Supervised Learning I

Yongjin Park
University of British Columbia
(with Keegan Korthauer)

01 March, 2022

Learning Objectives

- Introduction to Classification Problems
- A crash course on supervised learning methods
  - Fundamentls: Machine learning is a statistical inference problem
- Applications to real-world biological problems

Today's lecture

General discussions on supervised learning

Classification with a decision rule

Generalized linear model

Generative Modelling approach

Other Methods

What is Supervised Learning?

Supervised learning (today)

- Input: data $\mathcal{X} = \{\mathbf{x}_i, ...\}$
- Input 2: label $\mathcal{Y} = \{y_i, ...\}$
- Goal: learn $f : \mathcal{X} \to \mathcal{Y}$

X: gene expression samples; Y: disease labels

Unsupervised learning (previous)

- Input: data $\mathcal{X} = \{\mathbf{x}_i, ...\}$
- Goal 1: learn $f : \mathcal{Z} \to \mathcal{X}$
- Goal 2: hidden/latent states, $\mathcal{Z} = \{\mathbf{z}_i, ...\}$

X: gene expression matrices

Machine learning vs. Classical statistics

Machine Learning

- *"Statistical learning with the help of algorithm"*
- Large number of variables (less scrutiny on variable selection)
- Higher model complexity, non-linearity
- Focusing on **classification/prediction**
- Scalability, computation

Classical Statistics

- A handful of variables with clear
- Linear models, or generalized linear models
- Data generating process (Bayesian)
- More theory-oriented research
- More focuses on **scrutiny** (type-I-error, FDR, etc) and **variable selection**

Note: We take a bit more statistical stance.

Four Steps for Supervised Learning

1. Gather some training data (this is a part of research!):

$$\begin{aligned} \mathcal{X} &\equiv \{\mathbf{x}_i : i \in [n]\} \quad \text{predictors/covariates/features} \\ \mathcal{Y} &\equiv \{\mathbf{y}_i : i \in [n]\} \quad \text{outcome/response} \end{aligned}$$

2. Write down a model (classifier) $f : \mathcal{X} \to \mathcal{Y}$
   - What is the loss function?
3. Fit the model to the training data
4. Use the model
   - We can focus on the interpretation of model parameters
   - We can focus on prediction on *new* data points

Why do we do supervised learning?

Don't we already know the labels?

Today's lecture

General discussions on supervised learning

Classification with a decision rule

Generalized linear model

Generative Modelling approach

Other Methods

Both regression and classification is supervised learning

Multivariate Linear Regression

- X: $n \times p$ design matrix
- $\mathbf{y}$: $n \times 1$ outcome vector
- β_k: a regression coefficient for a covariate k
- ϵ: a vector of residual errors
- A regression model:

$$Y_i = \sum_k X_{ik}\beta_k + \epsilon_i$$

(or $\mathbf{y} = X\beta + \epsilon$)

Both regression and classification is supervised learning

Multivariate Linear Regression

- X: $n \times p$ design matrix
- $\mathbf{y}$: $n \times 1$ outcome vector
- β_k: a regression coefficient for a covariate k
- ϵ: a vector of residual errors
- A regression model:

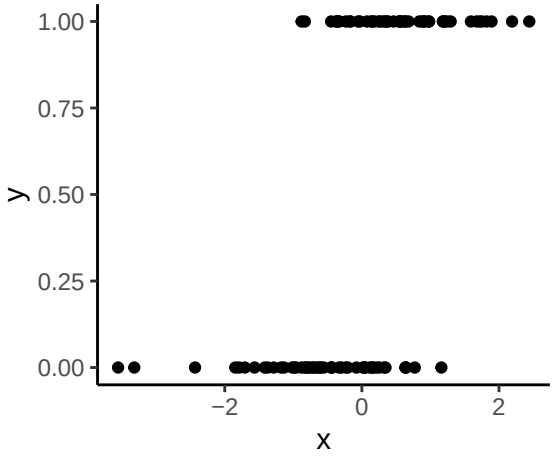
$$\mathbf{y} = X\beta + \epsilon$$

(A linear) Classifier

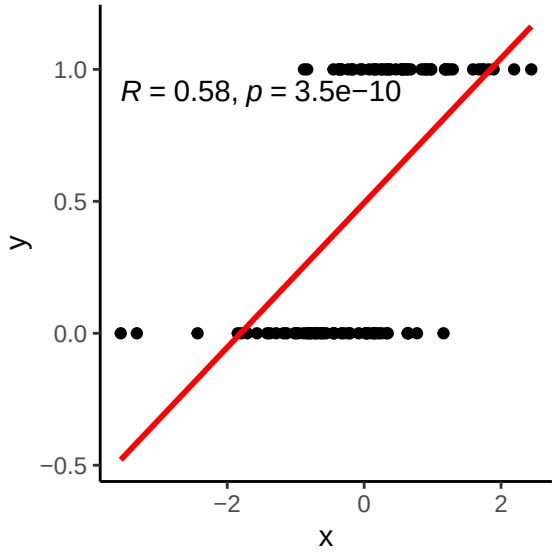
- X: $n \times p$ design matrix
- $\mathbf{y}$: $n \times 1$ **class label** vector, e.g., $\{0, 1\}^n$
- β_k: a coefficient for a covariate k
- A prediction model:

$$Y_i \leftarrow f\left(\sum_k X_{ik}\beta_k\right)$$

A primer question: Can we simply use regression for classification?



A primer question: Can we simply use regression for classification?



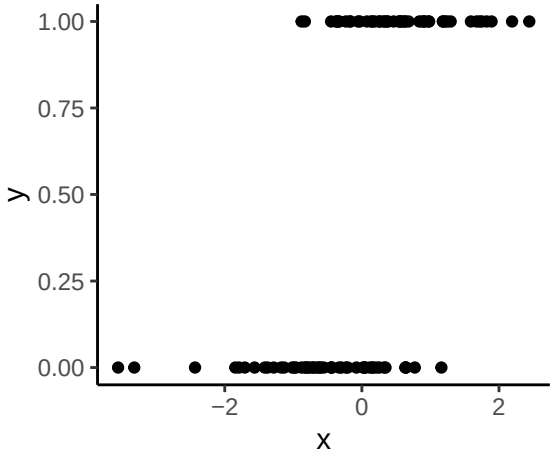
We don't need to cover all the "areas" of Y

```
.glm <- glm(y ~ x, family="binomial")
```

