Linear Binary Classification

MATH-412 - Statistical Machine Learning

Outline

Outline

Classification and plug-in predictors

Input space $\mathcal{X}$, output space $\mathcal{Y} = \{-1, 1\}$, decision space $\mathcal{A} = \{-1, 1\}$ and 0-1 loss $\ell(a, y) = 1_{\{a \neq y\}} = 1_{\{ay \leq 0\}}$

- Empirical risk for 0-1 loss and $\gamma : \mathcal{X} \to \{-1, 1\}$

$$\widehat{\mathcal{R}}_n^{\text{0-1}}(\gamma) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{\gamma(x_i) \neq y_i\}}$$

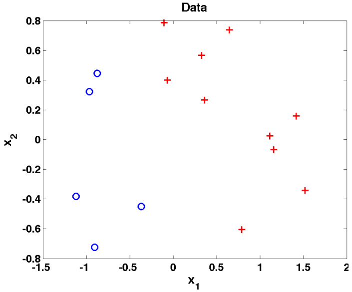
$\to$ Extend the definition of the 0-1 loss to real valued predictors by $\ell(a, y) = 1_{\{ay \leq 0\}}$

- Empirical risk for 0-1 loss and $f : \mathcal{X} \to \mathbb{R}$

$$\widehat{\mathcal{R}}_n^{\text{0-1}}(f) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{y_i\, f(x_i) \leq 0\}}$$

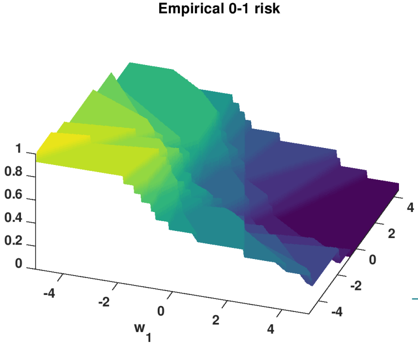
- Then use the *plug-in predictor* $\gamma(x_i) = \text{sign}(f(x_i))$.

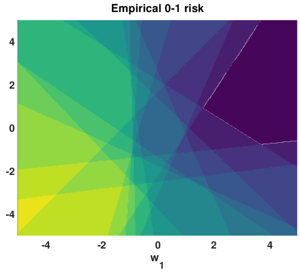
Hardness: Linear classification toy-example



$$\widehat{\mathcal{R}}_{0\text{-}1} = \frac{1}{n}\sum_{i=1}^{n} 1_{\{y_i(\boldsymbol{w}^\top \mathbf{x}_i + b) \le 0\}}$$

Plot of $\widehat{\mathcal{R}}_{0\text{-}1}$ as a function of $\boldsymbol{w} = (w_1, w_2)$ for $b = 1$:





- ER is non-convex and discontinuous

$\rightarrow$ NP-hard to optimize...

Outline

Classification *via* OLS regression

For regression, but assuming $Y \in \{-1, 1\}$

- the risk is
$$\mathbb{E}\left[\left(f(X) - Y\right)^2\right] = \mathbb{E}\left[\left(1 - Y f(X)\right)^2\right]$$
- the target function is $f^*(X) = \mathbb{E}[Y|X] \quad = 2\,\mathbb{P}(Y = 1|X) - 1$
- the excess risk is $\mathbb{E}\left[\left(f(X) - f^*(X)\right)^2\right]$

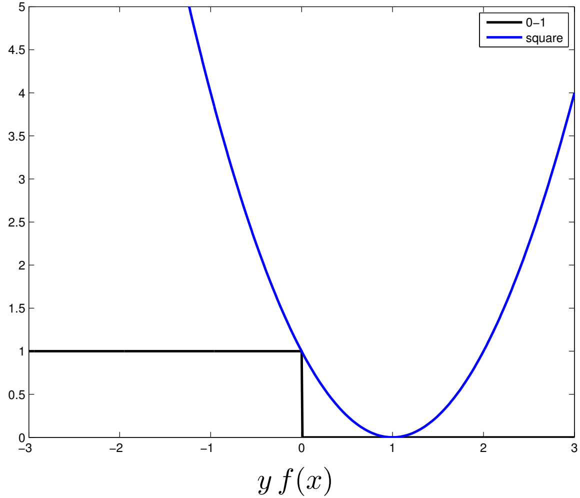
For classification

- the target function is $\arg\max_{y \in \{-1,1\}} \mathbb{P}(Y = y|x = x) = \mathrm{sign}(f^*(x))$

