Modul 215: Machine Learning

Linear Regression Methods: Linear & Logistic Regression

Wintersemester 2023/24

Tobias Kortus

Center for Technology and Transfer (ZTT)
University of Applied Sciences

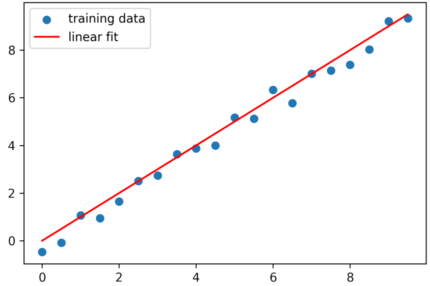
November 9, 2023

Recap: Regression

- **Goal:** Find a function f (in most cases a set of parameters) to predict a quantity y for input x by using training data:

$$\left((x^{(1)}, y^{(1)}), \ldots, (y^{(N)}, x^{(N)})\right) \tag{1}$$

- where $(x^{(i)}, y^{(i)})$ is one data sample
- e.g (kaggle.com):
  1. House prices (given location, size, etc.)
  2. Sales prediction (given advertisement)
  3. $C0_2$ emission (given engine size, etc.)
  4. Diabetes (disease progression)
  5. ...



Linear Regression

- **Goal**: Model the relationship between input and target by a linear equation:

$$\hat{y} = f(x_1, \dots, x_p) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j \tag{2}$$

- $\hat{\beta}_0$: Intercept, bias
- $\hat{\beta}_j$: Slope, weight for dimension j
- $\hat{y}$: Prediction for target y given input x
- Multiple approaches available to fit a linear model e.g.:
  1. Closed form solution: ordinary least squares
  2. Locally weighted linear regression
  3. Gradient descent
  4. ...

Ordinary Least Squares (OLS): Residual Sum of Squares

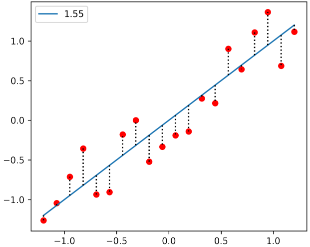
- **Goal:** find a function f that minimizes residual sum of squares:

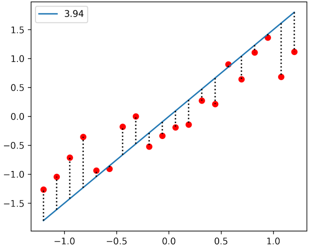
$$RSS(\hat{\beta}) = \sum_{i=1}^{N} \Big(\underbrace{y^{(i)} - f(x^{(i)})}_{residual} \Big)^2 \tag{3}$$

- with

$$f(x_1^{(i)}, \dots x_p^{(i)}) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j^{(i)} \tag{4}$$

- with target value $y^{(i)}$ for input $x^{(i)}$ and model parameters $\hat{\beta} = \{\hat{\beta}_0, \{\hat{\beta}_j\}_{1:p}\}$





Ordinary Least Squares (OLS): Matrix Representations I

- We can replace the sum of $x_j\hat{\beta}_j$ terms (linear combination) with inner product of vector x^T and $\hat{\beta}$:

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} x_j\hat{\beta}_j = \hat{\beta}_0 + \underbrace{x_1\hat{\beta}_1 + \cdots + x_j\hat{\beta}_j}_{\begin{pmatrix} x_1 & \dots & x_j \end{pmatrix}\begin{pmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_j \end{pmatrix}} \tag{5}$$

- Which gives us:

$$\hat{y} = \hat{\beta}_0 + x^T\hat{\beta} \tag{6}$$

Ordinary Least Squares (OLS): Matrix Representations II

- Further, we can combine:

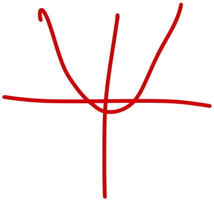
$$1\,\hat{\beta}_0 + x'^T\hat{\beta} = \begin{pmatrix} 1 & x_1 & \dots & x_j \end{pmatrix} \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_j \end{pmatrix} \tag{7}$$

