Boolean Classification

Jong-Han Kim

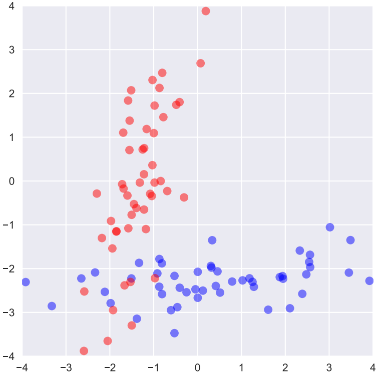
EE787 Machine learning
Kyung Hee University

Boolean classification

Boolean classification

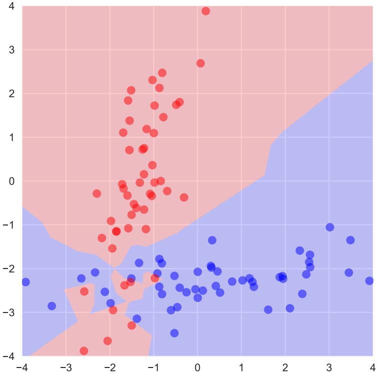
- supervised learning is called *boolean classification* when raw output variable v is a categorical that can take two possible values
- we denote these -1 and 1, and they often correspond to $\{\text{FALSE}, \text{TRUE}\}$ or $\{\text{NEGATIVE}, \text{POSITIVE}\}$
- for a data record u^i, v^i, the value $v^i \in \{-1, 1\}$ is called the *class* or *label*
- a *boolean classifier* predicts label $\hat{v}$ given raw input u

Classification



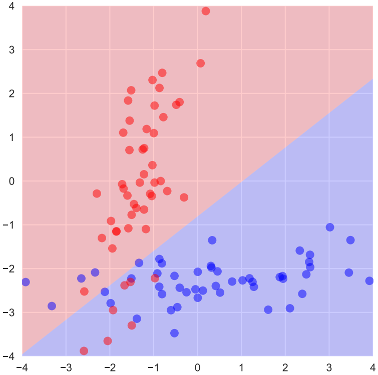
- here $u \in \mathbf{R}^2$
- red points have $v^i = -1$, blue points have $v^i = 1$
- we'd like a predictor that maps any $u \in \mathbf{R}^2$ into prediction $\hat{v} \in \{-1, 1\}$

Example: Nearest neighbor classsifier



- given u, let $k = \mathrm{argmin}_k \|u - u^k\|$, then predict $\hat{v} = v^k$
- red region is the set of u for which prediction is -1
- blue region is the set of u for which prediction is 1
- zero training error (all points classified correctly), but perhaps overfit

Example: Least squares classifier



- embed $x = (1, u)$ and $y = v$, apply least squares regression
- gives $\hat{y} = \theta_1 + \theta_2 u_1 + \theta_3 u_2$
- predict using $\hat{v} = \text{sign}(\hat{y})$
- 11% of points misclassified at training

Confusion matrix

The two types of errors

- measure performance of a specific predictor on a set of n data records
- each data point i has $v^i \in \{-1, 1\}$
- and corresponding prediction $\hat{v}^i = g(v^i) \in \{-1, 1\}$
- only four possible values for the data pair $\hat{v}^i$, v^i:
  - *true positive* if $\hat{v}^i = 1$ and $v^i = 1$
  - *true negative* if $\hat{v}^i = -1$ and $v^i = -1$
  - *false negative* or *type II error* if $\hat{v}^i = -1$ and $v^i = 1$
  - *false positive* or *type I error* if $\hat{v}^i = 1$ and $v^i = -1$

Confusion matrix

- for a predictor and a data set define the *confusion matrix*

$$C = \begin{bmatrix} \#\text{ true negatives} & \#\text{ false negatives} \\ \#\text{ false positives} & \#\text{ true positives} \end{bmatrix} = \begin{bmatrix} C_{\mathsf{tn}} & C_{\mathsf{fn}} \\ C_{\mathsf{fp}} & C_{\mathsf{tp}} \end{bmatrix}$$

  (warning: some people use the transpose of C)

- $C_{\mathsf{tn}} + C_{\mathsf{fn}} + C_{\mathsf{fp}} + C_{\mathsf{tp}} = n$ (total number of examples)
- $N_{\mathsf{n}} = C_{\mathsf{tn}} + C_{\mathsf{fp}}$ is number of negative examples
- $N_{\mathsf{p}} = C_{\mathsf{fn}} + C_{\mathsf{tp}}$ is number of positive examples
- diagonal entries give numbers of correct predictions
- off-diagonal entries give numbers of incorrect predictions of the two types

Some boolean classification measures

- confusion matrix $\begin{bmatrix} C_{\mathsf{tn}} & C_{\mathsf{fn}} \\ C_{\mathsf{fp}} & C_{\mathsf{tp}} \end{bmatrix}$