Plug-in principle

- Learn $\widehat{f}(x)$ using OLS regression
- Use the plug-in predictor for classification $\widehat{y} := \widehat{\gamma}(x) = \mathrm{sign}(\widehat{f}(x))$

Zero one loss *vs* square loss



0-1 loss

$$\ell(f(x), y) = 1_{\{y\,f(x) \leq 0\}}$$

Square loss

$$\ell(f(x), y) = (1 - y\,f(x))^2$$

Outline

Logistic regression

Logistic regression (Berkson, 1944)

Classification setting:

$$\mathcal{X} = \mathbb{R}^p, \mathcal{Y} \in \{0, 1\}.$$

Key assumption:

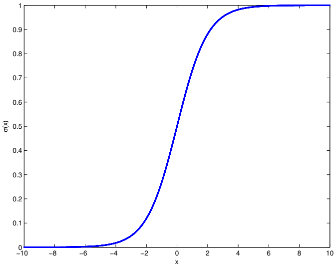
$$\log \frac{\mathbb{P}(Y = 1 \mid X = \mathbf{x})}{\mathbb{P}(Y = 0 \mid X = \mathbf{x})} = \boldsymbol{w}^\top \mathbf{x}$$

Implies that

$$\mathbb{P}(Y = 1 \mid X = \mathbf{x}) = \sigma(\boldsymbol{w}^\top \mathbf{x})$$

for $\qquad \sigma : z \mapsto \dfrac{1}{1 + e^{-z}},$

the logistic function.



- The logistic function is part of the family of *sigmoid functions*.
- Often called “the” sigmoid function.

Properties:

$$\begin{aligned}
\forall z \in \mathbb{R}, \quad \sigma(-z) \quad &= 1 - \sigma(z), \\
\forall z \in \mathbb{R}, \quad \sigma'(z) \quad &= \sigma(z)(1 - \sigma(z)) \\
&= \sigma(z)\sigma(-z).
\end{aligned}$$

Likelihood for logistic regression

Let $\eta := \sigma(\boldsymbol{w}^\top \mathbf{x} + b)$. W.l.o.g. we assume $b = 0$.
By assumption: $Y|X = \mathbf{x} \sim \text{Ber}(\eta)$.

Likelihood

$$p(Y = y|X = \mathbf{x}) = \eta^y (1-\eta)^{1-y} = \sigma(\boldsymbol{w}^\top \mathbf{x})^y \sigma(-\boldsymbol{w}^\top \mathbf{x})^{1-y}$$

because $1 - \sigma(z) = \sigma(-z)$.

Log-likelihood

$$\begin{aligned}
\ell(\boldsymbol{w}) &= y \log \sigma(\boldsymbol{w}^\top \mathbf{x}) + (1-y) \log \sigma(-\boldsymbol{w}^\top \mathbf{x}) \\
&= y \log \eta + (1-y) \log(1-\eta) \\
&= y \log \frac{\eta}{1-\eta} + \log(1-\eta) \\
&= y \boldsymbol{w}^\top \mathbf{x} + \log \sigma(-\boldsymbol{w}^\top \mathbf{x})
\end{aligned}$$

Maximizing the log-likelihood

Log-likelihood of a sample

Given an i.i.d. training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$

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

The log-likelihood is differentiable and concave.

$\Rightarrow$ Its global maxima are its stationary points.

Gradient of ℓ

$$\begin{aligned} \nabla \ell(\boldsymbol{w}) &= \sum_{i=1}^{n} y_i \mathbf{x}_i - \mathbf{x}_i \frac{\sigma(-\boldsymbol{w}^\top \mathbf{x}_i)\sigma(\boldsymbol{w}^\top \mathbf{x}_i)}{\sigma(-\boldsymbol{w}^\top \mathbf{x}_i)} \qquad \text{since} \quad \sigma'(z) = \sigma(-z)\sigma(z) \\ &= \sum_{i=1}^{n} (y_i - \eta_i)\mathbf{x}_i \qquad \text{with} \qquad \eta_i = \sigma(\boldsymbol{w}^\top \mathbf{x}_i). \end{aligned}$$

Thus, $\quad \nabla \ell(\boldsymbol{w}) = 0 \Leftrightarrow \sum_{i=1}^{n} \mathbf{x}_i (y_i - \sigma(\boldsymbol{w}^\top \mathbf{x}_i)) = 0.$

No closed form solution !

