

Introduction to Machine Learning

Logistic Regression

Department of Statistics – LMU Munich



LOGISTIC REGRESSION

A *discriminant* approach for directly modeling the posterior probabilities $\pi(\mathbf{x})$ of the labels is **logistic regression**.

For now we focus on the binary case $y \in \{0, 1\}$.

A naive approach would be to model

$$\pi(\mathbf{x}) = \mathbb{P}(y = 1 \mid \mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}.$$

Obviously this could result in predicted probabilities $\pi(\mathbf{x}) \notin [0, 1]$.

LOGISTIC REGRESSION

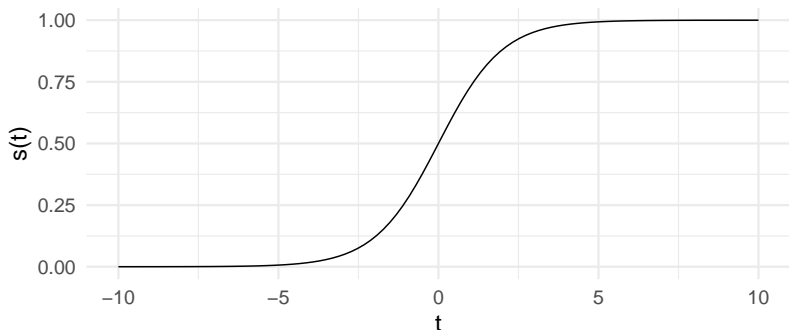
To avoid this, logistic regression “squashes” the estimated linear scores $\theta^\top \mathbf{x}$ to $[0, 1]$ through the **logistic function** s :

$$\pi(\mathbf{x}) = \mathbb{P}(y = 1 \mid \mathbf{x}) = \frac{\exp(\theta^\top \mathbf{x})}{1 + \exp(\theta^\top \mathbf{x})} = \frac{1}{1 + \exp(-\theta^\top \mathbf{x})} = s(\theta^\top \mathbf{x})$$

Note that we will again usually suppress the intercept in notation, i.e., $\theta^\top \mathbf{x} \equiv \theta_0 + \theta^\top \mathbf{x}$.

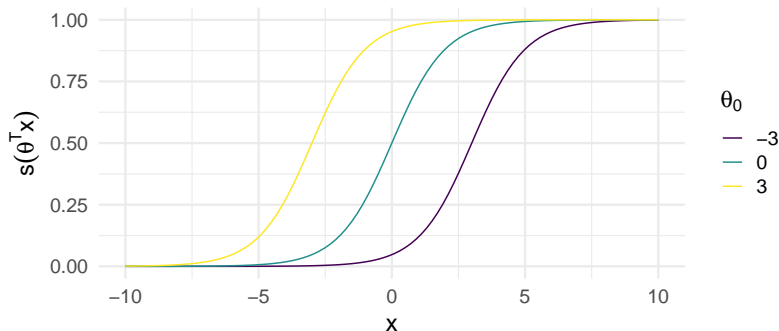
LOGISTIC FUNCTION

The logistic function $s(t) = \frac{\exp(t)}{1+\exp(t)}$ which we use to model the probability $\mathbb{P}(y = 1 \mid \mathbf{x}) = s(\boldsymbol{\theta}^\top \mathbf{x}) = \frac{\exp(\boldsymbol{\theta}^\top \mathbf{x})}{1+\exp(\boldsymbol{\theta}^\top \mathbf{x})}$



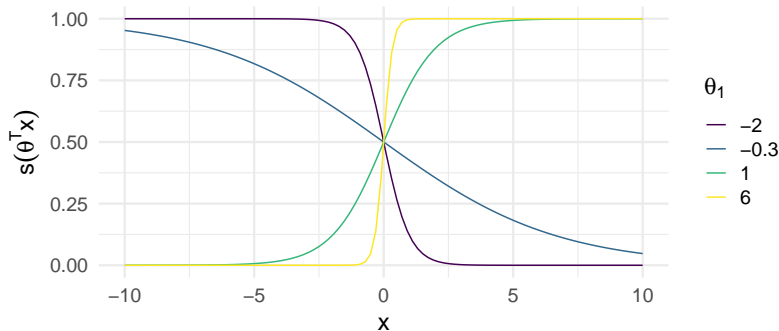
LOGISTIC FUNCTION

Changing the intercept shifts the logistic curve in x-axis direction. Let us assume $\theta_1 = 1$ for simplicity, so that $\mathbb{P}(y = 1 \mid \mathbf{x}) = \frac{\exp(\theta_0 + x)}{1 + \exp(\theta_0 + x)}$



LOGISTIC FUNCTION

Assuming a single feature and no intercept: $\mathbb{P}(y = 1 \mid \mathbf{x}) = \frac{\exp(\theta_1 x_1)}{1 + \exp(\theta_1 x_1)}$:
Parameter θ_1 controls the slope and direction of the logistic curve.