- the basic error measures:
  - *false positive rate* is C_{fp}/n
  - *false negative rate* is C_{fn}/n
  - *error rate* is $(C_{\mathsf{fn}} + C_{\mathsf{fp}})/n$

- error measures some people use:
  - *true positive rate* or *sensitivity* or *recall* is $C_{\mathsf{tp}}/N_{\mathsf{p}}$
  - *false alarm rate* is $C_{\mathsf{fp}}/N_{\mathsf{n}}$
  - *specificity* or *true negative rate* is $C_{\mathsf{tn}}/N_{\mathsf{n}}$
  - *precision* is $C_{\mathsf{tp}}/(C_{\mathsf{tp}} + C_{\mathsf{fp}})$

Neyman-Pearson error

- *Neyman-Pearson error* over a data set is $\kappa C_{\mathsf{fn}}/n + C_{\mathsf{fp}}/n$
- a scalarization of our two objectives, false positive and false negative rates
- κ is how much more false negatives irritate us than false positives
- when $\kappa = 1$, the Neyman-Pearson error is the *error rate*

- we'll use the Neyman-Pearson error as our scalarized measure

ERM

Embedding

- we embed raw input and output records as $x = \phi(u)$ and $y = \psi(v)$
- ϕ is the feature map
- ψ is *the identity map*, $\psi(v) = v$
- un-embed by $\hat{v} = \mathrm{sign}(\hat{y})$
- equivalent to $\hat{v} = \underset{v \in \{-1,1\}}{\mathrm{argmin}}\, |\hat{y} - \psi(v)|$
- *i.e.*, choose the nearest boolean value to the (real) prediction $\hat{y}$

ERM

- given loss function $\ell(\hat{y}, y)$, *empirical risk* on a data set is

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \ell(\hat{y}^i, y^i)$$

- for linear model $\hat{y} = \theta^\mathsf{T} x$, with $\theta \in \mathbf{R}^d$,

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta^\mathsf{T} x^i, y^i)$$

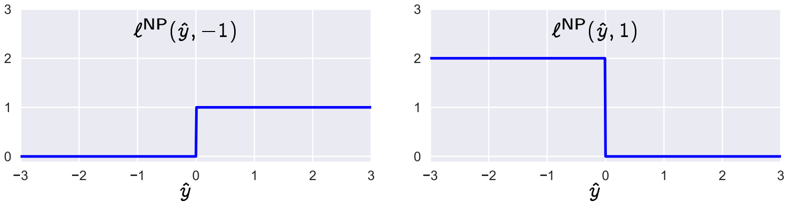
- ERM: choose θ to minimize $\mathcal{L}(\theta)$
- regularized ERM: choose θ to minimize $\mathcal{L}(\theta) + \lambda r(\theta)$, with $\lambda > 0$

Loss functions for boolean classification

- to apply ERM, we need a loss function on embedded variables $\ell(\hat{y}, y)$
- y can only take values -1 or 1
- but $\hat{y} = \theta^\mathsf{T} x \in \mathbf{R}$ can be any real number
- to specify ℓ, we only need to give two functions (of a scalar $\hat{y}$):
    - $\ell(\hat{y}, -1)$ is how much $\hat{y}$ irritates us when $y = -1$
    - $\ell(\hat{y}, 1)$ is how much $\hat{y}$ irritates us when $y = 1$
- we can take $\ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1)$, to reflect that false negatives irritate us a factor κ more than false positives

Neyman-Pearson loss

- Neyman-Pearson loss is
  - $\ell^{\text{NP}}(\hat{y}, -1) = \begin{cases} 1 & \hat{y} \geq 0 \\ 0 & \hat{y} < 0 \end{cases}$
  - $\ell^{\text{NP}}(\hat{y}, 1) = \kappa l^{\text{NP}}(\hat{y}, -1) = \begin{cases} \kappa & \hat{y} < 0 \\ 0 & \hat{y} \geq 0 \end{cases}$
- empirical Neyman-Pearson risk $\mathcal{L}^{\text{NP}}$ is the Neyman-Pearson error



The problem with Neyman-Pearson loss

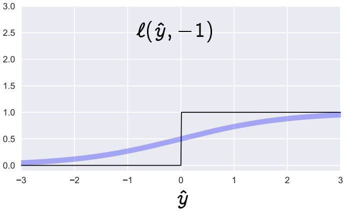
- empirical Neyman-Pearson risk $\mathcal{L}^{\mathrm{NP}}(\theta)$ is not differentiable, or even continuous (and certainly not convex)
- worse, its gradient $\nabla\mathcal{L}^{\mathrm{NP}}(\theta)$ is either zero or undefined
- so an optimizer does not know how to improve the predictor