- Which gives us the following expression for multiple datapoints:

$$\hat{y} = \mathbf{X}'^T\hat{\beta} \tag{8}$$

- where $\mathbf{X}'$ is a $N\times$ (p+1) matrix containing N feature vectors, $\hat{\beta}$ is a vector containing all parameters and $\hat{y}$ is a vector containing all predictions

Ordinary Least Square (OLS): Closed Form Solution



- Now we can reformulate the RSS objective as:

$$RSS(\hat{\beta}) = \left(y - \mathbf{X}'^T\hat{\beta}\right)^T \left(y - \mathbf{X}'^T\hat{\beta}\right) \tag{9}$$

- By differentiating $RSS(\hat{\beta})$ w.r.t. $\hat{\beta}$, setting the equation to zero, it gives us an unique solution for $\hat{\beta}$ ($\mathbf{X}'^T\mathbf{X}'$) must be invertible):

$$\hat{\beta} = \left(\mathbf{X}'^T\mathbf{X}'\right)^{-1}\mathbf{X}'^T y \tag{10}$$

Ordinary Least Square (OLS): Solving OLS Manually I

- Consider the following dataset $\mathcal{D} = \{(-1,-2),(1,0),(2,1)\}$ where each tuple $(x^{(i)}, y^{(i)})$ consists of a one-dimensional (scalar) feature $x^{(i)}$ and a scalar target $y^{(i)}$. Find the parameters β for a linear regression model using OLS.

$$X = \begin{pmatrix} -1 \\ 1 \\ 2 \end{pmatrix} \quad y = \begin{pmatrix} -2 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} = X' \quad AA^{-1} = I$$

$$\begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}$$

$$X' = \begin{pmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \quad X'^T = \begin{pmatrix} 1 & 1 & 1 \\ -1 & 1 & 2 \end{pmatrix}$$

$$X'^T X' = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} = A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad A^{-1} = \frac{1}{a \cdot d - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

$$= \frac{1}{14} \begin{pmatrix} 6 & -2 \\ -2 & 3 \end{pmatrix}$$

Ordinary Least Square (OLS): Solving OLS Manually II

$$X^T y = \begin{pmatrix} -1 \\ 4 \end{pmatrix}$$

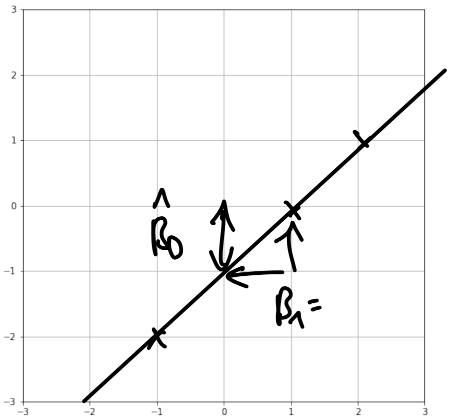
$$A^{-1} X^T y = \frac{1}{14} \begin{pmatrix} 6 & -2 \\ -2 & 3 \end{pmatrix} \begin{pmatrix} -1 \\ 4 \end{pmatrix}$$

$$= 1/14 \begin{pmatrix} -14 \\ 14 \end{pmatrix}$$

$$\hat{\beta} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

Ordinary Least Square (OLS): Solving OLS Manually III

$$\hat{\beta} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$



Ordinary Least Square (OLS): Scikit Learn I

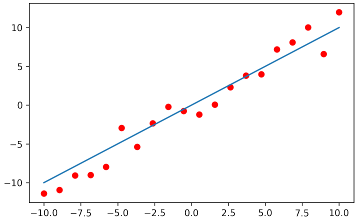
sklearn.linear_model.LinearRegression

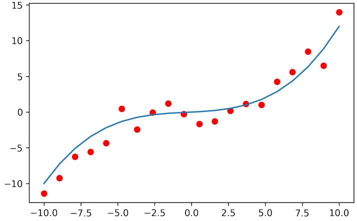
class sklearn.linear_model.LinearRegression(*, *fit_intercept=True*, *normalize='deprecated'*, *copy_X=True*, *n_jobs=None*, *positive=False*) [source]

