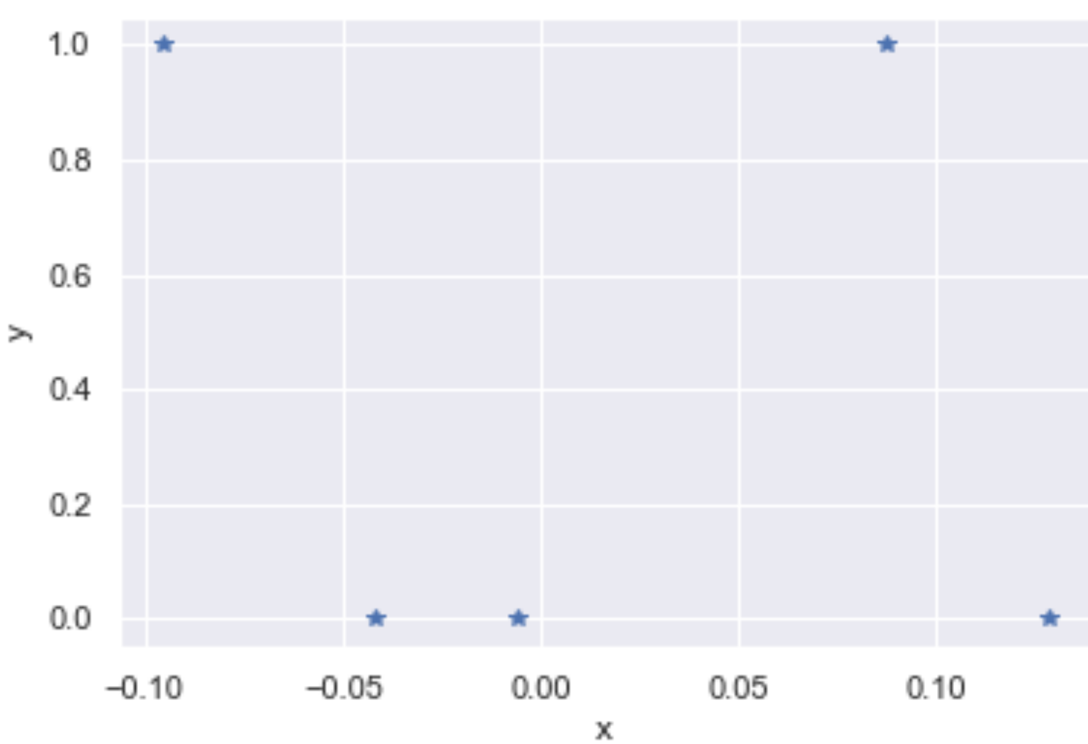
or the structured risk:

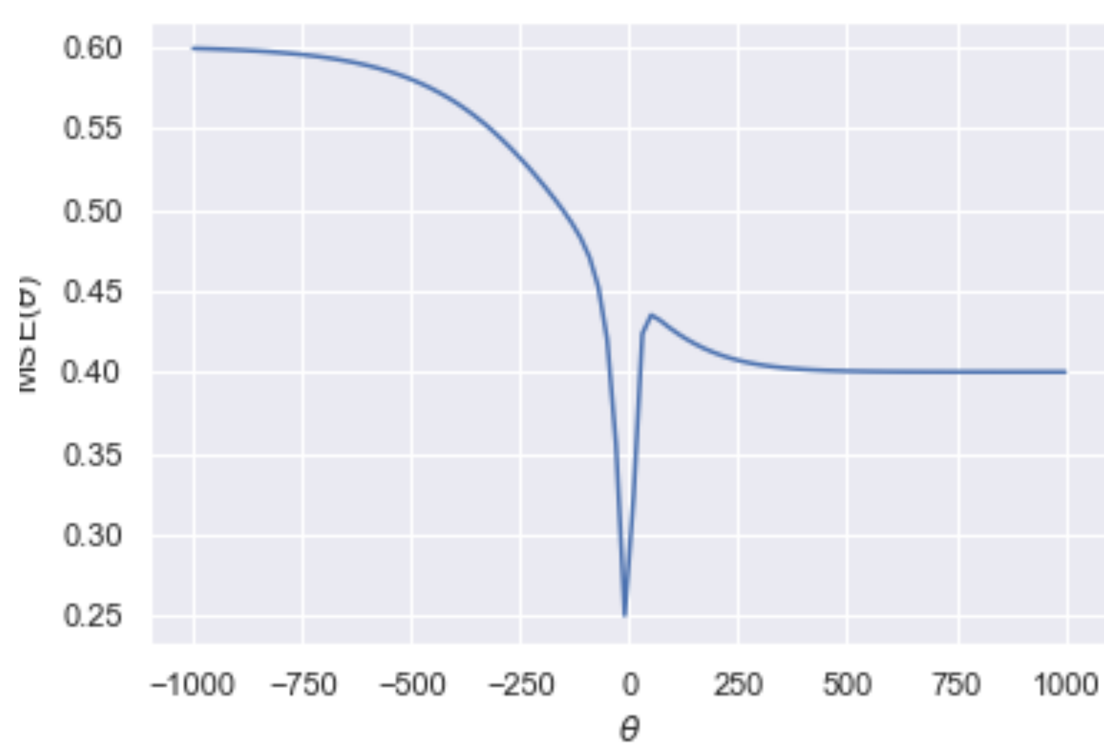
$$R\left(\boldsymbol{\theta}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma\left(\mathbf{x}_i^\top\boldsymbol{\theta}\right)\right)^2 + \lambda\|\boldsymbol{\theta}\|.$$

Pitfalls of squared loss with logistic regression

1. In logistic regression, squared–error loss may not be convex.

2. Squared–error loss is not well–defined since it is not of the form $L\left(y, \hat{y}\right)$.





Likelihood function

Let $X_1, X_2, \ldots, X_n$ be a random sample. The likelihood function is given by

$$L(\boldsymbol{\theta} \mid X_1, \ldots, X_n) \triangleq f(X_1, \ldots, X_n; \boldsymbol{\theta}) \, .$$

When $X_1, X_2, \ldots, X_n \overset{iid}{\sim} f(x \mid \boldsymbol{\theta})$, the above equation becomes