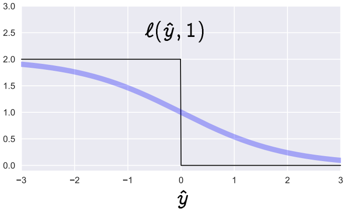
Idea of proxy loss

- we get better results using a *proxy loss* that
  - approximates, or at least captures the flavor of, the Neyman-Pearson loss
  - is more easily optimized (*e.g.*, is convex or has nonzero derivative)

- we want a proxy loss function
  - with $\ell(\hat{y}, -1)$ small when $\hat{y} < 0$, and larger when $\hat{y} > 0$
  - with $\ell(\hat{y}, +1)$ small when $\hat{y} > 0$, and larger when $\hat{y} < 0$
  - which has other nice characteristics, *e.g.*, differentiable or convex

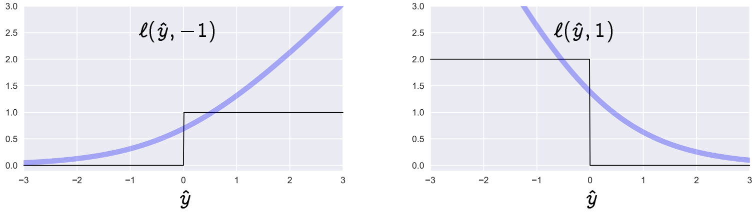
Sigmoid loss





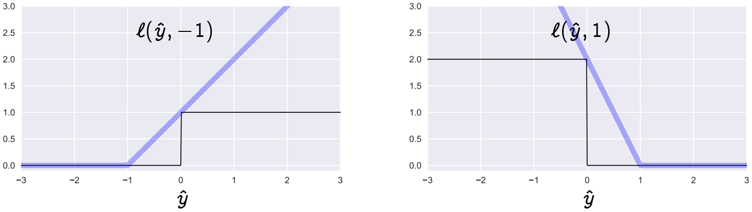
- $\ell(\hat{y}, -1) = \dfrac{1}{1 + e^{-\hat{y}}}, \quad \ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1) = \dfrac{\kappa}{1 + e^{\hat{y}}}$
- differentiable approximation of Neyman-Pearson loss
- but not convex

Logistic loss



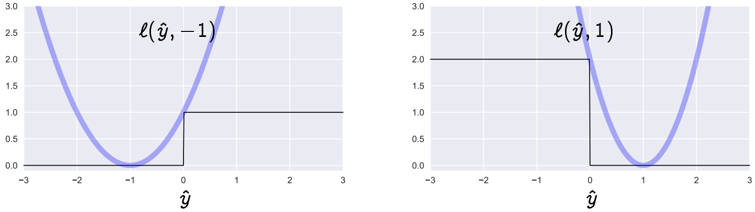
- $\ell(\hat{y}, -1) = \log(1 + e^{\hat{y}}), \quad \ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1) = \kappa \log(1 + e^{-\hat{y}})$
- differentiable and convex approximation of Neyman-Pearson loss

Hinge loss



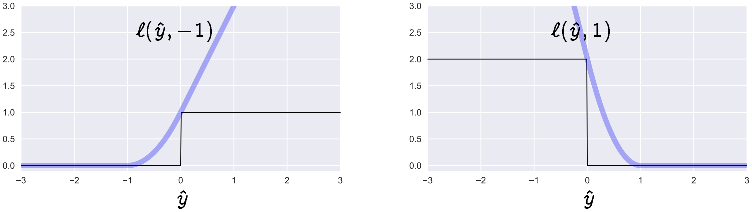
- $\ell(\hat{y}, -1) = (1 + \hat{y})_+, \quad \ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1) = \kappa(1 - \hat{y})_+$
- nondifferentiable but convex approximation of Neyman-Pearson loss

Square loss



- $\ell(\hat{y}, -1) = (1 + \hat{y})^2, \quad \ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1) = \kappa(1 - \hat{y})^2$
- ERM is least squares problem

Hubristic loss



- define the *hubristic loss* (huber + logistic) as

$$\ell(\hat{y}, -1) = \begin{cases} 0 & \hat{y} < -1 \\ (\hat{y}+1)^2 & -1 \leq \hat{y} \leq 0 \\ 1 + 2\hat{y} & \hat{y} > 0 \end{cases}$$

- $\ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1)$

Boolean classifiers

Least squares classifier

- use empirical risk with square loss

$$\mathcal{L}(\theta) = \frac{1}{n}\left(\sum_{i:y^i=-1}(1+\hat{y}^i)^2 \;+\; \kappa \sum_{i:y^i=1}(1-\hat{y}^i)^2\right)$$

  and your choice of regularizer

- with sum squares regularizer, this is *least squares classifier*
- we can minimize $\mathcal{L}(\theta) + \lambda r(\theta)$ using, *e.g.*, QR factorization

