Boolean Classification

Sanjay Lall and Stephen Boyd

EE104
Stanford University

Boolean classification

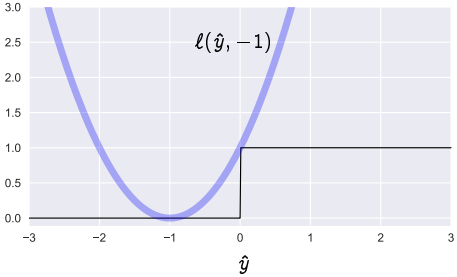
- embed the two classes as $y = \pm 1$
- use RERM to fit, with various loss functions and regularizers
- validate using Neyman-Pearson metric on test data, $\kappa E_{\mathsf{fn}} + E_{\mathsf{fp}}$
  - κ is our relative distaste for mistaking a positive example
  - for $\kappa = 1$, reduces to error rate

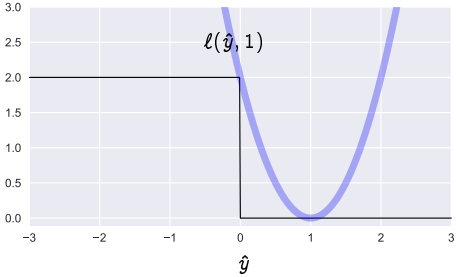
Loss functions

Loss functions for Boolean classification

- y can only take values -1 or 1, so to specify ℓ, we only need to give two functions of $\hat{y}$:
  - $\ell(\hat{y}, -1)$ 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 will define ℓ via a penalty function $p : \mathbf{R} \to \mathbf{R}$
  - $\ell(\hat{y}, -1) = p(\hat{y})$
  - $\ell(\hat{y}, 1) = \kappa p(-\hat{y}) = \kappa \ell(-\hat{y}, -1)$
- $p(\hat{y})$ should be small for $\hat{y}$ negative
- $p(\hat{y})$ should be larger $\hat{y}$ positive
- κ gives our relative dislike of mistaking $y = 1$

Square loss

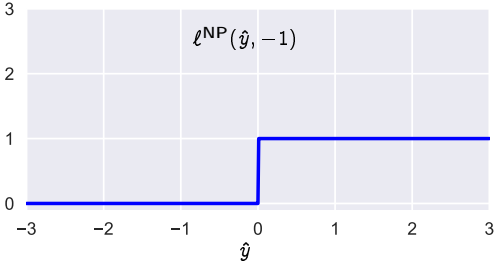


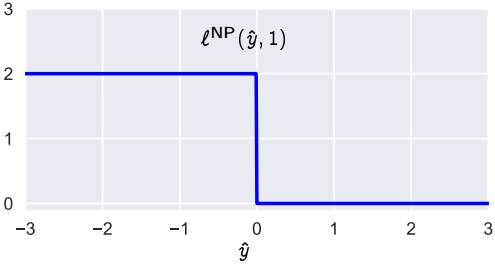


- $\ell(\hat{y}, -1) = (1 + \hat{y})^2, \quad \ell(\hat{y}, 1) = \kappa\ell(-\hat{y}, -1) = \kappa(1 - \hat{y})^2$
- doesn't satisfy desired properties, *e.g.*, $\ell(-3, -1)$ should be very small, not large
- ERM is least squares problem, and so, easy to solve

Neyman-Pearson loss

- Neyman-Pearson loss is
  - $\ell^{\mathsf{NP}}(\hat{y}, -1) = \begin{cases} 1 & \hat{y} \geq 0 \\ 0 & \hat{y} < 0 \end{cases}$
  - $\ell^{\mathsf{NP}}(\hat{y}, 1) = \kappa l^{\mathsf{NP}}(\hat{y}, -1) = \begin{cases} \kappa & \hat{y} < 0 \\ 0 & \hat{y} \geq 0 \end{cases}$