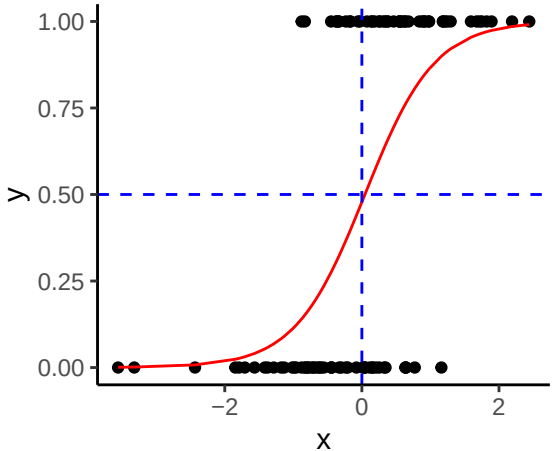
Unlike $X_i \in \mathbb{R}$, we have

$$Y_i \in \{0, 1\}$$

Classification = drawing a decision boundary

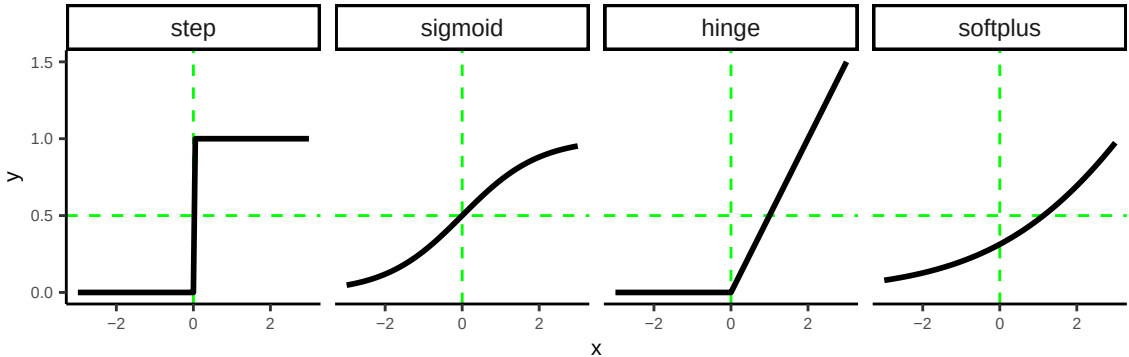


Classification = drawing a decision boundary



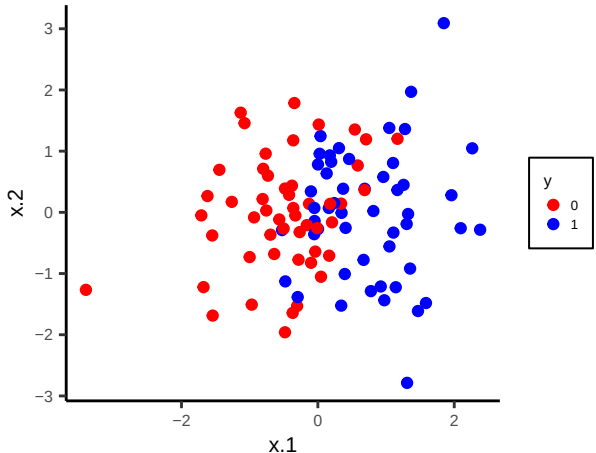
Step-like functions

We will use a step-like function to approximate classification rules



$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}, \; f(x) = \frac{e^x}{1+e^x}, \; f(x) = \max\{0, x\xi\}, \; f(x) = \log(1+\exp(x\alpha - \beta))$$

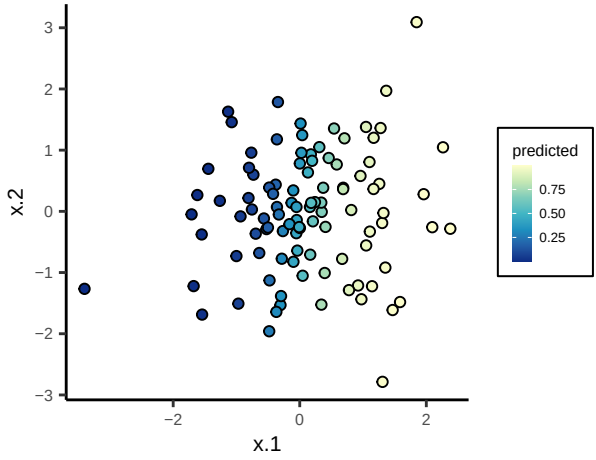
Classification = drawing a decision boundary - 2D



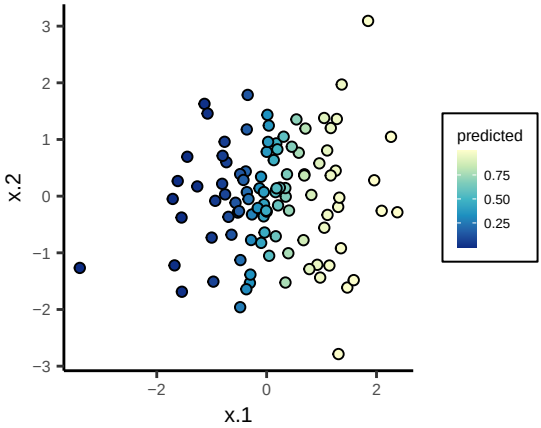
Classification = drawing a decision boundary - 2D

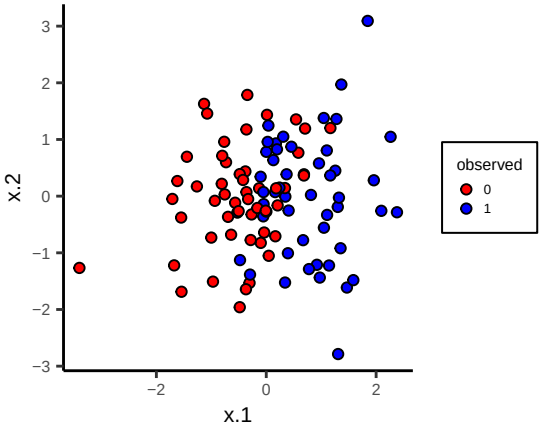
```
.glm <- glm(y ~ x, family = "binomial")
y.hat <-  fitted(.glm)
```

Classification = drawing a decision boundary - 2D



Classification = drawing a decision boundary - 2D





Understanding loss functions (regression vs. classification)

Regression loss

$$L_n = n^{-1} \sum_{i=1}^{n} (\underbrace{y_i}_{\text{observed}} - \underbrace{\mathbf{x}_i \beta}_{\text{predicted}})^2$$

Average distance between the observed and predicted.

Classification loss

Given a classifier spits out class label, e.g., $\{0, 1\}$,

$$L_n = n^{-1} \sum_{i=1}^{n} I \left\{ \underbrace{y_i}_{\text{observed}} \neq \underbrace{f(\mathbf{x}_i \beta)}_{\text{predicted}} \right\}$$

Average frequency of a wrong answer by the classifier.

- **Algorithm:** Minimize the loss function with respect to free parameters β.

Logistic regression: a probabilistic version of classification loss

If a classifier $f(\mathbf{x}_i\beta) \in (0, 1)$ provides a probability of $Y_i = 1$, $\widehat{p}(Y_i|\mathbf{x}_i, \beta)$:

Bernoulli likelihood:

$$\prod_{i=1}^{n} f^{Y_i}(1-f)^{1-Y_i}$$

Logistic regression: a probabilistic version of classification loss

If a classifier $f(\mathbf{x}_i\beta) \in (0, 1)$ provides a probability of $Y_i = 1$, $\hat{p}(Y_i|\mathbf{x}_i, \beta)$:

Bernoulli likelihood:

$$\prod_{i=1}^{n} f^{Y_i}(1-f)^{1-Y_i}$$

where for each data point i

$$\text{likelihood}_i = \begin{cases} f & \text{if } Y_i = 1 \\ (1-f) & \text{if } Y_i = 0 \end{cases}$$

We will use the sigmoid function

$$f(z) = \frac{e^z}{1+e^z}$$

Logistic regression: a probabilistic version of classification loss

If a classifier $f(\mathbf{x}_i\beta) \in (0, 1)$ provides a probability of $Y_i = 1$, $\hat{p}(Y_i|\mathbf{x}_i, \beta)$:

Bernoulli likelihood:

$$\prod_{i=1}^{n} f^{Y_i}(1-f)^{1-Y_i}$$

We can use negative log-likelihood as a "softer" version classification loss

$$L = -\log \sum_{i=1}^{n} \left[f(\mathbf{x}_i\beta)^{Y_i} \left[1 - f(\mathbf{x}_i\beta)\right]^{1-Y_i} \right]$$

Logistic regression: a probabilistic version of classification loss

If a classifier $f(\mathbf{x}_i\beta) \in (0,1)$ provides a probability of $Y_i = 1$, $\widehat{p}(Y_i|\mathbf{x}_i, \beta)$:

Bernoulli likelihood:

$$\prod_{i=1}^{n} f^{Y_i}(1-f)^{1-Y_i}$$

We can use negative log-likelihood as a "softer" version classification loss

what is this?

$$L = \sum_{i=1}^{n} Y_i \log \frac{1 - f(\mathbf{x}_i\beta)}{f(\mathbf{x}_i\beta)} - \log(1 - f(\mathbf{x}_i\beta))$$

What if $f \to 1$?

Understanding loss functions (soft vs. hard classification)

Soft classification loss

Given a classifier f outputs out class label probability $\in (0, 1)$

$$L_n = \sum_{i=1}^{n} Y_i \log \overbrace{\frac{1 - f(\mathbf{x}_i\beta)}{f(\mathbf{x}_i\beta)}}^{\text{mistake odds ratio}} \underbrace{- \log(1 - f(\mathbf{x}_i\beta))}_{\text{will try to predict } f=0}$$

Total "mistake" log-odds ratio

Classification loss

Given a classifier spits out class label, e.g., $\{0, 1\}$,

$$L_n = \frac{1}{n} \sum_{i=1}^{n} I \left\{ \underbrace{y_i}_{\text{observed}} \neq \underbrace{f(\mathbf{x}_i\beta)}_{\text{predicted}} \right\}$$

Average frequency of a wrong answer by the classifier.

Logistic loss function

Minimize this with respect to β:

$$L_n = \sum_{i=1}^{n} \left[-Y_i \overbrace{\sum_{k=1}^{p} X_{ik}\beta_k}^{\text{log-odds}} \underbrace{+ \log\left(1 + e^{\sum_{k=1}^{p} X_{ik}\beta_k}\right)}_{\text{"price" for the positive}} \right]$$

$\Longleftrightarrow$

maximize the likelihood

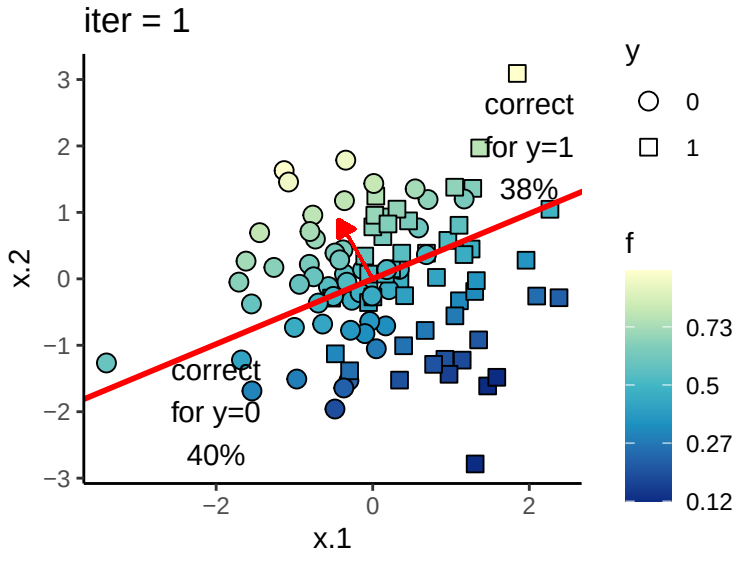
$$\mathcal{L}_n = \sum_{i=1}^{n} \left[Y_i \sum_{k=1}^{p} X_{ik}\beta_k - \log\left(1 + e^{\sum_{k=1}^{p} X_{ik}\beta_k}\right) \right]$$

A classification rule

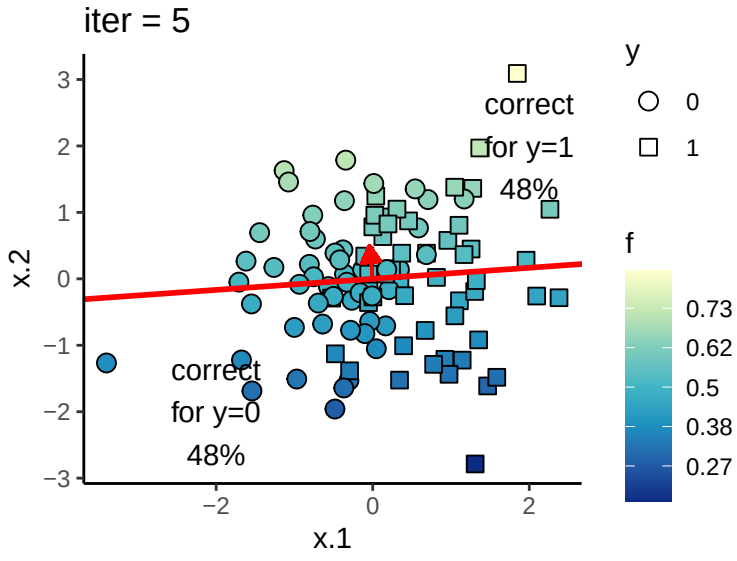
Classifier

$$\hat{Y}_i = \begin{cases} 1 & \sum_{k=1}^{p} X_{ik}\beta_k > 0 \\ 0 & \sum_{k=1}^{p} X_{ik}\beta_k \leq 0 \end{cases}$$