- fit_intercept (bool): Use parameter β_0 during data fit. (Recommended in most use-cases, data must be centered otherwise)
- normalize: $\rightarrow$ will be removed in v1.2
- copy_X: if true, X will be copied (otherwise it might be overwritten)
- n_jobs: The number of jobs to use for the computation. Requires sufficiently large task.
- positive: When set to True, forces the coefficients to be positive.

Going Beyond Linear Functions: Nonlinear Features I

- In reality observed data is often not linear (e.g. quadratic, polynomial, etc.) $\rightarrow$ we still want to use our linear model
- How to extend our previous OLS framework to deal with this kind of data?





Going Beyond Linear Functions: Nonlinear Features II (Polynomial)

- **Single feature x with polynomial target y:**

  Suppose: $\mathcal{D} = \{(x^{(j)}, y^{(j)})\}_{1:N}$ $\qquad$ $y(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$

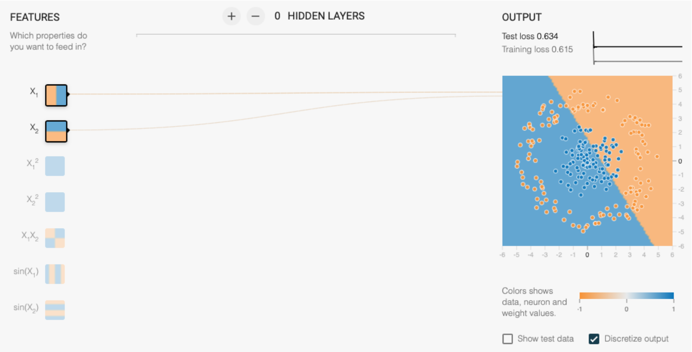
- **Polynomial features x with linear target y:**

  Now: $\mathcal{D} = \{([x^{(j)}, (x^{(j)})^2, (x^{(j)})^3], y^{(j)})\}_{1:N}$ $\qquad$ $y(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$

- Data is now linear separable in higher dimensional feature space
- General notation: "polynomial feature transform" $\phi(x) = [1, x, x^2, x^3, \dots]$
- Can be expanded to any kind of nonlinear transformation (e.g sin, cosine, sqrt)

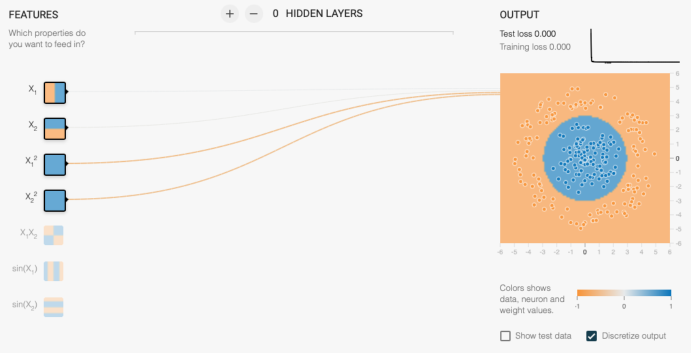
Tensorflow Playground[1] - Polynomial Features I

- **Usage**: Taking only input feature without hidden dimension and linear activation function corresponds to linear model optimized using gradient descent.



[1] http://playground.tensorflow.org/

- **Summary**: By using nonlinear transformation X_1^2, X_2^2 we can fit our dataset without any modification of the linear model.