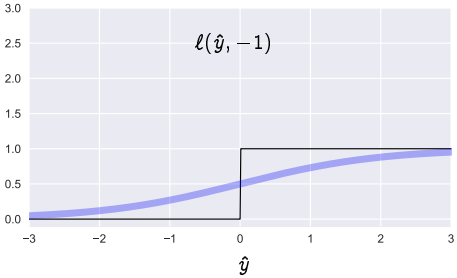


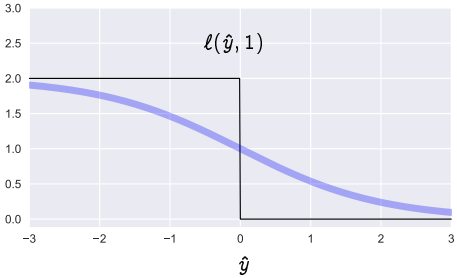


Neyman-Pearson loss

- it's the same as our performance metric, which would seem good
- but it's very hard to minimize $\mathcal{L}(\theta)$, since it's discontinuous, has zero derivative almost everywhere
- surprisingly, we get better performance using different loss functions, that are also easier to minimize
- if they're convex, and the regularizer is convex, we can solve the RERM problem efficiently

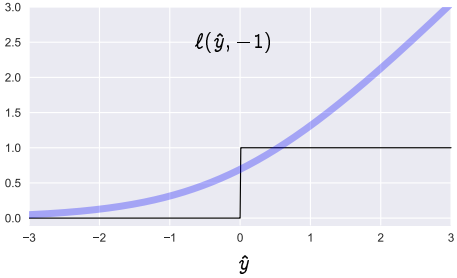
Sigmoid loss

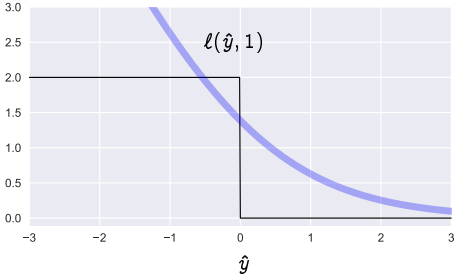




- $\ell(\hat{y}, -1) = \dfrac{1}{1 + e^{-\hat{y}}}, \quad \ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1) = \dfrac{\kappa}{1 + e^{\hat{y}}}$
- differentiable approximation of Neyman-Pearson loss
- but not convex

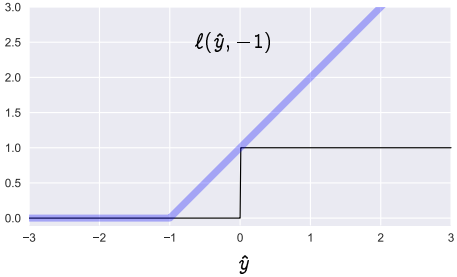
Logistic loss

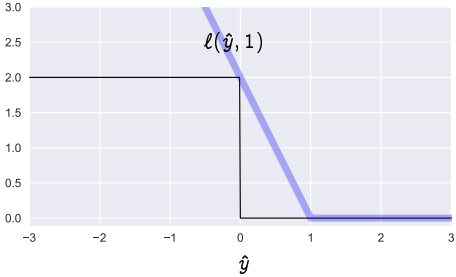




- $\ell(\hat{y}, -1) = \log(1 + e^{\hat{y}}), \quad \ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1) = \kappa \log(1 + e^{-\hat{y}})$
- differentiable and convex approximation of Neyman-Pearson loss

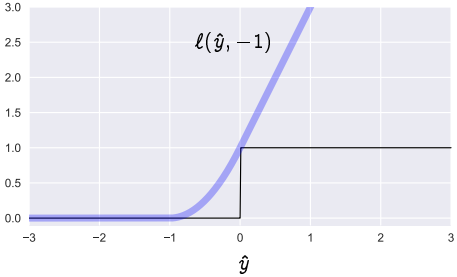
Hinge loss

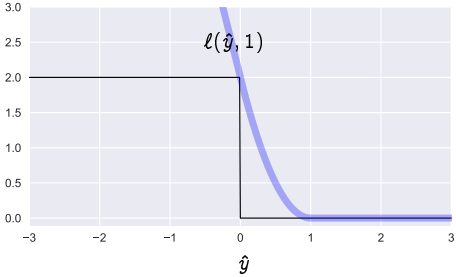




- $\ell(\hat{y}, -1) = (1 + \hat{y})_+, \quad \ell(\hat{y}, 1) = \kappa\ell(-\hat{y}, -1) = \kappa(1 - \hat{y})_+$
- another convex approximation of Neyman-Pearson loss