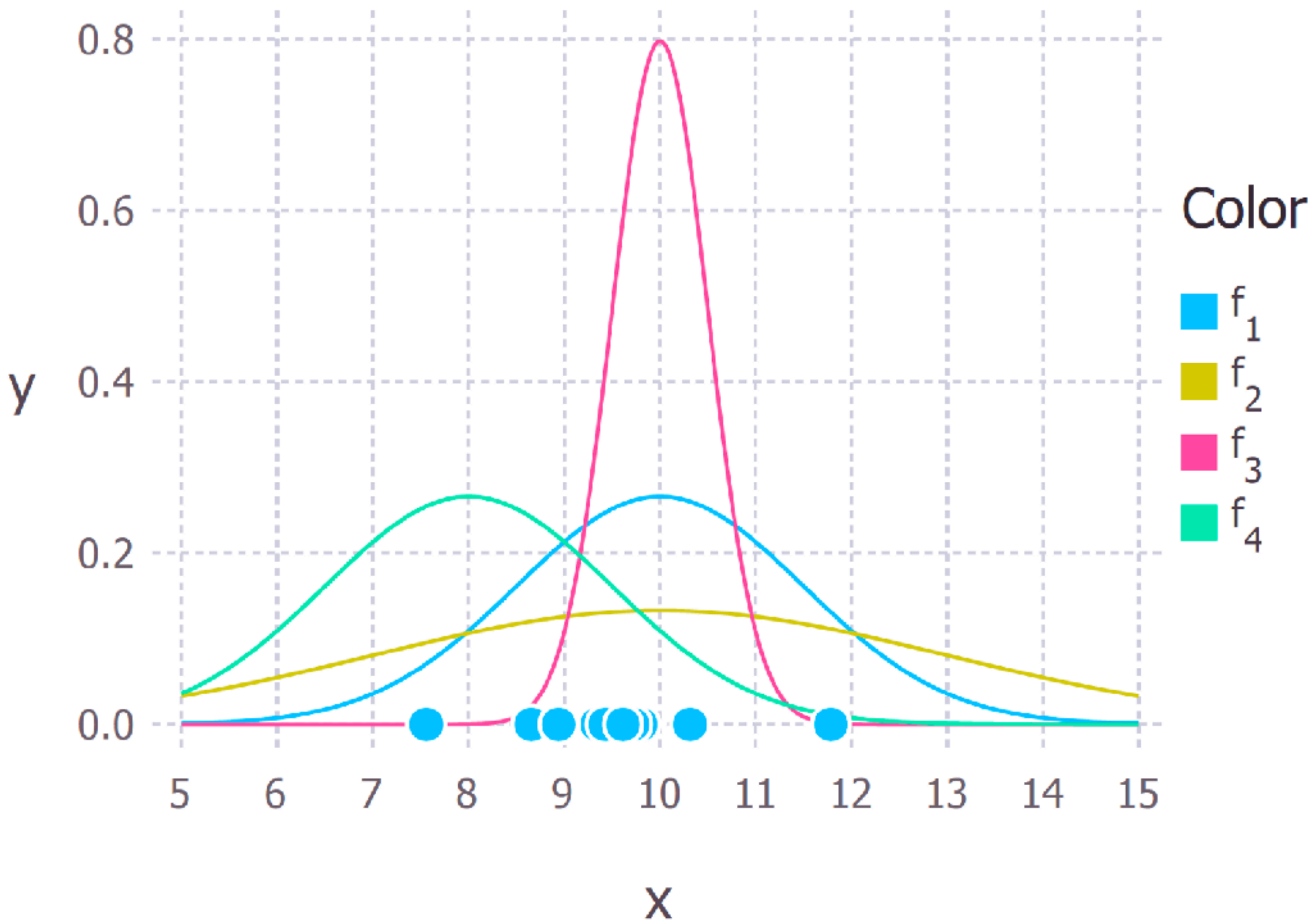
$$\boxed{L(\boldsymbol{\theta} \mid X_1, \ldots, X_n) = \prod_{i=1}^{n} f(X_i; \boldsymbol{\theta}) \, .}$$

easier to compute

Maximum likelihood estimation

- Find the parameter that is “most likely” to observe your data.

- Maximizing the likelihood function is equivalent to maximizing the log-likelihood function (for computational issues) since logarithm is a monotone function.



https://towardsdatascience.com/probability-concepts-explained-maximum-likelihood-estimation-c7b4342fdbb1

Log–likelihood function

Maximizing the likelihood function is difficult (since lots of products is involved), but maximizing the log–likelihood is much easier:

$$\begin{aligned} \ell(\boldsymbol{\theta} \,|\, X_1, \ldots, X_n) &\triangleq \log L(\boldsymbol{\theta} \,|\, X_1, \ldots, X_n) \\ &= \sum_{i=1}^{n} \log f(X_i; \boldsymbol{\theta}) \,. \end{aligned}$$

MLE for logistic regression

Assume $Y_i | \mathbf{X}_i = \mathbf{x}_i$ are independent sample from Bernoulli $\left(\sigma\left(\mathbf{x}_i^\top \boldsymbol{\theta}\right)\right)$. The likelihood function becomes

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} \sigma\left(\mathbf{x}_i^\top \boldsymbol{\theta}\right)^{y_i} \left[1 - \sigma\left(\mathbf{x}_i^\top \boldsymbol{\theta}\right)\right]^{(1-y_i)},$$