- **Consider**:

$$y = \beta + \beta_1 x_1 + \beta_2 x_2 \tag{11}$$

- **Following**:

$$\frac{\partial y}{\partial x_1} = \beta_1 \tag{12}$$

- This means changing x_1 by one unit will change y by β_1 $\rightarrow$ we cannot capture any dependence on x_2 (independent variables)

Exclusive-Or Gate: XOR

Input		Output
x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Going Beyond Linear Functions: Interactions II

- **In Contrast**:

$$y = \beta + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 \tag{13}$$

- **Following**:

$$\frac{\partial y}{\partial x_1} = \beta_1 + \beta_3 X_2 \tag{14}$$

- Modeled interactions (combinations) of x_1 with x_2 captures dependencies between those two variables

Exclusive-Or Gate: XOR

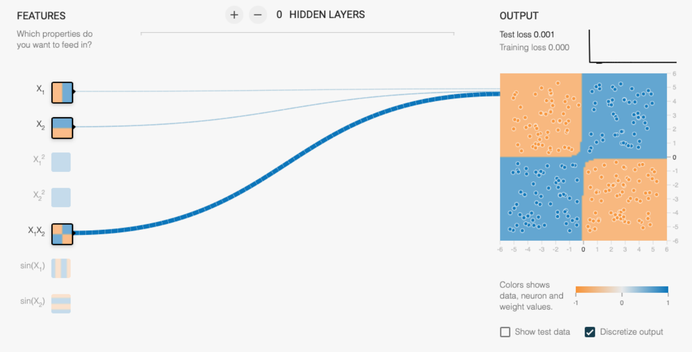
Input		Output
x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Tensorflow Playgroud - Interactions I

- **Recap**: Taking only input feature without hidden dimension and linear activation function corresponds to linear model optimized using gradient descent.

Tensorflow Playground - Interactions II

- **Summary**: By using interaction term X_1X_2 we can fit XOR dataset without any modification of the linear model.



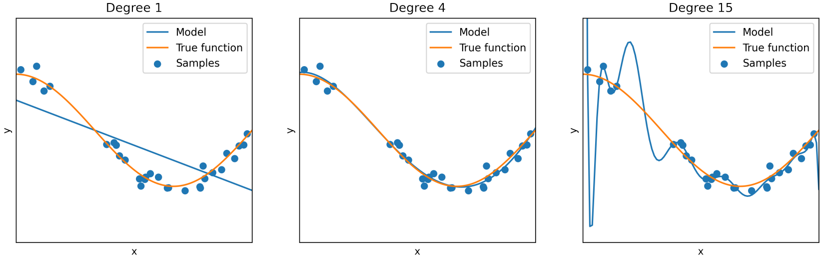
Going Beyond Linear Functions: Scikit-Learn

sklearn.preprocessing.PolynomialFeatures

class sklearn.preprocessing.PolynomialFeatures(*degree=2, *, interaction_only=False, include_bias=True, order='C'*) [source]

- degree (int): Maximal degree of the polynomial features (e.g. x_1, x_2 with degree=2: $[x_1, x_2, x_1^2, x_2^2, x_1 x_2]$)
- intearaction_only (bool): If True, only interaction features are produced.
- include_bias (bool): If True (default), then include a bias column.

Going Beyond Linear Functions: Final Notes



Parametric vs. Nonparametric Learning Algorithms

- **"Parametric" learning algorithm**: fit a fixed set of parameters to data $\leftarrow$ e.g. Linear Regression, Neural Networks, Naive Bayes
- **"Nonparametric" learning algorithm**: amount of data/parameters you need to keep grows (linearly) w.r.t size of data $\rightarrow$ **do not make strong assumptions about the form of the mapping function** (stay flexible). $\leftarrow$ e.g. Locally Weighted Regression,