CLASSIFICATION LOSS

In order to find the “optimal” model represented by θ , we need to define a *loss function*.

For a single observation, it makes sense to simply compare the probability implied by the model to the actually observed target variable:

$$\begin{aligned}\text{“accuracy”}^{(i)} &:= \begin{cases} \pi(\mathbf{x}^{(i)}, \theta) & \text{if } y^{(i)} = 1 \\ 1 - \pi(\mathbf{x}^{(i)}, \theta) & \text{if } y^{(i)} = 0 \end{cases} \\ &= \pi(\mathbf{x}^{(i)}, \theta)^{y^{(i)}} (1 - \pi(\mathbf{x}^{(i)}, \theta))^{1-y^{(i)}}\end{aligned}$$

For the entire data set, we combine these predicted probabilities into a joint probability of observing the target vector given the model:

$$\text{“global accuracy”} := \prod_{i=1}^n \pi(\mathbf{x}^{(i)}, \theta)^{y^{(i)}} (1 - \pi(\mathbf{x}^{(i)}, \theta))^{1-y^{(i)}}$$

CLASSIFICATION LOSS

We want a *loss* function, so we actually need the inverse of that:

$$\text{“global inaccuracy”} := \frac{1}{\prod_{i=1}^n \pi(\mathbf{x}^{(i)}, \boldsymbol{\theta})^{y^{(i)}} (1 - \pi(\mathbf{x}^{(i)}, \boldsymbol{\theta}))^{1-y^{(i)}}}$$

Finally, we want the empirical risk to be a *sum* of loss function values, not a *product*

$$\text{recall: } \mathcal{R}_{\text{emp}} = \sum_{i=1}^n L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right)$$

so we turn the product into a sum by taking its log – the same parameters minimize this, which is all we care about, and we end up with the **logistic** or **cross entropy loss function**:

$$\begin{aligned} L(y, f(\mathbf{x})) &= -y \log[\pi(\mathbf{x})] - (1 - y) \log[1 - \pi(\mathbf{x})] \\ &= -y \boldsymbol{\theta}^\top \mathbf{x} + \log[1 + \exp(\boldsymbol{\theta}^\top \mathbf{x})] \end{aligned}$$

CLASSIFICATION LOSS

Remember that $\log[\pi(\mathbf{x})] = -\log[1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x})]$.

For $y = 0$ and $y = 1$ and $f(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$, this yields:

$$y = 0 \quad : \quad \log[1 + \exp(f(\mathbf{x}))]$$

$$y = 1 \quad : \quad \log[1 + \exp(-f(\mathbf{x}))]$$

If we encode the labels with $\mathcal{Y} = \{-1, +1\}$ instead, we can simplify this as:

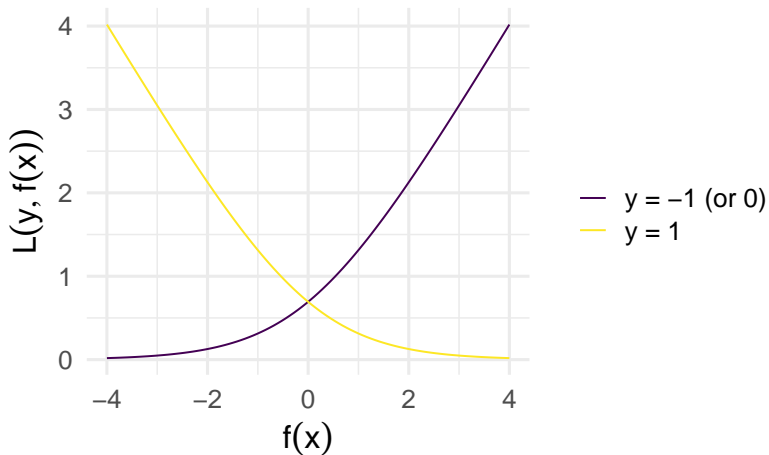
$$L(y, f(\mathbf{x})) = \log[1 + \exp(-yf(\mathbf{x}))]$$

This is called **Bernoulli loss**.

Logistic regression minimizes this, and we can use these loss functions for any other discriminative classification model which directly models $f(\mathbf{x})$.