Logistic regression

- use empirical risk with logistic loss

$$\mathcal{L}(\theta) = \frac{1}{n}\left(\sum_{i:y^i=-1} \log(1+e^{\hat{y}^i}) \; + \; \kappa \sum_{i:y^i=1} \log(1+e^{-\hat{y}^i})\right)$$

  and your choice of regularizer
- can minimize $\mathcal{L}(\theta) + \lambda r(\theta)$ using prox-gradient method
- we will find an actual minimizer if r is convex

Support vector machine

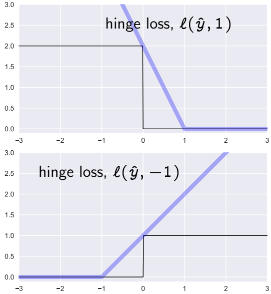
(usually abbreviated as *SVM*)

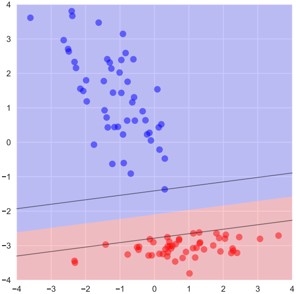
- use empirical risk with hinge loss

$$\mathcal{L}(\theta) = \frac{1}{n}\left(\sum_{i:y^i=-1}(1+\hat{y}^i)_+ \;+\; \kappa \sum_{i:y^i=1}(1-\hat{y}^i)_+\right)$$

  and sum squares regularizer
- $\mathcal{L}(\theta) + \lambda r(\theta)$ is convex
- it can be minimized by various methods (but not prox-gradient)

Support vector machine





- decision boundary is $\theta^\mathsf{T} x = 0$
- black lines show points where $\theta^\mathsf{T} x = \pm 1$
- what is the training risk here?

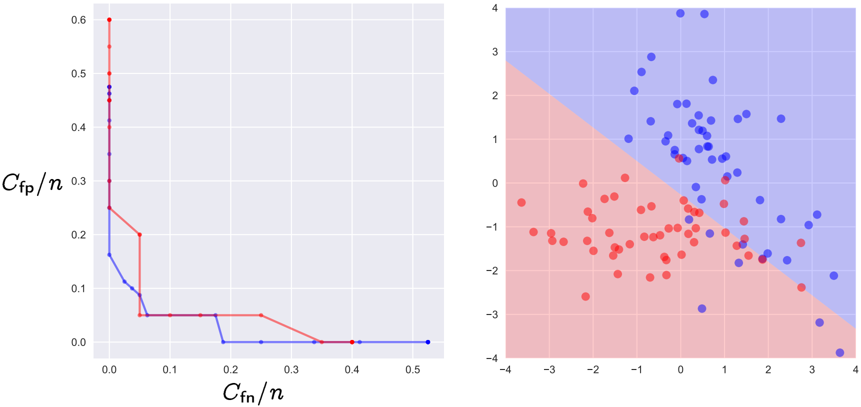
ROC

Receiver operating characteristic

(always abbreviated as *ROC*, comes from WWII)

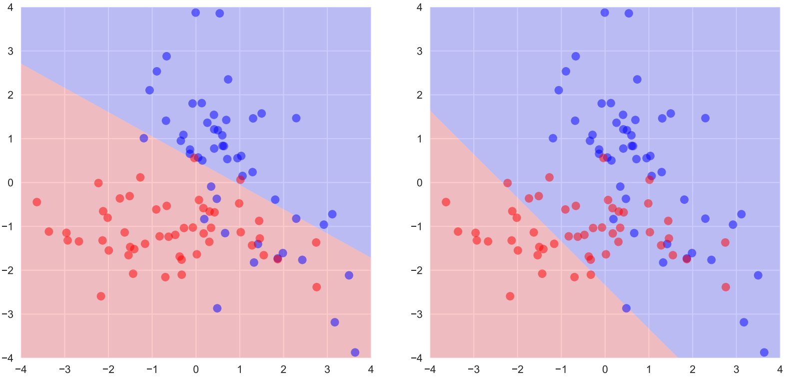
- explore trade-off of false negative versus false positive rates
- create classifier for many values of κ
- for each choice of κ, select hyper-parameter λ via validation on test set with Neyman-Pearson risk
- plot the test (and maybe train) false negative and false positive rates against each other
- called *receiver operating characteristic* (ROC) (when viewed upside down)

Example



- square loss, sum squares regularizer
- left hand plot shows training errors in blue, test errors in red
- right hand plot shows minimum-error classifier (*i.e.*, $\kappa = 1$)

Example



- left hand plot shows predictor when $\kappa = 0.4$
- right hand plot shows predictor when $\kappa = 4$