Going Beyond Linear Functions: Locally Weighted Linear Regression

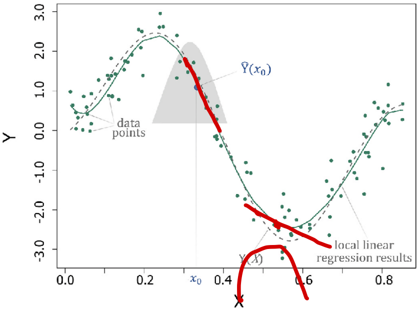
- Find a set of parameter β' for each point in test set that minimizes weighted sum of squares:

$$WRSS(\hat{\beta}) = \sum_{i}^{m} w^{(i)} \left(y^{(i)} - \hat{\beta}^T x^{(i)} \right)^2 \tag{15}$$

- where w is a non-negative weight (scaling factor) with

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2} \right) \tag{16}$$

- Requires no training phase. Instead coefficients are calculated for each sample during prediction.



Summary: Linear Regression

- Ordinary Least Squares Regression
  - RSS objective and closed form solution for OLS
  - How to solve linear regression problems by hand
- Nonlinear features
  - Polynomial features + how they solve the nonlinearity problem
  - Interactions + how they work
  - Overadaption of linear models to highly nonlinear features
- Locally Weighted Linear Regression
  - Non-parametric machine learning model
  - Requires no specific assumption about function mapping

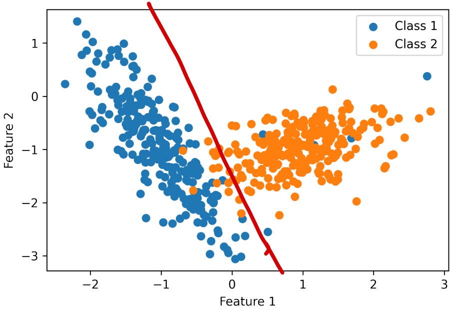
Any Questions ?

Recap: Classification

- **Goal:** Find a function f that divides the input space (feature space) X into decision regions separating all classes contained in Y using data $\rightarrow$ discrete outcome:

$$(y^{(1)}, x^{(1)}), \ldots \ldots, (y^{(N)}, x^{(N)}) \qquad (17)$$

- with $y^{(i)} \in \mathbb{N}$ and $x^{(i)} \in \mathbb{R}^n$
- e.g (kaggle.com):
  1. Optical character recognition (given image of character)
  2. Heart attack classification (given ECG)
  3. Skin cancer classification (given skin image)
  4. ...



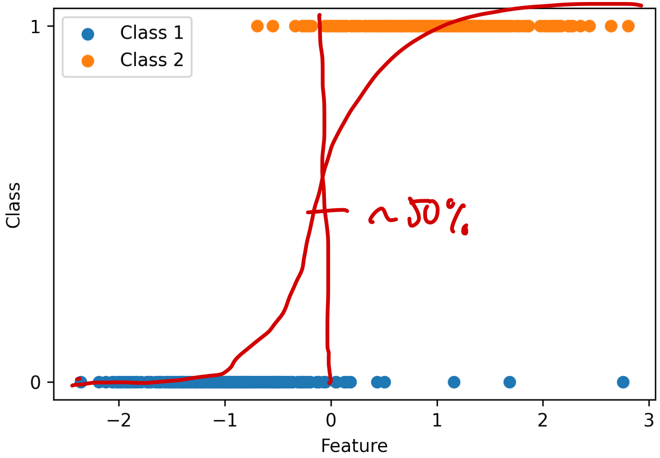
Logistic Regression I: Binary Classification and Class Probabilities

- In contrast to linear regression the type of the dependent (predicted) variable is binary instead of continuous $\rightarrow$ we need some modification to the existing framework.
- Additional we want to calculate class probabilities using linear functions.
- Given a **binary classification** problem with features $x \in \mathbb{R}^p$, the class probabilities are defined formally as:

$$P(Y = 0|X = x) \tag{18}$$

$$P(Y = 1|X = x) \tag{19}$$

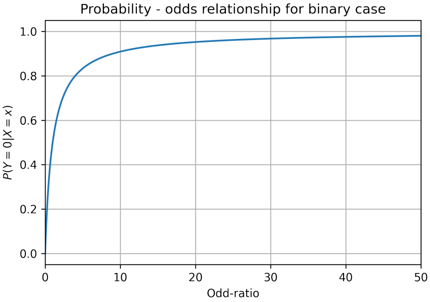
Binary Classification - Features, Target Relationship