Hubristic loss





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

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

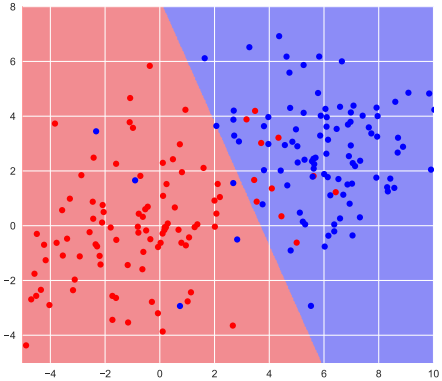
- $\ell(\hat{y}, 1) = \kappa \ell(-\hat{y}, -1)$

Boolean classifiers

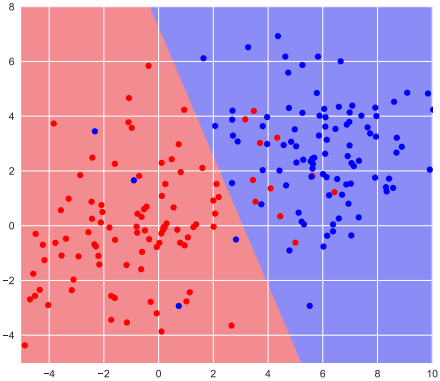
Boolean classifiers with names

- *least squares classifier* uses square loss, square regularizer
- *logistic regression* uses logistic loss, any regularizer, as in, logistic regression with ℓ_1 regularizer
- *support vector machine* (SVM) uses hinge loss, square regularizer

Example

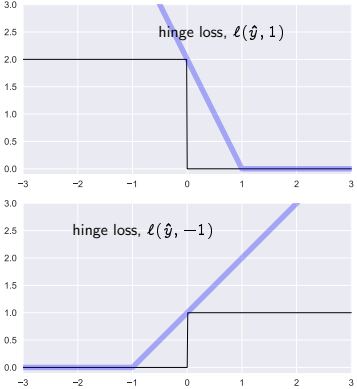


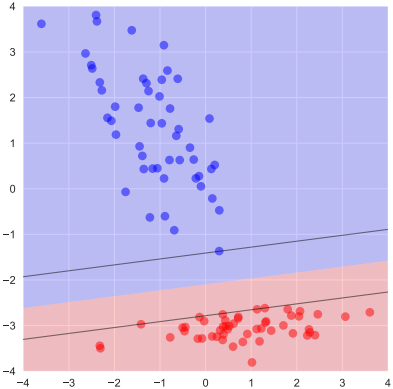
logistic loss: $C = \begin{bmatrix} 89 & 5 \\ 11 & 95 \end{bmatrix}$



squared loss: $C = \begin{bmatrix} 89 & 4 \\ 11 & 96 \end{bmatrix}$

Support vector machine





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

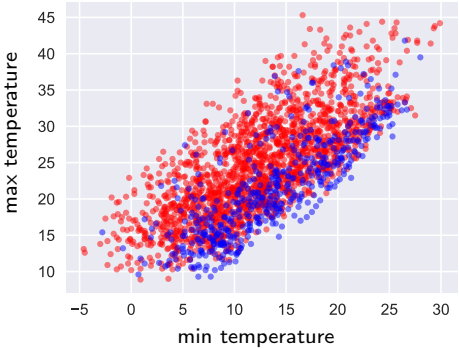
Example: Australian weather

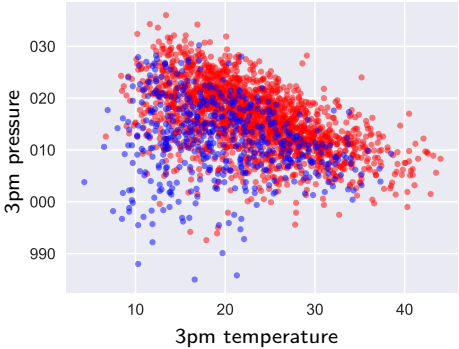
- we have measurements of multiple attributes of weather at multiple locations in Australia
- over 10 years from 2007 to 2017
- 142,193 records
- given measurements from today, predict if it will rain tomorrow
- removing records with missing data leaves 112,925 records
- data from Australian weather stations, downloaded from `https://www.kaggle.com/jsphyg/weather-dataset-rattle-package`

