Mathematical techniques in data science

Lecture 4: Logistic Regression

DSI Competition

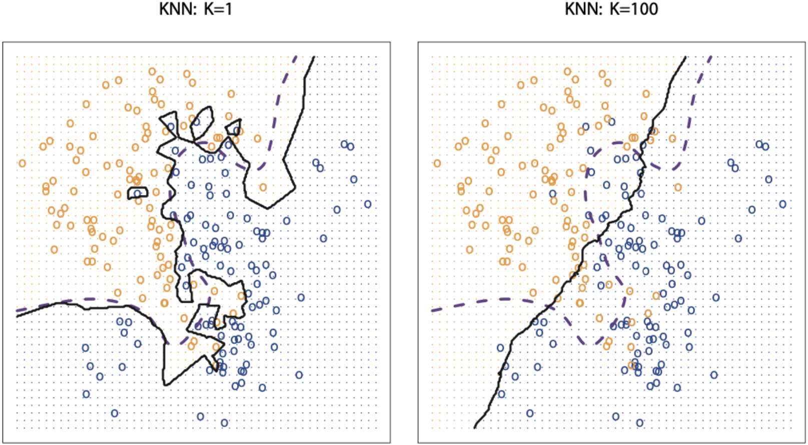
Link for DSI Association Competition
(Prize: Free Google COLAB Subscription!)

Last lecture: Nearest Neighbors

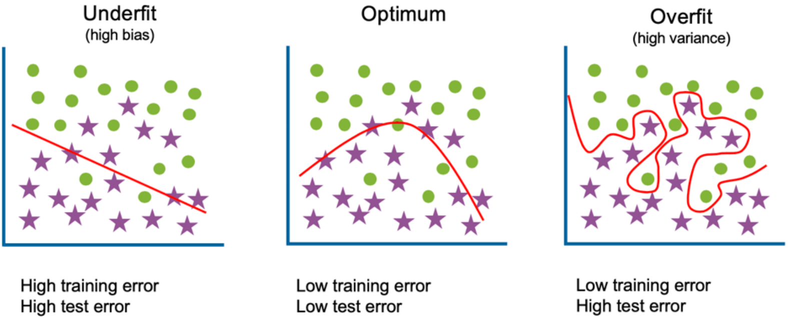
General steps to build ML models

- Get and pre-process data
- Visualize the data (optional)
- Split data into training/test sets
- Create a model
- Train the model on training set; i.e. call model.fit()
- Predict on test data
- Compute evaluation metrics (accuracy, mean squared error, etc.)
- Visualize the trained model (optional)

Underfiting/Overfitting

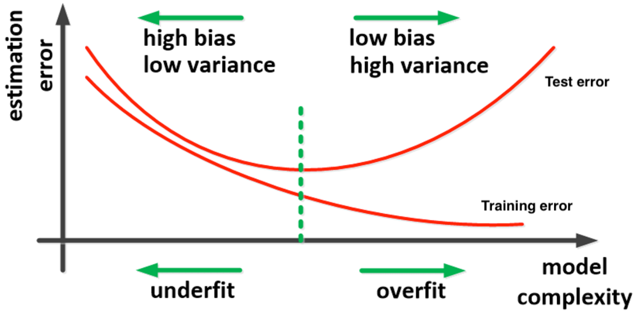


Underfiting/Overfitting



(Source: IBM)

Underfiting/Overfitting



Nearest neighbors: pros and cons

Pros:

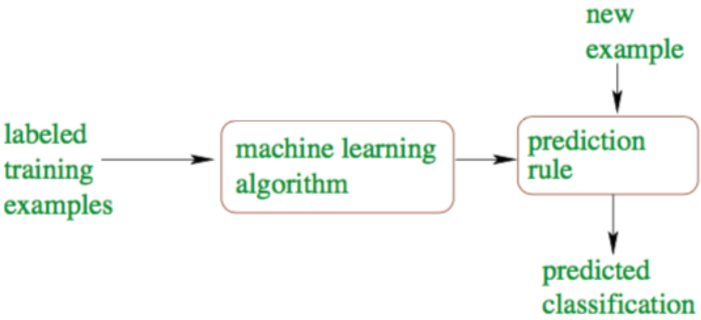
- Simple algorithm
- Easy to implement, no training required
- Can learn complex target function

Cons:

- Prediction is slow
- Don't work well with high-dimensional inputs (e.g., more than 20 features)