Alternate formulation of logistic regression

If $y \in \{-1, 1\}$, then

$$\mathbb{P}(Y = y | X = \mathbf{x}) = \sigma(y\, \boldsymbol{w}^\top \mathbf{x})$$

Log-likelihood

$$\ell(\boldsymbol{w}) = \log \sigma(y \boldsymbol{w}^\top \mathbf{x}) = -\log\left(1 + \exp(-y \boldsymbol{w}^\top x)\right)$$

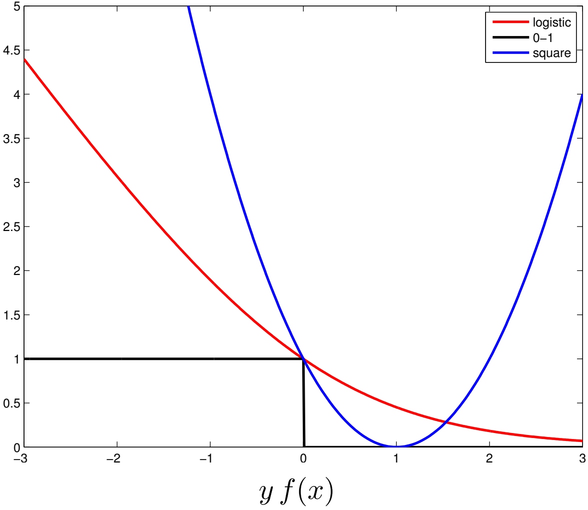
Log-likelihood for a training set

$$\ell(\boldsymbol{w}) = -\sum_{i=1}^{n} \log\left(1 + \exp(-y_i \boldsymbol{w}^\top x_i)\right)$$

The negative log-likelihood takes the form of an empirical risk with loss

$$\ell(a, y) = \log\left(1 + e^{-y\, a}\right)$$

Comparing losses



0-1 loss

$$\ell(f(x), y) = 1_{\{y\,f(x) \leq 0\}}$$

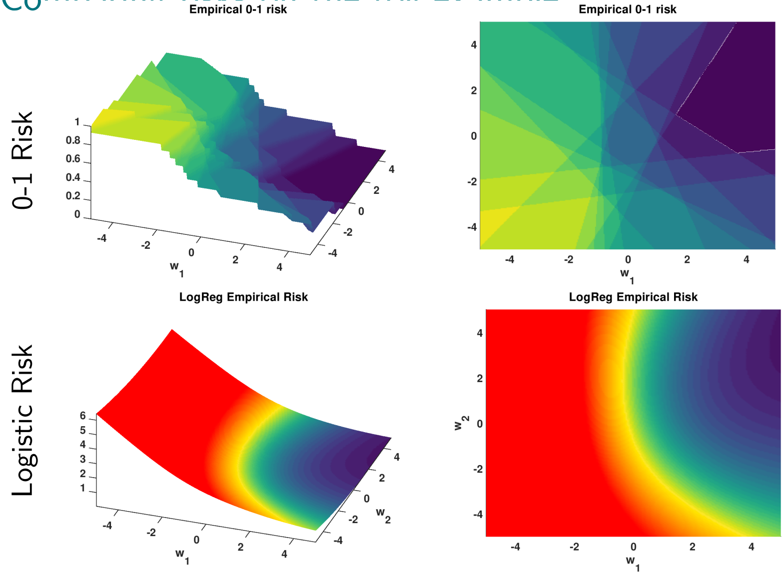
Square loss

$$\ell(f(x), y) = (1 - y\,f(x))^2$$

Logistic loss

$$\frac{\ell(f(x), y)}{\log 2} = \frac{\log(1 + e^{-y\,f(x)})}{\log 2}$$

Comparing risks on the toy example

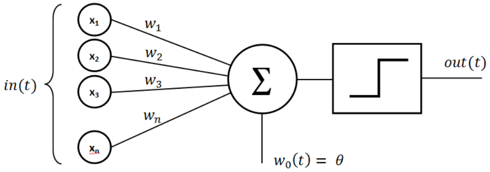


Risks for the 0-1 and logistic loss of the predictor $\widehat{f}(\mathbf{x}) = \boldsymbol{w}^\top \mathbf{x} + b$ as a function of $\boldsymbol{w} = (w_1, w_2)$ for fixed $b = 1$.

