Classification problem

Two kinds of supervised learning problems: *regression* and *classification*.

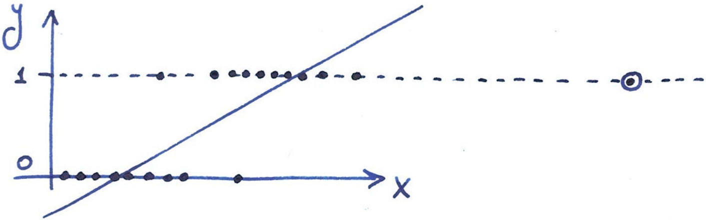
In classification, each sample is assigned to one of the several classes, i.e. the response variable y is *categorical*.

Classification problems can be *binary* or *multinomial* (*multiclass*).

It is convenient to denote $y \in \{0, 1\}$ for binary classification and $y \in \{0, 1, \ldots, K-1\}$ for multiclass classification with K classes.

We will deal the binary classification first.

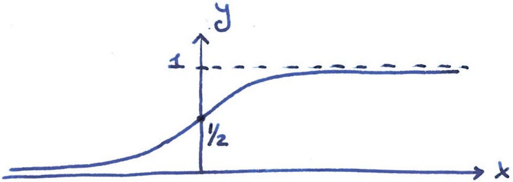
Why not use linear regression?



We want the prediction $\hat{y}$ to be in $[0, 1]$.

Predicting probabilities

We want $\hat{y} \in [0, 1]$ and not in $\{0, 1\}$ because we will interpret it as a probability $P(y = 1)$. Predicting probabilities is arguably more useful and meaningful than predicting class membership.



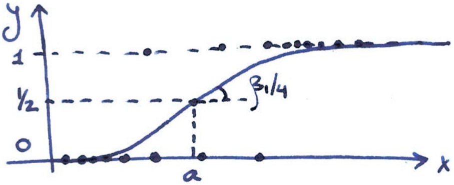
Logistic (sigmoid) function:

$$g(x) = \frac{1}{1 + e^{-x}}.$$

Logistic regression parameters

For a single predictor x:

$$h(x) = g(\beta_0 + \beta_1 x) = g\big(\beta_1 (x - a)\big) = \frac{1}{1 + e^{-\beta_1 (x-a)}}.$$



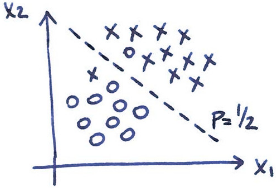
For multiple predictors:

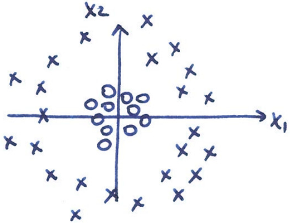
$$h(x) = g(\boldsymbol{\beta}^\top \mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^\top \mathbf{x}}}.$$

Linear classifier

Probability (and/or class) are predicted based on a linear function of the predictors.

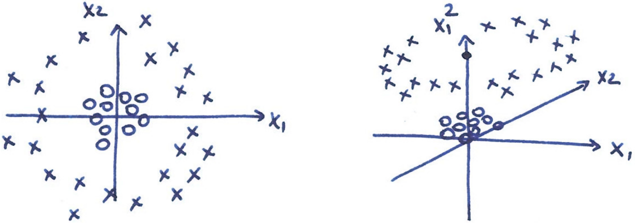
Decision boundary is linear.





Linear classifier with polynomial features

Adding more features to the model (e.g. x_1^2, x_2^2, x_1x_2, etc.) can make a problem solvable by a linear classifier.



Loss function

We have

$$\hat{y} = P(y = 1) = h(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^\top \mathbf{x}}}.$$

Can we still use the mean squared error (MSE) loss function?

$$\mathcal{L} = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$$

It is not ideal for probabilistic predictions: $\hat{y}_i = 0.99$ and $\hat{y}_i = 0.9999$ have similar loss when $y_i = 0$.

Recall that MSE followed from the assumption of Gaussian noise using maximum likelihood (Lecture 3). What is the noise here?

Maximum likelihood

Bernoulli random variable: $Y = 1$ with probability p and $Y = 0$ with probability $1 - p$.

The likelihood:

$$\prod_{i|y_i=1} h(\mathbf{x}_i) \cdot \prod_{i|y_i=0} \big(1 - h(\mathbf{x}_i)\big).$$

The negative log-likelihood:

$$\begin{aligned}\mathcal{L} &= -\sum_{i|y_i=1} \log h(\mathbf{x}_i) - \sum_{i|y_i=0} \log\big(1 - h(\mathbf{x}_i)\big) \\ &= -\sum_i \Big[y_i \log h(\mathbf{x}_i) + (1 - y_i) \log\big(1 - h(\mathbf{x}_i)\big)\Big].\end{aligned}$$

Logistic regression

The loss function is

$$\mathcal{L} = -\sum_i \Big[y_i \log h(\mathbf{x}_i) + (1 - y_i) \log \big(1 - h(\mathbf{x}_i)\big) \Big]$$

where

$$h(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^\top \mathbf{x}}}.$$

This is not a linear model! It is a *generalized linear model* (GLM).