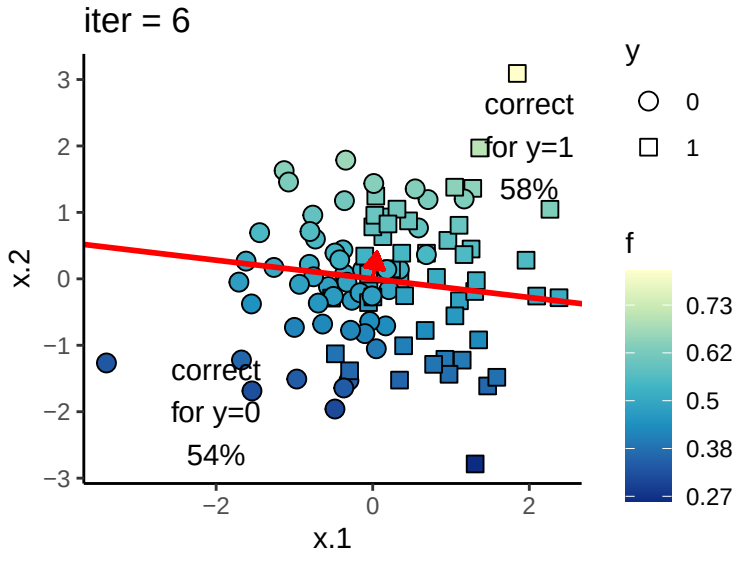
Binary classification in action



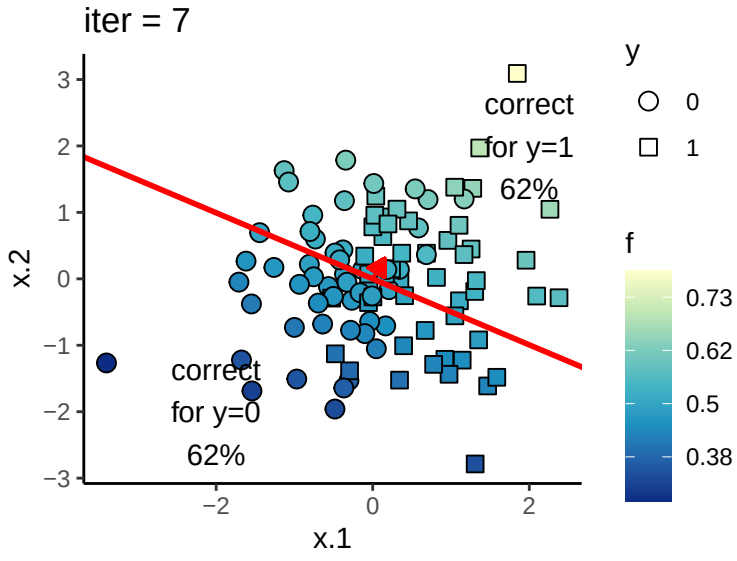
Binary classification in action



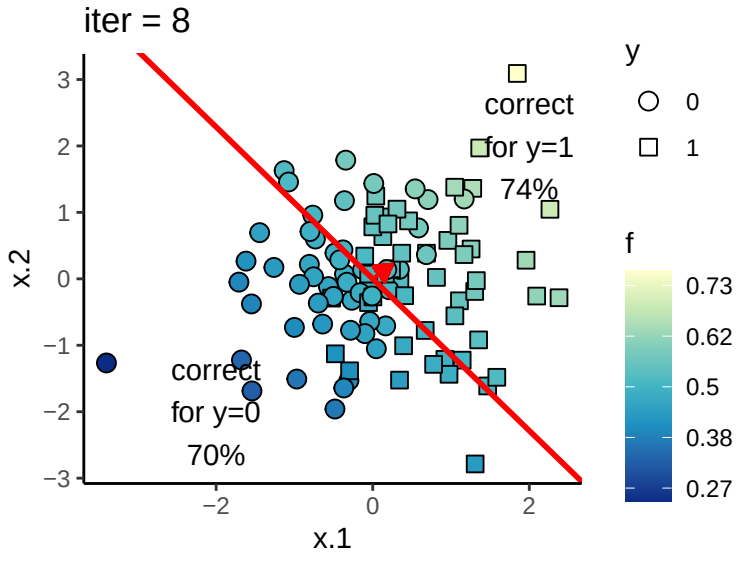
Binary classification in action



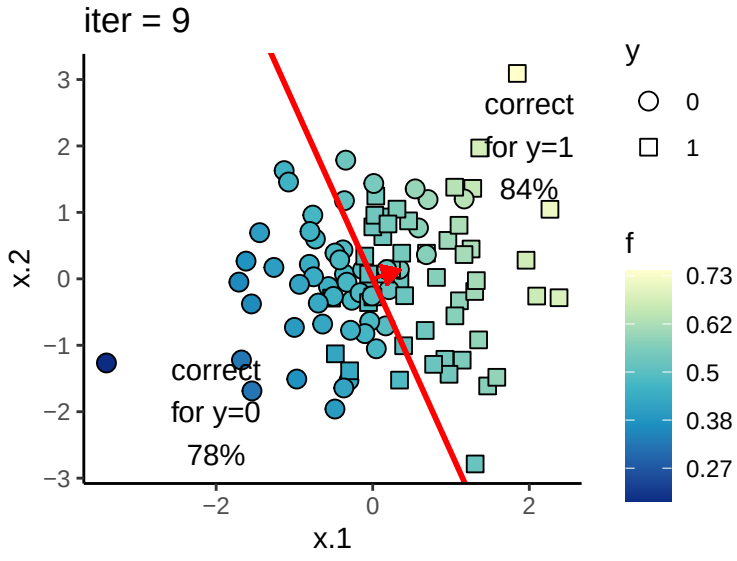
Binary classification in action



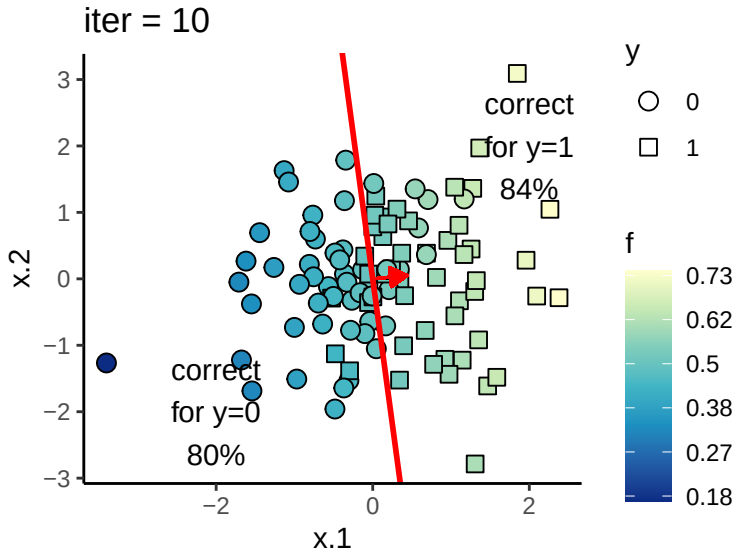
Binary classification in action



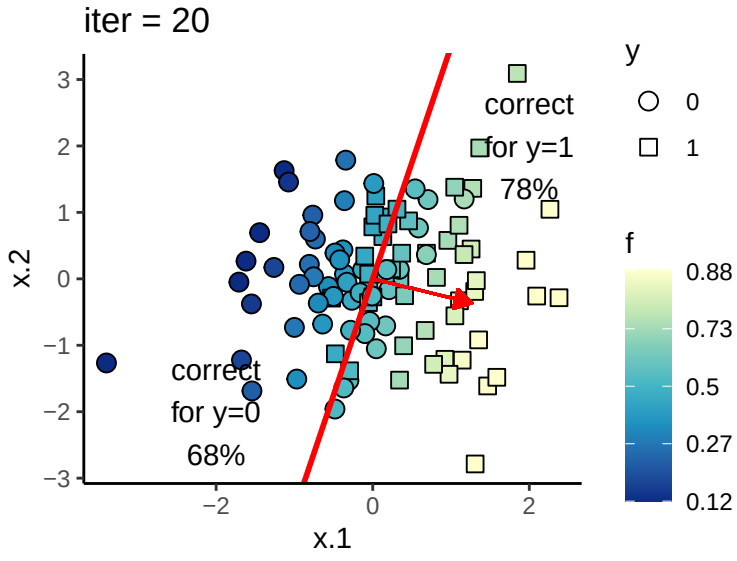
Binary classification in action



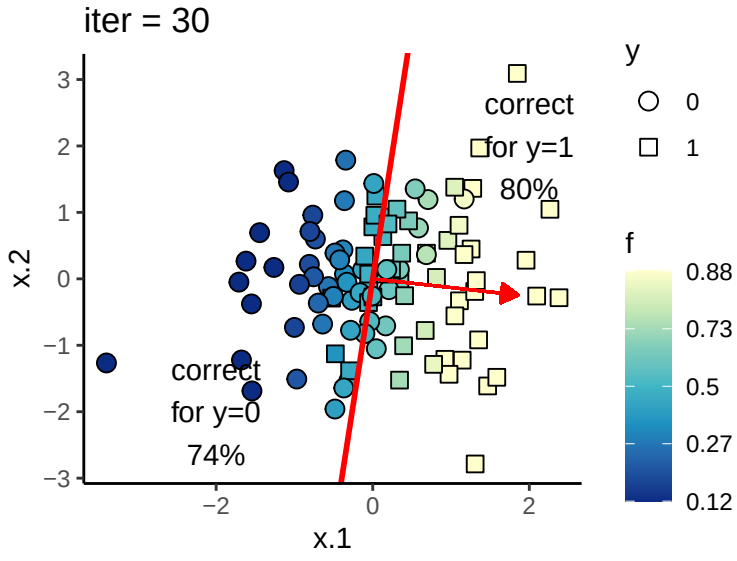
Binary classification in action



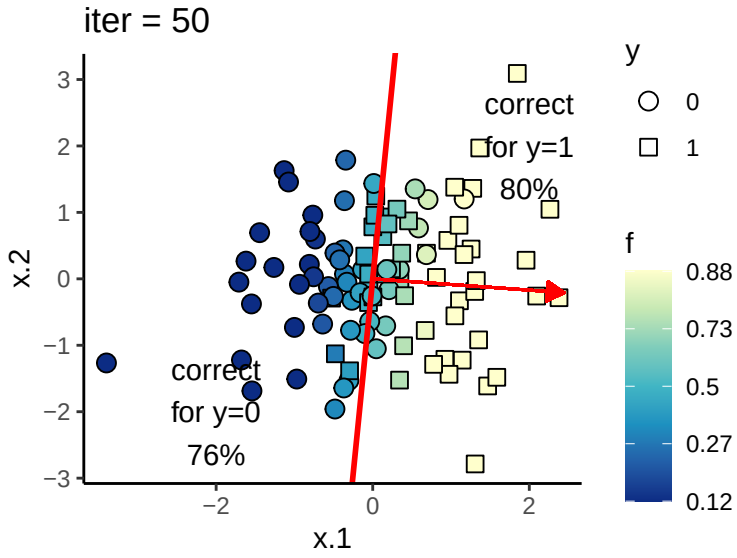
Binary classification in action



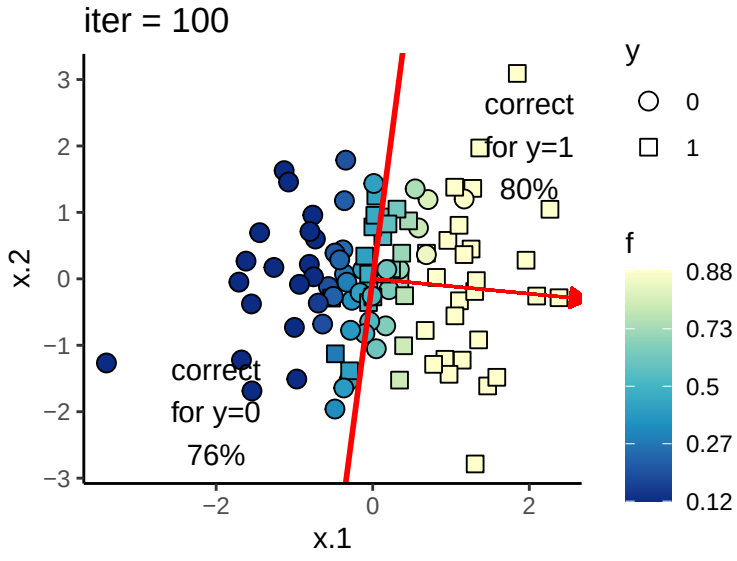
Binary classification in action



Binary classification in action



Binary classification in action



Today's lecture

General discussions on supervised learning

Classification with a decision rule

Generalized linear model

Generative Modelling approach

Other Methods

Revisit logistic regression as a "linear" model

(blurring the distinction between regression and classification)

A **linear** combination of features:

$$\sum_{k=1}^{p} X_{ik}\beta_k$$

Revisit logistic regression as a "linear" model

(blurring the distinction between regression and classification)

A **linear** combination of features:

$$\sum_{k=1}^{p} X_{ik}\beta_k$$

Letting $\eta_i = \sum_{k=1}^{p} X_{ik}\beta_k$, we define log-likelihood as:

$$\mathcal{L} = \sum_{i=1}^{n} \log p(Y_i|\eta_i) = \sum_{i=1}^{n} \sigma(\eta_i) + (1 - Y_i)\sigma(-\eta_i)$$