Logistic Regression II: Class Probabilities and Odd-Ratio

- Consider ratio between odds of events (odd-ratio):

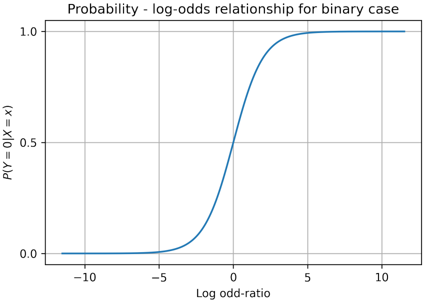
$$\text{odd-ratio} = \frac{P(Y=0|X=x)}{P(Y=1|X=x)} = \frac{P(Y=0|X=x)}{1-P(Y=0|X=x)} \tag{20}$$



- Non-symmetric function
- Odd-ratios for $P \in [0,1]$ ranges from 0 to infinity

Logistic Regression II: Class Probabilities and Log Odd-Ratio

- Now consider log odd ratio (log-odds) log(odd-ratio)
- Symmetric function $\rightarrow$ log-odds against (-) and in (+) favour while value stays the same



$$P(Y=0|X=x)\begin{cases} > & P(Y=1|X=x): \quad \text{the log odd-ratio is positive} \\ = & P(Y=1|X=x): \quad \text{the log odd-ratio is zero} \\ < & P(Y=1|X=x): \quad \text{the log odd-ratio is negative} \end{cases} \tag{21}$$

Logistic Regression III: From Log-Odds to Probabilities

- Suppose event with binary outcome with $p = 0.2$ and $\overline{p} = 0.8$ we can get the following conversion rules:

$$odds = \frac{0.2}{0.8} = 0.25 \quad \rightarrow \quad p = \frac{odds}{1 + odds} = 0.2 \tag{22}$$

$$\log(odds) = \ln \frac{0.2}{0.8} = -1.3863 \quad \rightarrow \quad p = \frac{\exp(\log(odds))}{1 + \exp(\log(odds))} = 0.2 \tag{23}$$

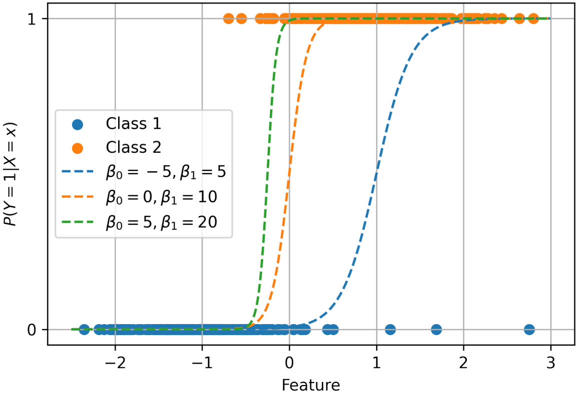
Logistic Regression V

- Decision boundaries is defined by the hyperplane where the log-odds are zero $\{x|\beta_0 + \beta^T x = 0\}$
- Leaves us with:

$$P(Y = 0|X = x) = \frac{\exp(\beta_0 + \beta^T x)}{1 + exp(\beta_0 + \beta^T x)} \tag{24}$$

$$P(Y = 1|X = x) = \frac{1}{1 + \exp(\beta_0 + \beta^T x)} \tag{25}$$

Logistic Regression VI: Examples for Different Linear Functions



Interactive demo in *logistic_regression_interactive.ipynb*

Logistic Regression VII: Maximum Likelihood Fit

- **Likelihood**: Factorized probability of a set of N observations as a function of β

$$\mathcal{L}(\beta) = \prod_{i=0}^{N} P_\beta(Y = y_i | X = x_i) \tag{26}$$

- **Maximum Likelihood**: Find the best fitting parameter β that maximizes the likelihood:

$$\hat{\beta} = \arg\max_\beta \mathcal{L}(\beta) \tag{27}$$

Logistic Regression VIII: Maximum Likelihood Fit (Log Likelihood)