MLE.pdf第12頁、
MLE.mp4第41:27

and the log-likelihood function:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} y_i \cdot \log\left(\sigma\left(\mathbf{x}_i^\top \boldsymbol{\theta}\right)\right) + (1 - y_i) \cdot \log\left(1 - \sigma\left(\mathbf{x}_i^\top \boldsymbol{\theta}\right)\right).$$

Cross-entropy loss

Maximizing $\ell(\boldsymbol{\theta})$ is equivalent to minimizing $-\ell(\boldsymbol{\theta})$:

$$-\ell(\boldsymbol{\theta}) = \boxed{\frac{1}{n}} \sum_{i=1}^{n} \boxed{-\left[y_i \cdot \log\left(\sigma\left(\mathbf{x}_i^\top \boldsymbol{\theta}\right)\right) + (1 - y_i) \cdot \log\left(1 - \sigma\left(\mathbf{x}_i^\top \boldsymbol{\theta}\right)\right)\right]}$$

does not affect minimization

cross-entropy loss

Classification by logistic regression

Binary classification by logistic regression

- Let $\hat{\boldsymbol{\theta}}$ be an estimate of $\boldsymbol{\theta}$, then for a new observation $\mathbf{X} = \mathbf{x}$ we have

$$\hat{P}\left(Y = 1 \,|\, \mathbf{X} = \mathbf{x}\right) = \hat{p}(\mathbf{x}) = \sigma\left(\mathbf{x}^{\top}\hat{\boldsymbol{\theta}}\right)$$

- We have to make predictions based on $\hat{p}(\mathbf{x})$, for example,

$$\hat{h}(\mathbf{x}) = \begin{cases} 1, & \text{if } \hat{p}(\mathbf{x}) \geq 1/2 \\ 0, & \text{if } \hat{p}(\mathbf{x}) < 1/2 \end{cases}$$

Classification rule

- The function $h(\mathbf{x})$ is often referred as a classification rule

- More generally, a classification rule can be written as

$$h(\mathbf{x}) = \begin{cases} 1, & \text{if } \hat{p}(\mathbf{x}) \geq \delta \\ 0, & \text{if } \hat{p}(\mathbf{x}) < \delta \end{cases}$$

with $0 \leq \delta \leq 1$ is a classification threshold

Decision cutoff

- In decision theory, δ can be determined by minimizing the risk function with a specified loss function

- For example, If the 0–1 loss is used, the conditional risk function becomes

$$R_{Y|\mathbf{X}=\mathbf{x}}\left(h(\mathbf{x})\right) = \begin{cases} p(\mathbf{x}) & \text{if } y = 1 \text{ and } d(\mathbf{x}) = 0 \\ 1 - p(\mathbf{x}) & \text{otherwise} \end{cases}$$

and it can be shown that $R_{Y|\mathbf{X}=\mathbf{x}}$ is minimized by $\delta = 1/2$

Decision cutoff (cont'd)

- In computational learning, we may need to determine δ by optimizing some other **metrics**

Evaluating binary classifiers

From Data 100, Fall 2020 @ UC Berkeley

Summary

- Logistic regression is derived from Bernoulli model
  - logistic (sigmoid function)
  - cross-entropy loss
- Make predictions by a decision rule (threshold)
- Metrics to evaluate a logistic regression model
  - accuracy, precision, recall
  - ROC curves, AUC

Readings

- Lecture 18 and 19 of Berkeley's Data 100
- Chapter 17 of Principles and Techniques of Data Science
- Chapter 16 of Data Science from Scratch: First Principles with Python

Homework: binary logistic regression

Fit a logistic regression model (feature engineering is welcome) to the breast cancer wisconsin dataset by sklearn.linear_model.LogisticRegression. Evaluate your model by a repeated stratified 10-fold cross validation.