where $\sigma(z) = 1/(1 + \exp(-z))$.

Many interesting data types can be modelled as a GLM

A linear combination of features

$$\eta_i = \sum_{k=1}^{p} X_{ik}\beta_k$$

can be morphed to many different types of Y:

$$Y_i \sim p(Y_i|\eta_i)$$

Many interesting data types can be modelled as a GLM

A linear combination of features

$$\eta_i = \sum_{k=1}^{p} X_{ik}\beta_k$$

can be morphed to many different types of Y:

$$Y_i \sim p(Y_i|\eta_i)$$

Poisson

$$p(Y_i|\lambda_i) = \frac{\lambda_i^{Y_i} e^{-\lambda_i}}{Y_i!}$$

where $\lambda_i = \exp(\eta_i)$

Many interesting data types can be modelled as a GLM

A linear combination of features

$$\eta_i = \sum_{k=1}^{p} X_{ik}\beta_k$$

can be morphed to many different types of Y:

$$Y_i \sim p(Y_i|\eta_i)$$

Poisson

$$p(Y_i|\lambda_i) = \frac{\lambda_i^{Y_i} e^{-\lambda_i}}{Y_i!}$$

where $\lambda_i = \exp(\eta_i)$

Gamma

$$p(Y_i|\mu_i, \phi) = \frac{(\phi\mu_i)^{-1/\phi}}{\Gamma(1/\phi)} Y_i^{1/\phi - 1} e^{-Y_i/\mu_i\phi}$$

where $\mu_i = \exp(\eta_i)$ and ϕ is an overdispersion parameter

Negative Binomial GLM: directly modelling RNA-seq count

Poisson and Gamma distributions are the building blocks of NB.

$$p(Y_i|\mu_i,\phi) = \int \overbrace{p(Y_i|\lambda_i)}^{\text{Poisson}} \underbrace{p(\lambda_i|\mu_i,\phi)}_{\text{Gamma}} d\lambda_i$$

where we model

$$\mu_i = \exp\left(\sum_{k=1}^{p} X_{ik}\beta_k\right).$$

Negative Binomial GLM: directly modelling RNA-seq count

Poisson and Gamma distributions are the building blocks of NB.