- Consider the likelihood for $N \to \infty$ events:

$$\lim_{N\to\infty} \mathcal{L}(\beta) = \lim_{N\to\infty} \prod_{i=1}^{N} P(Y = y_i|X = x_i) \quad \begin{cases} 1 & P(Y = y_i|X = x_i) = 1 \\ 0 & P(Y = y_i|X = x_i) < 1 \end{cases} \tag{28}$$

- Numerical instabilities $\to$ It is often more convenient to maximize the **log likelihood**
- **Log likelihood:**

$$\ell(\beta) = \log \mathcal{L}(\beta) = \sum_{i=1}^{N} \log P(Y = y_i|X = x_i) \tag{29}$$

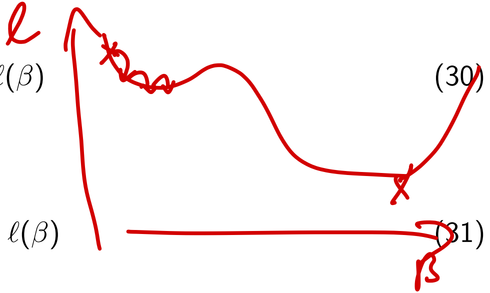
Logistic Regression VIII: Maximum Likelihood Fit (Iterative Solutions)

- No closed form solution for logistic regression $\rightarrow$ instead iterative optimization $\rightarrow$ convergence to optimal solution not guaranteed.

$$\hat{\beta} = \arg\max_{\beta} \ell(\beta) \tag{30}$$

- or equalently

$$\hat{\beta} = \arg\min_{\beta} \; -\ell(\beta) \tag{31}$$



- Many algorithms to find a good maximum likelihood fit: (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

Logistic Regression VIV: Maximum Likelihood - Improved Loss Function

- **Recap**: The loss function of the log-likelihood ℓ is defined as:

$$\ell(\beta) = \log \mathcal{L}(\beta) = \sum_{i=1}^{N} \log P(Y = y_i | X = x_i) \tag{32}$$

- where y_i is the target class for y_i.

$$P(Y = y_i | X = x_i) = \begin{cases} \log P(Y = 1 | X = x_i) & \text{if } y_i = 1 \\ \log\left(1 - P(Y = 1 | X = x_i)\right) & \text{if } y_i = 0 \end{cases} \tag{33}$$

- This gives us :

$$\ell = \sum_{i=1}^{N} \{y_i \log P(y = 1 \, | X = x) + (1 - y_i) \log(1 - P(Y = 1 | X = x))\} \tag{34}$$

Logistic Regression X: Dealing with $K \geq 2$ Classes I

- Let $\mathrm{Y} = \{\ 1,\ 2,\ 3,\ \ldots,\ \mathrm{K}\ \}$
- We want a logistic regression model that provides the posterior probabilities of all K classes.
- **Requirements**:
  1. Sum to one
  2. All outputs remain in [0, 1]

Logistic Regression X: Dealing with $K \geq 2$ Classes II

- We can define K-1 log-odds using K-1 linear models with:

$$\log \frac{P(Y=1|X=x)}{P(Y=K|X=x)} = \beta_{10} + \beta_1^T x \tag{35}$$

$$\log \frac{P(Y=2|X=x)}{P(Y=K|X=x)} = \beta_{20} + \beta_2^T x \tag{36}$$

$$\log \frac{P(Y=K-1|X=x)}{P(Y=K|X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x \tag{37}$$