CLASSIFICATION LOSS

Bernoulli / Logistic Loss:



CLASSIFICATION LOSS

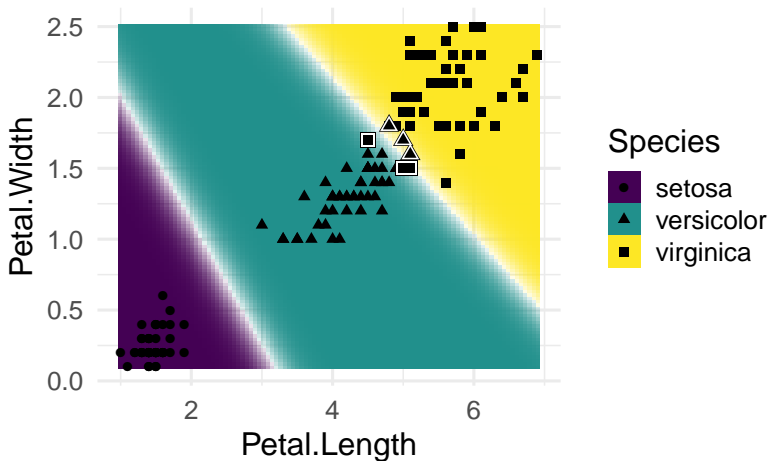
In order to minimize the loss (misclassification), we should predict $y = 1$ if

$$\pi(\mathbf{x}, \boldsymbol{\theta}) = \mathbb{P}(y = 1 | \mathbf{x}, \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}^\top \mathbf{x})}{1 + \exp(\boldsymbol{\theta}^\top \mathbf{x})} \geq 0.5,$$

which is equivalent to

$$\boldsymbol{\theta}^\top \mathbf{x} \geq 0 \implies y = 1.$$

ESTIMATING PROBABILITIES - MULTINOMIAL REGRESSION AND SOFTMAX



ESTIMATING PROBABILITIES - MULTINOMIAL REGRESSION AND SOFTMAX

For a categorical response variable $y \in \{1, \dots, g\}$ with $g > 2$ the model extends to

$$\pi_k(\mathbf{x}) = \mathbb{P}(y = k \mid \mathbf{x}) = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x})}{\sum_{j=1}^g \exp(\boldsymbol{\theta}_j^\top \mathbf{x})}.$$

The latter function is called the *softmax*, and defined on a numerical vector \mathbf{z} :

$$s(\mathbf{z})_k = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

This is a generalization of the logistic function (check for $g=2$). It “squashes” a g -dimensional real-valued vector \mathbf{z} to a vector of the same dimension, with every entry in the range $[0, 1]$ and all entries adding up to 1.

ESTIMATING PROBABILITIES - MULTINOMIAL REGRESSION AND SOFTMAX

By comparing the posterior probabilities of two categories k and l we end up in a linear function (in \mathbf{x}),

$$\log \frac{\pi_k(\mathbf{x})}{\pi_l(\mathbf{x})} = (\boldsymbol{\theta}_k - \boldsymbol{\theta}_l)^\top \mathbf{x}.$$

The class boundaries lie where these (linear) functions are zero, i.e., where the predicted class probabilities are equal.

Remark:

- $\boldsymbol{\theta}_j$ are vectors here.
- Well-definedness: $\pi_k(\mathbf{x}) \in [0, 1]$ and $\sum_k \pi_k(\mathbf{x}) = 1$

ESTIMATING PROBABILITIES - MULTINOMIAL REGRESSION AND SOFTMAX

This approach can be extended in exactly the same fashion for other score based models. For discriminating each class k from all others, we define a binary score model $f_k(\mathbf{x})$ with parameter vector θ_k . We then combine these models through the softmax function

$$\mathbb{P}(y = k \mid \mathbf{x}) = \pi_k(\mathbf{x}) = s_k(f_1(\mathbf{x}), \dots, f_g(\mathbf{x}))$$

and optimize all parameter vectors of the f_k jointly.

ESTIMATING PROBABILITIES - MULTINOMIAL REGRESSION AND SOFTMAX

Further comments:

- For linear $f(\mathbf{x} \mid \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}$, this is also called *softmax regression*. (Note that x can include derived features like polynomials or interactions as well.)
- One set of parameters is “redundant”: If we subtract any fixed vector from all $\boldsymbol{\theta}_k$, the predictions do not change. The model is “overparameterized”, the minimizer of $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$ is not unique. Hence, we set $\boldsymbol{\theta}_g = (0, \dots, 0)$ and only optimize the other $\boldsymbol{\theta}_k$, $k = 1, \dots, g - 1$.
(Compare: logistic regression for binary classification also has only *one* parameter vector for discriminating between two classes...).
- A similar approach can be used for many ML models: multiclass LDA, naive Bayes, neural networks and boosting.

LOGISTIC AND SOFTMAX REGRESSION

Representation: Design matrix \mathbf{X} , coefficients θ .

Evaluation: Logistic/Bernoulli loss function.

Optimization: Numerical optimization, typically gradient descent based methods.