$$
\begin{aligned}
p(Y_i|\mu_i,\phi) &= \int \overbrace{p(Y_i|\lambda_i)}^{\text{Poisson}} \underbrace{p(\lambda_i|\mu_i,\phi)}_{\text{Gamma}} d\lambda_i \\
&= \underbrace{\frac{\Gamma(Y_i+1/\phi)}{\Gamma(Y_i+1)\Gamma(1/\phi)}}_{\text{negative binomial}} \underbrace{\left(\frac{\mu_i}{1/\phi+\mu_i}\right)^{Y_i}}_{\text{success rate}} \underbrace{\left(\frac{1/\phi}{1/\phi+\mu_i}\right)^{1/\phi}}_{\text{failure rate}}
\end{aligned}
$$

where we model

$$
\mu_i = \exp\left(\sum_{k=1}^{p} X_{ik}\beta_k\right).
$$

Negative Binomial GLM: directly modelling RNA-seq count

- Y: number of successfully "observed" reads in RNA-seq (~targeting)
- r: number of permitted "dropped" reads until Y observed (~budget)
- ρ: success rate

$$p(Y_i|\mu_i,\phi) = \underbrace{\binom{Y_i + r - 1}{Y_i}}_{\text{negative binomial}} \underbrace{\rho_i^{Y_i}}_{\text{success rate}} \underbrace{(1-\rho_i)^r}_{\text{drop rate}}$$

Negative Binomial GLM: directly modelling RNA-seq count

- Y: number of successfully "observed" reads in RNA-seq (~targeting)
- r: number of permitted "dropped" reads until Y observed (~budget)
- ρ: success rate

$$
\begin{aligned}
p(Y_i|\mu_i,\phi) &= \underbrace{\binom{Y_i+r-1}{Y_i}}_{\text{negative binomial}} \underbrace{\rho_i^{Y_i}}_{\text{success rate}} \underbrace{(1-\rho_i)^r}_{\text{drop rate}} \\
&= \mathsf{NB}(Y_i|r=\phi^{-1},\rho_i=\mu_i/(\phi^{-1}+\mu_i))
\end{aligned}
$$

Negative Binomial GLM: directly modelling RNA-seq count

- Y: number of successfully "observed" reads in RNA-seq (~targeting)
- r: number of permitted "dropped" reads until Y observed (~budget)
- ρ: success rate

$$
\begin{aligned}
p(Y_i|\mu_i,\phi) &= \underbrace{\binom{Y_i + r - 1}{Y_i}}_{\text{negative binomial}} \underbrace{\rho_i^{Y_i}}_{\text{success rate}} \underbrace{(1-\rho_i)^r}_{\text{drop rate}} \\
&= \mathsf{NB}(Y_i|r = \phi^{-1}, \rho_i = \mu_i/(\phi^{-1} + \mu_i)) \\
\text{or} \quad &= \mathsf{NB}(Y_i|\text{mean} = \mu_i, \text{overdispersion} = \phi)
\end{aligned}
$$

Negative Binomial GLM: directly modelling RNA-seq count

- Y: number of successfully "observed" reads in RNA-seq (~targeting)
- r: number of permitted "dropped" reads until Y observed (~budget)
- ρ: success rate

$$
\begin{aligned}
p(Y_i|\mu_i,\phi) &= \underbrace{\binom{Y_i+r-1}{Y_i}}_{\text{negative binomial}} \underbrace{\rho_i^{Y_i}}_{\text{success rate}} \underbrace{(1-\rho_i)^r}_{\text{drop rate}} \\
&= \mathsf{NB}(Y_i|r=\phi^{-1},\rho_i=\mu_i/(\phi^{-1}+\mu_i)) \\
\text{or} \quad &= \mathsf{NB}(Y_i|\text{mean}=\mu_i,\text{overdispersion}=\phi)
\end{aligned}
$$

We can check:

mean: $\mathbb{E}[Y_i|r,\rho] = \rho r/(1-\rho) = \mu_i$

variance: $\mathbb{V}[Y_i|r,\rho] = \rho r/(1-\rho)^2 = \mu_i + \mu_i^2\phi$ (overdispersed mean-variance)

Four Steps for Supervised Learning

1. Gather some training data (this is a part of research!):
2. Write down a model (classifier) $f : \mathcal{X} \to \mathcal{Y}$
3. Fit the model to the training data
4. Use the model

Step 1. Cell type deconvolution data

Data set

- Human pancreatic islet gene expression data: GSE50244
- Single cell RNA-seq data in the same tissue: E-MTAB-5061
- A processed data set is available here: https://github.com/xuranw/MuSiC/tree/master/vignettes/data

Problem definition

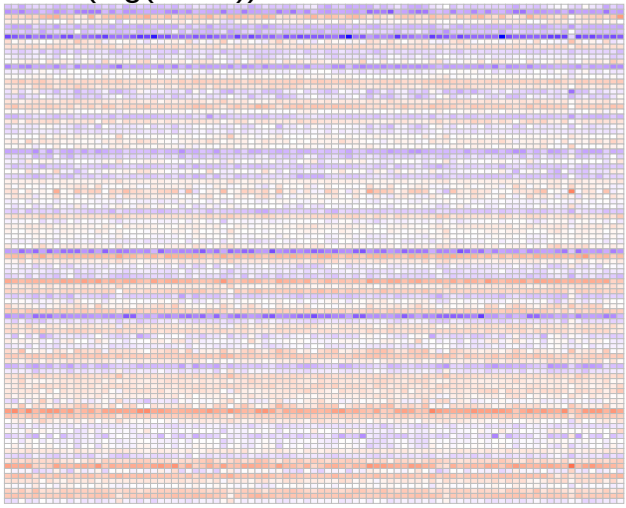
- $\mathbf{y}$: bulk gene expression (gene $\times$ sample)
- X: cell-type-specific single-cell expression (gene $\times$ cell type)

Goal

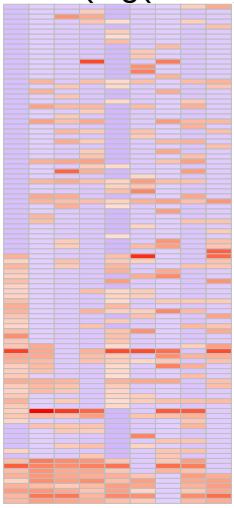
1. Fit a model regressing the bulk profile $\mathbf{y}_i$ of a sample i on the single-cell-type-specific matrix X.
2. What are the estimated cell type fractions in the bulk sample?

Step 1. (show the data)

scale(log(1 + Y))



scale(log(1 + X))



Step 2. NB GLM for the cell type deconvolution problem

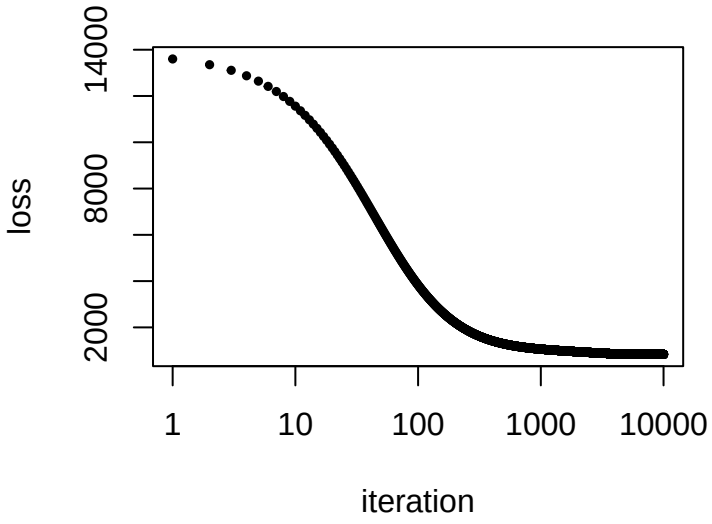
Q: Can we estimate cell type fractions in tissue-level *bulk* data?