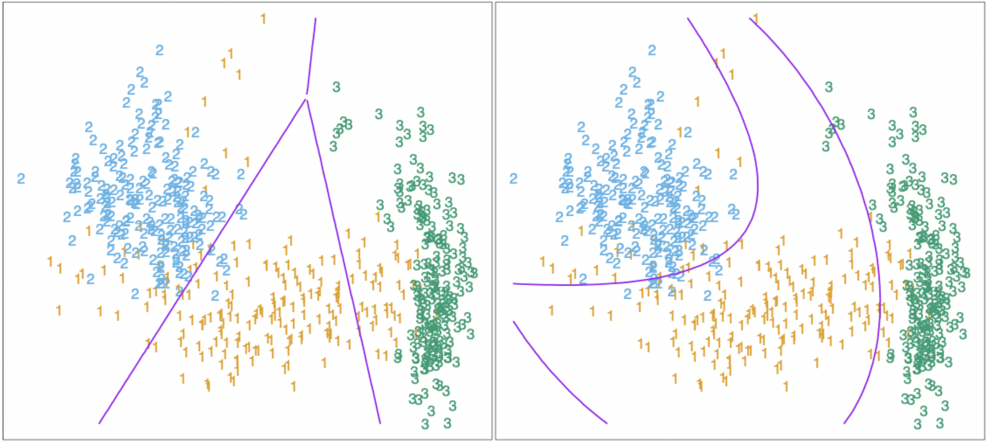
- where the choice of denominator (last class) is arbitrary.

Logistic Regression XI: Dealing with $K \geq 2$ Classes III

- Further we can transform our log-odds to posterior probabilities, respectively as

$$P(Y = 1|X = x) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)}, k = 1, \dots K - 1 \tag{38}$$

$$P(Y = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)} \tag{39}$$



Source: Hastie et al.

Notes on Using Linear Regression

- Remember that you can expand your feature space $\rightarrow$ non-linear dependencies
- For nominal features: use one-hot-encoding:
  - Suppose $X \in \{red, green, blue\}$
  - Instead of X use transformed feature: $I(X = x)$
  - red = (1,0,0), green = (0,1,0), blue = (0, 0, 1)

Logistic Regression: Scikit-Learn

sklearn.linear_model.LogisticRegression ¶

class sklearn.linear_model.LogisticRegression(*penalty='l2'*, ***, *dual=False*, *tol=0.0001*, *C=1.0*, *fit_intercept=True*, *intercept_scaling=1*, *class_weight=None*, *random_state=None*, *solver='lbfgs'*, *max_iter=100*, *multi_class='auto'*, *verbose=0*, *warm_start=False*, *n_jobs=None*, *l1_ratio=None*) [source]

- penalty: Regularization method (we cover this next lecture).
- tol: Tolerance for stopping criteria.
- fit_intercept: Specifies if a constant (a.k.a. bias or intercept) should be added to the decision function.
- class_weight: Weights associated with classes in the form class_label: weight. If not given, all classes are supposed to have weight one.
- solver: Algorithm to use in the optimization problem
- max_iter: Maximum number of iterations taken for the solvers to converge.

Summary: Logistic Regression

- Relationship between posterior probabilities, odds, and log-odds.
- Modeling of log-odd$\rightarrow$posterior relationships with linear models.
- Optimization of logistic regression models using maximum likelihood estimation.
- Logistic regression for $K \geq 2$ classes.

Any Questions ?

References

1. Hastie, T., Tibshirani, R., Friedman, J. H. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
2. Deisenroth, M. P., Faisal, A. A.,, Ong, C. S. (2020). Mathematics for Machine Learning. Cambridge University Press.
3. Hosmer, D. W., Lemeshow, S. (2000). Applied logistic regression.
4. https://www.montana.edu/rotella/documents/502/Prob_odds_log-odds.pdf

Figures:

1. https://www.researchgate.net/figure/Schematic-depiction-of-the-locally-weighted-least-squares-kernel-regressi fig3_351477495