Outline

Perceptron (Rosenblatt,1957)

Setting of classification with $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \{-1, 1\}$.



$$\widehat{\gamma}(\mathbf{x}) = \mathrm{sign}(\widehat{f}(\mathbf{x})) \quad \text{with} \quad \widehat{f}(\mathbf{x}) = \boldsymbol{w}^\top \mathbf{x}$$

Perceptron loss:

- If $\mathbf{x}_i$ is well classified pay 0
- If $\mathbf{x}_i$ is miss-classified pay the distance to the classification boundary

Perceptron loss function

Corresponds to using the loss function:

$$\ell(a, y) = \max(-ay, 0)$$

Perceptron algorithm: separable case

Stochastic gradient descent with fixed step-size

repeat
  for $i = 1 \dots n$ **do**
    if $y_i\, \boldsymbol{w}^\top \mathbf{x}_i < 0$ **then**
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \gamma\, y_i \mathbf{x}_i$
    end if
  end for
until all training points well classified

- If the data are separable the algorithm converges in a finite number of steps to a hyperplane that separates them
- The solution found depends on the initialization
- In practice take $\gamma = 1$
- But if the data are not separable the algorithm does not converge

Perceptron algorithm: non separable-case

Stochastic gradient descent with decreasing step size

repeat

$t \to t + 1$

Pick a training pair (x_i, y_i) at random

if $y_i\, \boldsymbol{w}^\top \mathbf{x}_i < 0$ **then**

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \gamma_t\, y_i \mathbf{x}_i$

end if

until $\boldsymbol{w}$ stabilizes

with

$$\sum_t \gamma_t^2 < \infty \qquad \sum_t \gamma_t = \infty.$$

- Always converges
- But slow

Outline

Stochastic gradient descent (SGD) algorithm

Let

- $\mathcal{D}$ be a closed set
- $f : \mathcal{D} \subset \mathbb{R}^p \to \mathbb{R}$ be a differentiable function,
- G a stochastic process defining for all $\theta \in \mathcal{D}$ a r.v. such that $\mathbb{E}[G(\theta)] = \nabla f(\theta)$

e.g., $f(\theta) := \mathcal{R}(h_\theta) = \mathbb{E}[\ell(h_\theta(X), Y)]$ and $G(\theta) = \nabla_\theta\big(\ell(h_\theta(X), Y)\big)$.

Algorithm 1 Projected stochastic gradient descent

1: Initialize θ_0
2: **for** $k = 1$ to $n - 1$ **do**
3: $\quad \theta_k = \Pi_{\mathcal{D}}\big(\theta_{k-1} - \gamma_k\, G_k(\theta_{k-1})\big)$ where $\Pi_{\mathcal{D}}$ is the Euclidean projection on $\mathcal{D}$
4: **end for**
5: **return** $\bar{\theta}_n := \frac{1}{n}\sum_{k=0}^{n-1} \theta_k$ *(Polyak-Ruppert averaging)*

Convergence of the algorithm

- The algorithm is shown to be convergent for $\gamma_k = Ck^{-\alpha}$ with $0 < \alpha < 1$.
- The idea of computing a Cesàro mean $\bar{\theta}_n$ was proposed independently by Polyak et Ruppert in the 80ies and is thus called *Polyak-Ruppert averaging*. They showed it was more efficient to allow θ_k to take larger steps and to stabilize the sequence with averaging.