Example: Australian weather

- numeric fields
  - MinTemp, MaxTemp, Rainfall, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, Temp9am, Temp3pm
- categorical fields
  - location (44 possible locations)
  - WindGustDir, WindDir9am, WindDir3pm (16 compass points)
  - RainToday (YES or NO)
- additional field: date

Some data





- here we look at a random 2% of the data, for a few features
- blue points indicate next day rainfall

Embedding

- for $x = \phi(u)$
  - embed 12 numeric fields via identity map
  - embed 3 wind directions as one-hot (16 compass points)
  - embed RainToday as $\{-1, 1\}$
  - do not use date or location fields (did not improve validation performance)
  - standardize
  - add constant feature
  - results in $x \in \mathbf{R}^{62}$
- embed $y = \psi(v)$ as $\{-1, 1\}$ where v is RainTomorrow

ERM

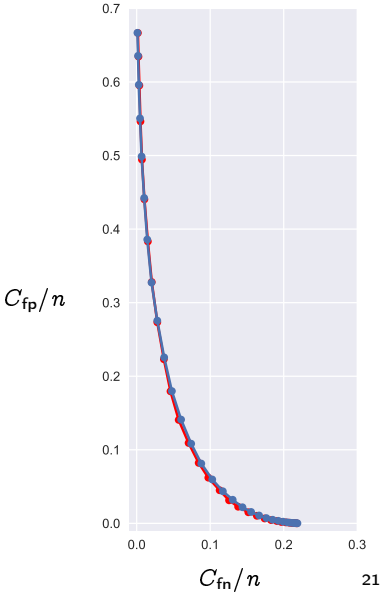
- use logistic loss function

$$\ell(\hat{y}, -1) = \log(1 + e^{\hat{y}}), \qquad \ell(\hat{y}, 1) = \kappa \log(1 + e^{-\hat{y}})$$

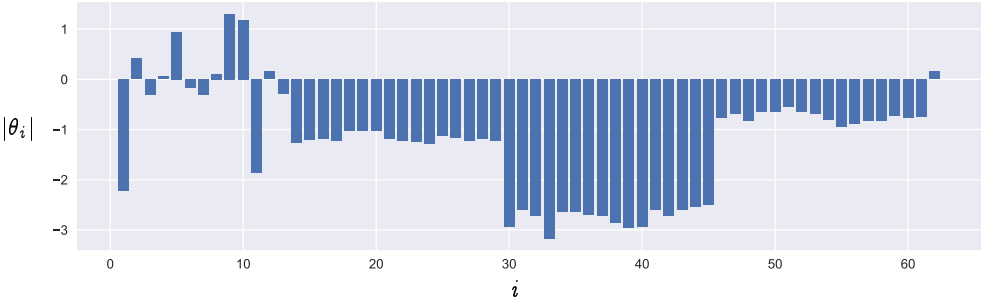
- linear predictor $\hat{y} = \theta^\top x$
- and square regularization $r(\theta) = ||\theta_{2:}||_2^2$

ROC

- randomly split 80/20 into train/test sets
- test and train results very similar (test in red, train in blue)
- minimum probability of error = 16%
- rain frequency = 22%, so a predictor that always predicts *no rain* will achieve 22% error



Important features



- important feature: Pressure9am - Pressure3pm ($i = 10, 11$)
- rapidly falling pressure indicates a storm is coming
- note 16 one-hot embedded values for WindDir9am ($i = 30, \ldots, 45$) all sum to one
- retraining with 6 features: MinTemp, MaxTemp, WindGustSpeed, Humidity3pm, Pressure9am, Pressure3pm achieves 16.5% probability of error