$$\overset{\text{bulk}}{\mathbf{y}_i} \sim \text{NB}\left(\text{mean} = \overset{\text{scale factor}}{s_i} \sum_{t} X_{gt} \overset{\text{GOAL}}{\theta_{ti}}, \text{overdispersion} = \phi\right)$$

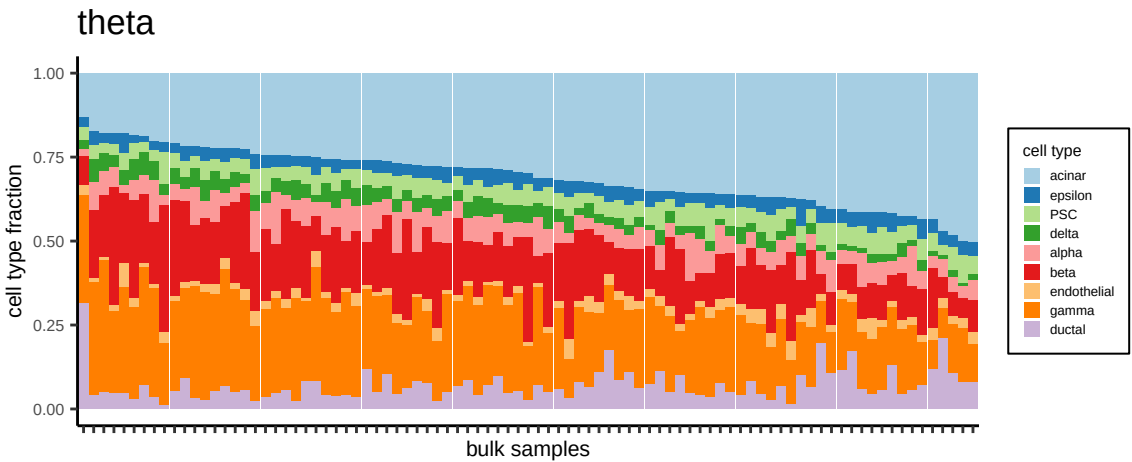
cell-type-sorted

- We use the same data set used in the vignette of MuSic package (Wang *et al.*, Nature Comm., 2019)

Step 3. Fit the model to the data



Step 4. Show the cell type fraction estimates



Today's lecture

General discussions on supervised learning

Classification with a decision rule

Generalized linear model

Generative Modelling approach

Other Methods

Discriminative vs. generative approach

- Y: class label/outcome, X: covariates/predictor variables

Discriminative learning

- Directly learn $p(Y|X)$

Generative learning

- First learn $p(X|Y)$
- Apply **Bayes** rule: $p(Y|X) \propto p(X|Y)p(Y)$

- If we have domain-specific knowledge, $p(X|Y)$, a generative-modelling approach can be more powerful.
- If our focus is solely on classification, a discriminative learning approach usually outperforms with small to moderate sample size.

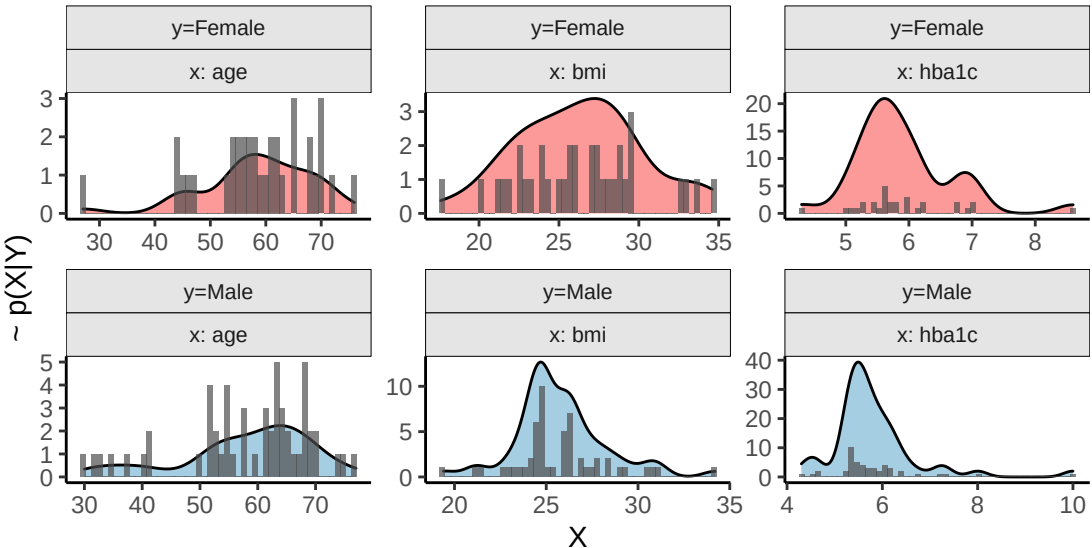
Naive Bayes Classifier: How do we estimate $P(X|Y)$?

If $\mathbf{x}_i = (X_{i1}, \ldots, X_{ip})$, how do we estimate $p(\mathbf{x}_i|Y_i = 1)$ or $p(\mathbf{x}_i|Y_i = 0)$?

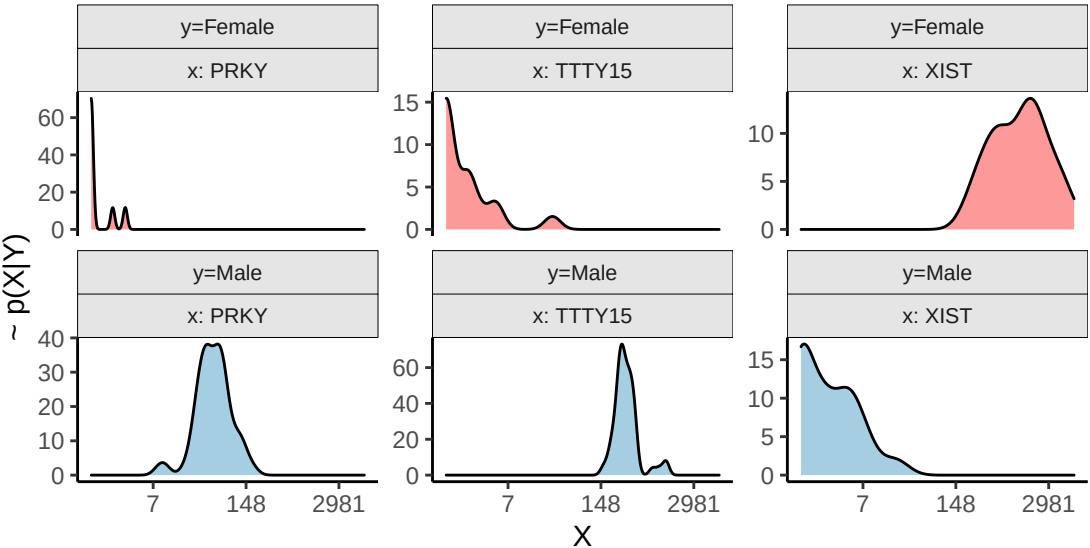
$$p(X_{i1}, \ldots, X_{ip}|Y_i = y_i) \underset{\text{fully-factored}}{\approx} \prod_{k=1} p(X_{ik}|Y_i)$$

- Is this accurate?
- What have we missed?