Good news: this loss function is convex. Bad news: the minimum is not available in closed form.

Usually logistic regression is optimized using 2nd order methods (Newton's method), but for simplicity we will consider gradient descent.

Gradient descent for logistic regression

Exercise: $g'(x) = g(x)(1 - g(x))$.

From here we get:

$$\nabla \log h(\mathbf{x}) = \nabla \log g(\boldsymbol{\beta}^\top \mathbf{x}) = (1 - h(\mathbf{x}))\mathbf{x},$$
$$\nabla \log (1 - h(\mathbf{x})) = \nabla \log (1 - g(\boldsymbol{\beta}^\top \mathbf{x})) = -h(\mathbf{x})\mathbf{x}.$$

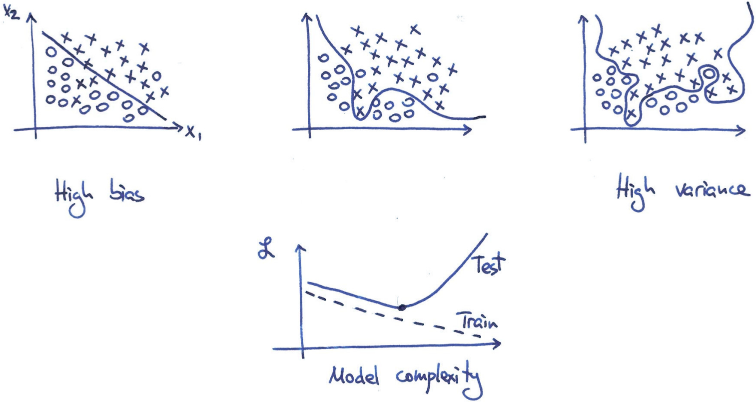
And finally:

$$\begin{aligned}\nabla \mathcal{L} &= -\nabla \sum_i \Big[y_i \log h(\mathbf{x}_i) + (1 - y_i) \log (1 - h(\mathbf{x}_i)) \Big] \\ &= -\sum_i (y_i - h(\mathbf{x}_i))\mathbf{x}_i = -\mathbf{X}^\top(\mathbf{y} - \hat{\mathbf{y}}).\end{aligned}$$

Amazingly, this formula is identical to $\nabla \mathcal{L}$ in linear regression (Lecture 2), up to a constant factor. This hints at how useful the GLM framework is.

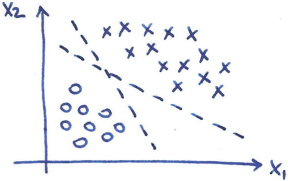
Overfitting and regularization

Overfitting, bias–variance tradeoff, regularization (ridge/lasso) — all these concepts still apply.



Perfect separation

In linear regression in the *interpolation* regime, training loss is zero and there are many $\hat{\beta}$ solutions (Lecture 4). Minimum-norm solution can in some cases perform well.



In logistic regression in the *perfect separation* regime, training loss converges to zero but $\hat{\beta}$ diverges to infinity. It may still in some cases perform well in terms of class predictions, but probabilities may be miscalibrated (all $\hat{y} \to 0$ or 1).

From probabilities to binary predictions

Logistic regression is designed to give accurate probabilistic predictions. A *cutoff* (threshold) is needed to convert them into binary predictions.

One may want to choose a cutoff to maximize the *accuracy*. But it may also be that errors in certain direction (0 or 1) are preferable. One should think about true positive vs. false positive rates.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

A note on accuracy and class imbalance

Accuracy can be very misleading if the classes are *unbalanced*.

But logistic regression still works fine. (Assuming that the class imbalance is the same in the training and in the test data.)

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Multinomial logistic regression

One way to generalize logistic regression to multinomial case is via a *softmax* function.

Each class $i = \{0, 1, \dots, K-1\}$ gets its own $\boldsymbol{\beta}_i$. Then

$$P(y = k) = \frac{e^{\boldsymbol{\beta}_k^\top \mathbf{x}}}{\sum_i e^{\boldsymbol{\beta}_i^\top \mathbf{x}}}.$$

Note that adding any vector $\boldsymbol{\psi}$ to all $\boldsymbol{\beta}_i$ vectors will not change the probabilities. So we can e.g. constrain $\boldsymbol{\beta}_{K-1} = \mathbf{0}$. For a binary problem this becomes equivalent to logistic regression.