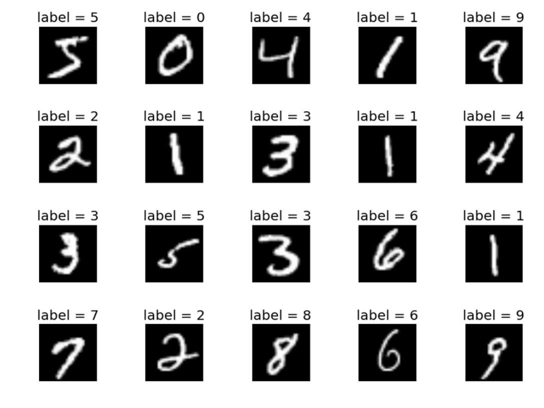
Logistic regression

Supervised learning



Learning a function that maps an input to an output based on example input-output pairs

Hand-written digit recognition (MNIST dataset)



Classification algorithms

- Logistic regression
- Linear Discriminant Analysis
- Support Vector Machines
- Nearest neighbours

Logistic regression

- Despite the name "regression", is a classifier
- Only for binary classification
- Data point $(\mathbf{x}, y)$ where
  - $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ is a vector with d features
  - y is the label (0 or 1)
- Logistic regression models $P[y = 1|X = \mathbf{x}]$
- Then

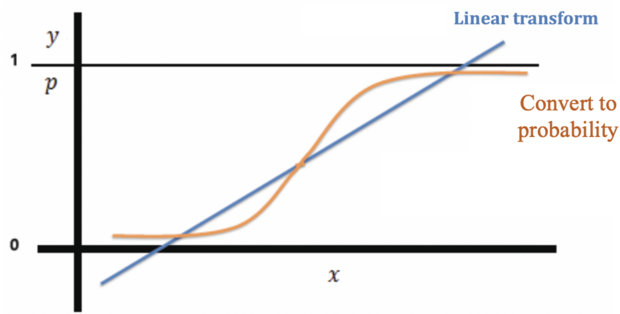
$$P[y = 0|X = \mathbf{x}] = 1 - P[y = 1|X = \mathbf{x}]$$

Logistic regression

Linear transform

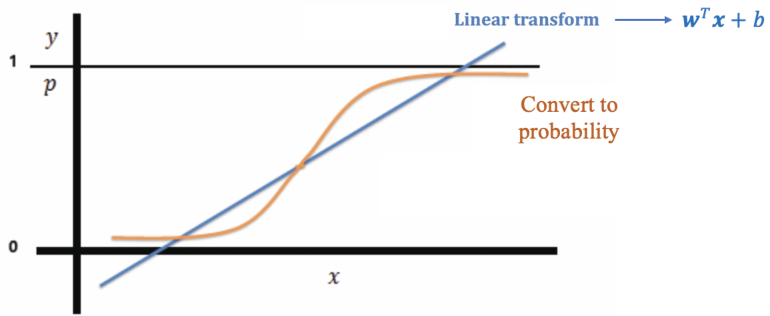
Convert to probability

$\boldsymbol{x} \longrightarrow \; ? \; \longrightarrow P(y = 1|\boldsymbol{x})$

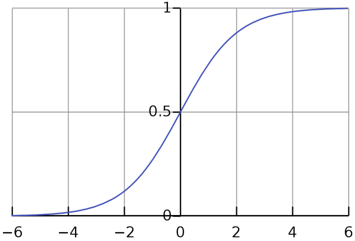


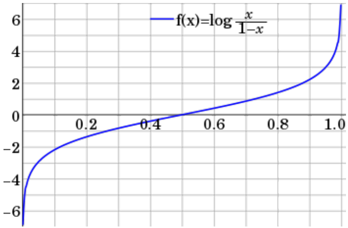
Logistic regression

$$\boldsymbol{x} \xrightarrow{\text{Linear transform}} \boldsymbol{w}^T\boldsymbol{x} + b \xrightarrow{\text{Convert to probability}} P(y = 1|\boldsymbol{x})$$



Transformation between $(-\infty, \infty)$ and $[0, 1]$



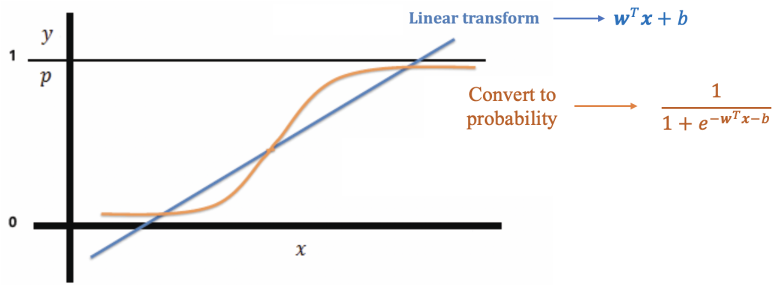


$$f(x) = \frac{e^x}{1+e^x}$$

$$logit(p) = \log \frac{p}{1-p}$$

Logistic regression

$$\boldsymbol{x} \xrightarrow{\text{Linear transform}} \boldsymbol{w}^T\boldsymbol{x} + b \xrightarrow{\text{Convert to probability}} \frac{1}{1 + e^{-\boldsymbol{w}^T\boldsymbol{x} - b}}$$