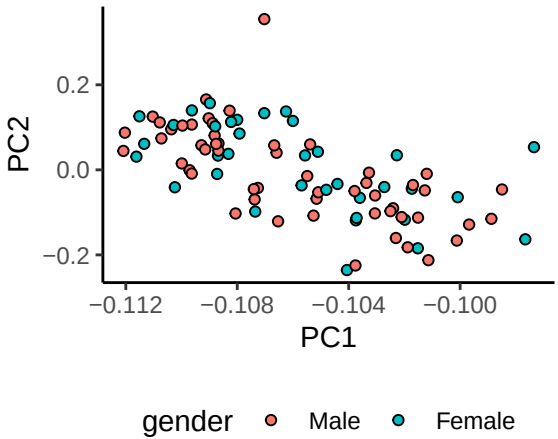
Let's take an example from the pancreatic islet data (GSE50244).



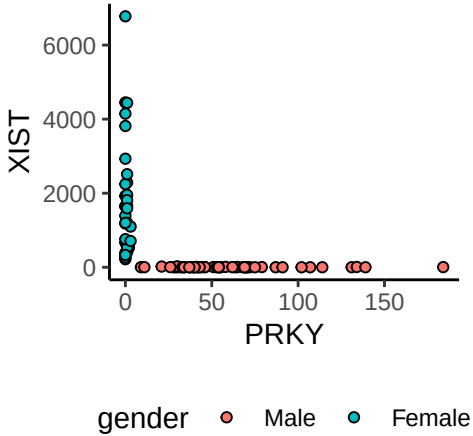
Let's take an example from the pancreatic islet data (GSE50244).



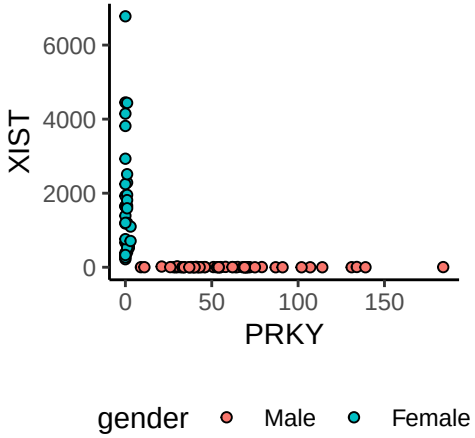
Not every gene can predict well

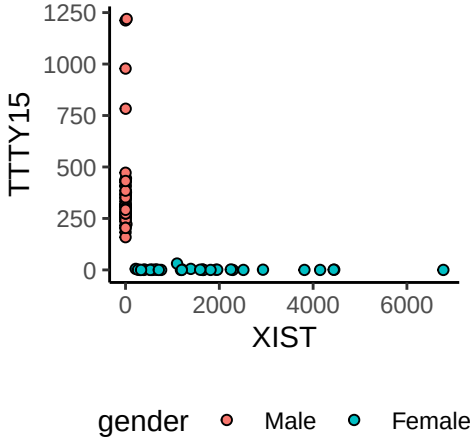


These genes are very informative



These genes are very informative





Naive Bayes Classifier

$$p(Y_i = 1|\mathbf{x}_i) \approx \frac{\overbrace{p(Y_i = 1)}^{\text{prior}} \overbrace{\prod_k \hat{p}(X_{ik}|\theta_{k1})}^{\text{fully-factored}}}{\sum_{c=0}^{1} p(Y_i = c) \prod_k \hat{p}(X_{ik}|\theta_{kc})}$$

- Prior can be $1/2$ if the class labels are balanced (or $1/K$ for K classes).

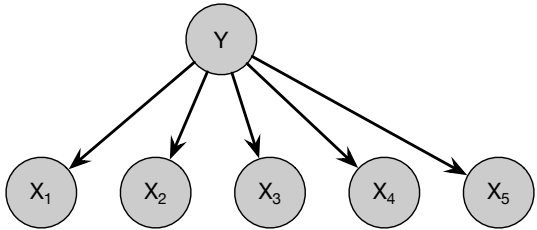
Naive Bayes Classifier

$$p(Y_i = 1|\mathbf{x}_i) \approx \frac{\overbrace{p(Y_i = 1)}^{\text{prior}} \overbrace{\prod_k \hat{p}(X_{ik}|\theta_{k1})}^{\text{fully-factored}}}{\sum_{c=0}^{1} p(Y_i = c) \prod_k \hat{p}(X_{ik}|\theta_{kc})}$$

$$\approx \frac{p(Y_i = 1) \prod_k \overbrace{\mathcal{N}(X_{ik}|\mu_{k1}, \sigma_{k1}^2)}^{\text{Gaussian approximation}}}{\sum_{c=0}^{1} p(Y_i = c) \prod_k \underbrace{\mathcal{N}(X_{ik}|\mu_{kc}, \sigma_{kc}^2)}_{\text{Gaussian approximation}}}$$

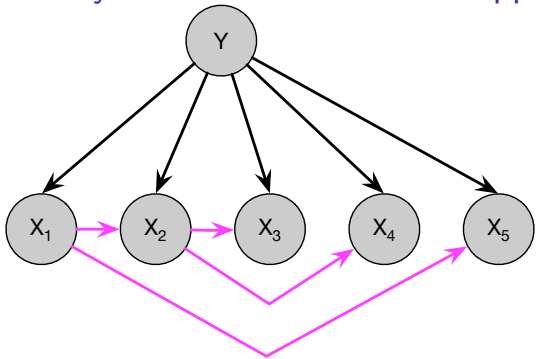
- Prior can be $1/2$ if the class labels are balanced (or $1/K$ for K classes).

Can we improve Naive Bayes Classifier?



$$p(Y_i = 1|\mathbf{x}_i) \approx \frac{\overbrace{p(Y_i = 1)}^{\text{prior}} \overbrace{\prod_k \hat{p}(X_{ik}|\theta_{k1})}^{\text{fully-factored}}}{\sum_{c=0}^{1} p(Y_i = c) \prod_k \hat{p}(X_{ik}|\theta_{kc})}$$

Tree-augmented Naive Bayes Classifier can better approximate $P(X|Y)$



$$p(Y_i = 1|\mathbf{x}_i) \approx \frac{\overbrace{p(Y_i = 1)}^{\text{prior}} \overbrace{\prod_k \hat{p}(X_{ik}|X_{i\text{Pa}(k)}, \theta_{k1})}^{\text{tree-structured}}}{\sum_{c=0}^{1} p(Y_i = c) \prod_k \hat{p}(X_{ik}|X_{i\text{Pa}(k)}, \theta_{kc})}$$

- Can we use prior knowledge, such as Gene Ontology?

Discussions

- Can we include all the genes in a NB classifier?
- Can we first try out DEG analysis to select informative features?
- How do we deal with high-dimensional classification problems?

Today's lecture

General discussions on supervised learning

Classification with a decision rule

Generalized linear model

Generative Modelling approach

Other Methods

1. K-nearest neighbour classifier (will discuss in the single-cell lecture)
2. Running average estimator
   - Local piece-wise regression (`loess`)
   - Useful for low-dimensional settings
3. Ensemble models (previous + next lectures)
   - A mixture of (logistic) regression
   - Boosting/bagging (combining the power of many weak classifiers)
4. Deep learning (will discuss in the single-cell lecture)