Theorem (SGD convergence rate)

For $f : \mathcal{D} \subset \mathbb{R}^p \to \mathbb{R}$ convex differentiable, B-Lipschitz with $\mathcal{D}$ a closed convex set such that $\mathcal{D} \subset \{\theta \mid \|\theta\|_2 \leq D\}$, if G is such that $\|G(\theta)\|_2^2 \leq B$ a.s., then for $\gamma_k = \frac{D}{B}\frac{1}{\sqrt{k}}$,

$$\forall \theta_* \in \mathcal{D}, \qquad \mathbb{E}[f(\bar{\theta}_n)] - f(\theta_*) \leq \frac{3BD}{\sqrt{n}}.$$

- If f is strongly convex, and with $\gamma_k = \frac{c}{k}$ for c sufficiently large $\mathbb{E}[f(\theta_n)] - f(\theta_*) = O(\frac{1}{n})$

Applying SGD to learn in supervised learning

Let $H = \{h_\theta \mid \theta \in \Theta\}$ be a hypothesis/predictor set.

Risk minimization

$$f(\theta) := \mathcal{R}(h_\theta) = \mathbb{E}[\ell(h_\theta(X_i), Y_i)].$$

and $G_i(\theta) = \nabla_\theta \ell(h_\theta(X_i), Y_i)$ is a stochastic gradient.

Note that many loss functions are Lipschitz w.r.t. θ (e.g. logistic, perceptron)

Empirical risk minimization

$$f(\theta) := \widehat{\mathcal{R}}_n(h_\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(h_\theta(X_i), Y_i).$$

and $G_i(\theta) = \nabla_\theta \ell(h_\theta(X_i), Y_i)$ is a stochastic gradient.

What is the difference?

SGD on the risk *vs* on the empirical risk

- From the risk, we can only draw n independent stochastic gradients, since we only have a training set of size n.
- From the empirical risk, we can draw as many as we want since the distribution is exactly

$$P_n = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$$

So

- The first pass through the data starts to optimize the risk
- Subsequent passes over the data then optimize the empirical risk (and possibly gradually overfit)
- The best generalization is usually obtained for more than one pass...
- SGD and its fancier cousins Adagrad and Adam are the method of choice if you have a very large dataset.

Outline

Fisher discriminant analysis

Generative classification

$X \in \mathbb{R}^p$ and $Y \in \{0,1\}$. Instead of modeling directly $p(y \mid \mathbf{x})$ model $p(y)$ and $p(\mathbf{x} \mid y)$ and deduce $p(y \mid \mathbf{x})$ using Bayes rule.
In classification $\mathbb{P}(Y = 1 \mid X = \mathbf{x}) =$

$$\frac{\mathbb{P}(X = \mathbf{x} \mid Y = 1)\,\mathbb{P}(Y = 1)}{\mathbb{P}(X = \mathbf{x} \mid Y = 1)\,\mathbb{P}(Y = 1) + \mathbb{P}(X = \mathbf{x} \mid Y = 0)\,\mathbb{P}(Y = 0)}$$

For example one can assume

- $\mathbb{P}(Y = 1) = \pi$
- $\mathbb{P}(X = \mathbf{x} \mid Y = 1) \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$
- $\mathbb{P}(X = \mathbf{x} \mid Y = 0) \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$.

Fisher's discriminant aka Linear Discriminant Analysis (LDA)

Previous model with the constraint $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}$. Given a training set, the different model parameters can be estimated using the maximum likelihood principle, which leads to

$$(\widehat{\pi}, \widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\mu}}_0, \widehat{\boldsymbol{\Sigma}}_1, \widehat{\boldsymbol{\Sigma}}_0).$$

Then we have

$$\begin{aligned}
\mathbb{P}(Y = 1 \mid X = \mathbf{x}) &= \left(1 + \frac{p(\mathbf{x} \mid Y = 0)\,\mathbb{P}(Y = 0)}{p(\mathbf{x} \mid Y = 1)\,\mathbb{P}(Y = 1)}\right)^{-1} \\
&= \left(1 + \frac{1-\pi}{\pi} \frac{\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_0)\right)}{\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_1)\right)}\right)^{-1} \\
&= \left(1 + \exp\left((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} + b\right)\right)^{-1} \\
&= \sigma(\boldsymbol{w}^\top \mathbf{x} + b)
\end{aligned}$$

with $\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ and $b = \log \frac{1-\pi}{\pi} + \frac{1}{2}\boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 - \frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1$.