Logistic regression

- Model: Given $X = \mathbf{x}$, Y is a Bernoulli random variable with parameter $p(\mathbf{x}) = P[Y = 1|X = \mathbf{x}]$ and

$$logit(p(\mathbf{x})) = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d$$

  for some vector $\beta = (\beta_0, \beta_1, \ldots, \beta_d) \in \mathbb{R}^{d+1}$.
- Goal: Find $\hat{\beta}$ that best "fits" the data

To review

- Probability/Statistics
  - Independence
  - Bernoulli random variables
  - Maximum-likelihood (ML) estimation
- Calculus
  - Partial derivatives
  - Finding critical points of a function

Parameter estimation

- Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, we have
- For a Bernoulli r.v. with parameter p

$$P[Y = y] = p^y (1 - p)^{1-y}, \quad y \in \{0, 1\}$$

- Likelihood of the parameter (probability of the dataset):

$$L(\beta) = \prod_{i=1}^{n} p(\mathbf{x}_i, \beta)^{y_i} (1 - p(\mathbf{x}_i, \beta))^{1-y_i}$$

Parameter estimation: maximum likelihood

- The log-likelihood can be computed as

$$\begin{aligned}\ell(\beta) &= \log L(\beta) \\ &= \sum_{i=1}^{n} \left[y_i \log p(\mathbf{x}_i, \beta) + (1 - y_i) \log(1 - p(\mathbf{x}_i, \beta)) \right]\end{aligned}$$

- Maximize $\ell(\beta)$ to find $\beta \rightarrow$ the maximum-likelihood method
- The term

$$-[y \log(p) + (1 - y) \log(1 - p)]$$

is known in the field as the log-loss, or the binary cross-entropy loss

Logistic regression: estimating the parameter

- The optimization needs to be performed by a numerical optimization method
- Penalties can be added to regularize the problem to avoid overfitting

$$\max_{\beta} \ell(\beta) - \frac{1}{C} \sum_i |\beta_i|$$

or

$$\min_{\beta} -\ell(\beta) + \frac{1}{C} \sum_i |\beta_i|^2$$

Logistic regression with more than 2 classes

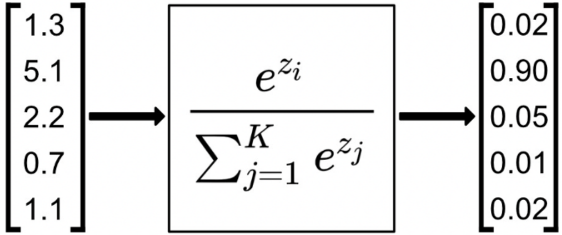
- Suppose now the response can take any of $\{1, \ldots, K\}$ values
- We use the categorical distribution instead of the Bernoulli distribution

$$P[Y = k | X = \mathbf{x}] = p_k(\mathbf{x}), \quad \sum_{k=1}^{K} p_k(\mathbf{x}) = 1.$$

- Model

$$p_k(\mathbf{x}) = \frac{e^{w_k^T \mathbf{x}_k + b_k}}{\sum_{k=1}^{K} e^{w_k^T \mathbf{x}_k + b_k}}$$

Softmax function



Logistic regression: pros and cons

Pros:

- Simple algorithm
- Prediction is fast
- Easy to implement
- The forward map has a closed-form formula of the derivatives

$$\frac{\partial \ell}{\partial \beta_j}(\beta) = \sum_{i=1}^{n} \left[y_i x_{ij} - x_{ij} \frac{e^{x_i^T \beta}}{1 + e^{x_i^T \beta}} \right].$$

Cons:

- Linear model

How to make logistic regression better?

We want a model that

- compute the derivatives (of the objective function, with respect to the parameters) easily
- can capture complex relationships

This is difficult because complex models often have high numbers of parameters and don't have closed-form derivatives, and computations of

$$\frac{\partial \ell}{\partial \beta_i}(x) \approx \frac{\ell(x+\epsilon_i) - \ell(x)}{\epsilon_i}$$

are large (and unstable)

How to make logistic regression better?

- Automatic differentiation and back-propagation
- Ideas:
  - Organizing information using graphs (networks)
  - Chain rule

$$(f \circ g)'(x) = f'(g(x))g'(x)$$