EECE 5644: Logistic Regression

Mark Zolotas

E-mail: m.zolotas@northeastern.edu
Webpage: https://coe.northeastern.edu/people/zolotas-mark/

Tentative Course Outline (Wks. 3-4)

<table>
<tr><th>Topics</th><th>Dates</th><th>Assignments</th><th>Additional Reading</th></tr>
<tr><td>~~Naïve Bayes Classifier &~~
~~*Homework 0 Practice Lab*~~</td><td>~~07/18~~</td><td rowspan="3">**Homework 2** released on Canvas on 07/22
Due 08/01</td><td>~~N/A~~</td></tr>
<tr><td>~~Model Fitting/Training: Bayesian Parameter Estimation~~</td><td>~~07/19-20~~</td><td>~~Chpts. 4.1-4.3, 8.7.2-3 Murphy 2022~~</td></tr>
<tr><td>Logistic Regression</td><td>07/21</td><td>Chpt. 10
Murphy 2022</td></tr>
<tr><td>Model Selection: Hyperparameter Tuning, k-fold Cross-Validation</td><td>07/25</td><td rowspan="3">**Homework 3** released on Canvas on 07/29
Due 08/08</td><td>Chpts. 4.5, 5.2, 5.4.3
Murphy 2022</td></tr>
<tr><td>Regularization, Ridge and Lasso Regression</td><td>07/26</td><td>Chpts. 4.5, 11.1-11.4
Murphy 2022</td></tr>
<tr><td>Neural Networks: Multilayer Perceptrons & Backpropagation</td><td>07/27-28</td><td>Chpts. 13.1-13.5
Murphy 2022</td></tr>
</table>

Maximum Likelihood Estimation (MLE)

- Given i.i.d. samples $\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ from a dataset, take **log likelihood (LL)**:

$$\text{LL}(\boldsymbol{\theta}) = \log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}, y^{(i)} \mid \boldsymbol{\theta})$$

- Or if **unsupervised** then unconditional: $\text{LL}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})$

- **Key Idea:** Good values of $\boldsymbol{\theta}$ should assign high probability to D

- Motivates the choice to **MLE criterion**:

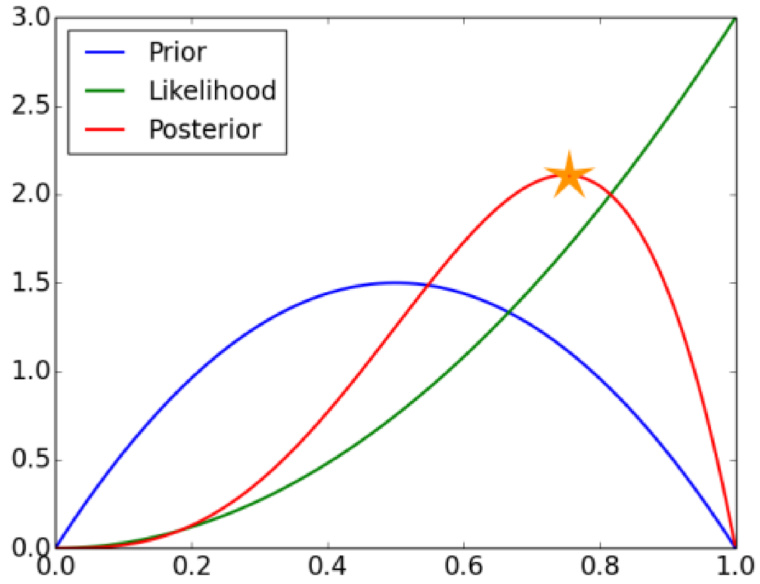
$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})$$

Maximum a Posteriori (MAP) Estimation

- To convert Bayesian parameter estimation into an **optimization** problem, take the most probable parameter estimate (**mode**)

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid D) = \arg\max_{\boldsymbol{\theta}} \left[\log p(D \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right]$$

- Can obtain different loss functions from the posterior distribution
  - Min. MSE => Mean
  - Min. Absolute Error => Median
  - Identical for Gaussian posterior



MAP Estimation Algorithm

Similar framework to MLE, which can be summarized as:

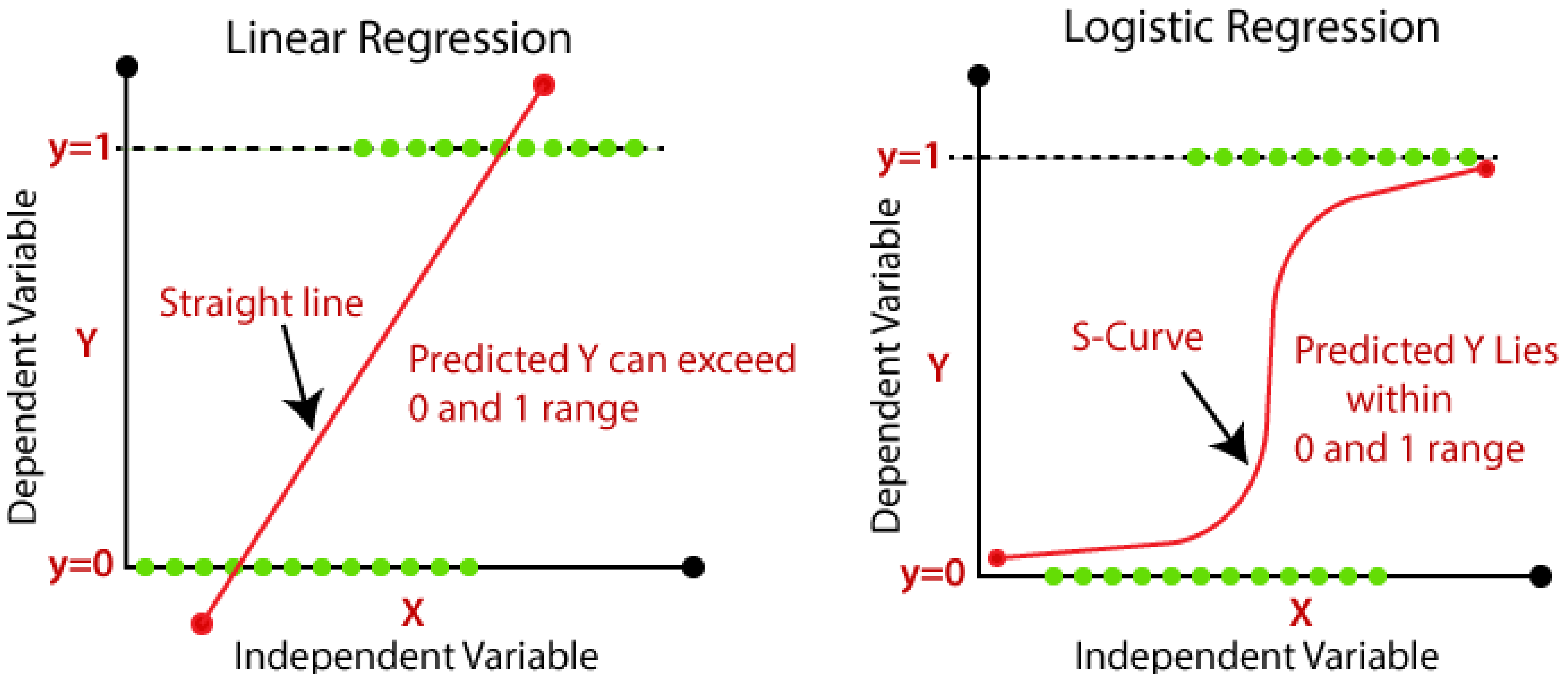
1. Choose parametric model for $p(D \mid \boldsymbol{\theta})$ AND prior $p(\boldsymbol{\theta})$, e.g., **conjugate** prior

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \left[\log p(D \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right]$$

2. Write out log-posterior as log-likelihood plus log-prior, express as an optimization problem

3. Use an optimization algorithm, e.g., GD or SGD, to calculate argmax and derive $\widehat{\boldsymbol{\theta}}_{MAP}$

Logistic Regression



Remember Linear Classifiers?

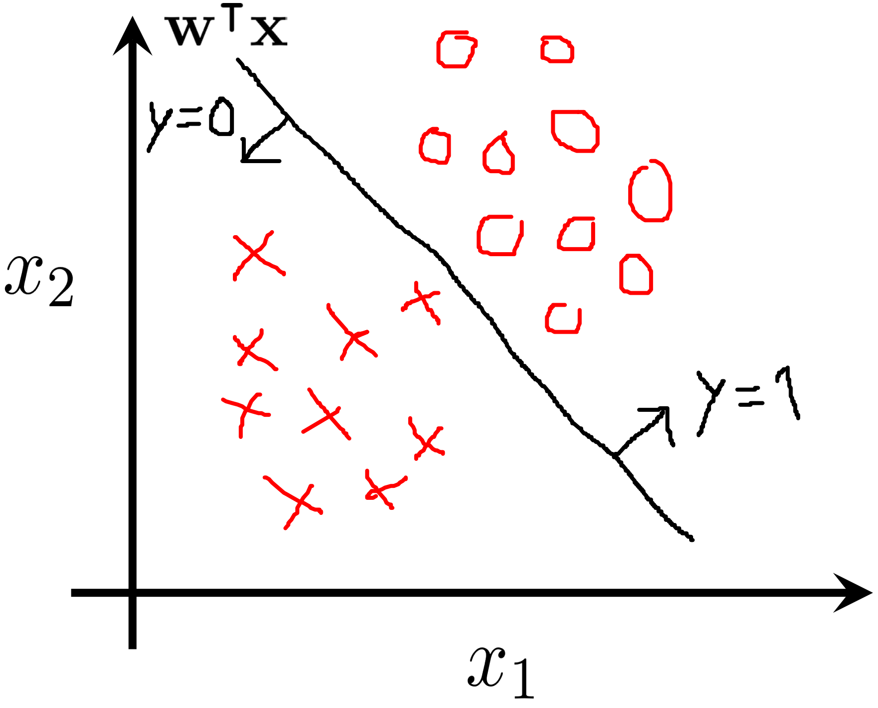
Let $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, N training samples
Inputs $\mathbf{x} \in \mathbb{R}^n$, discrete valued labels $y \in \{0, \ldots, C\}$

- Find decision boundaries by **hyperplane**
- A **linear classifier** is typically of the form:

$$y = g(\mathbf{x}; \boldsymbol{\theta}) = g(\mathbf{w}^\intercal \mathbf{x})$$

- Decision rule:

$$g(\mathbf{x}; \boldsymbol{\theta}) = \begin{cases} 1 & \text{if } \mathbf{w}^\intercal \mathbf{x} > \gamma \\ 0 & \text{otherwise} \end{cases}$$



Logistic Regression – Sigmoid Function

- Takes a probabilistic approach to learning discriminative functions
- Desire $g(\mathbf{w}^T\mathbf{x})$ to output probabilities $p(y = 1 \mid \mathbf{x}; \boldsymbol{\theta})$

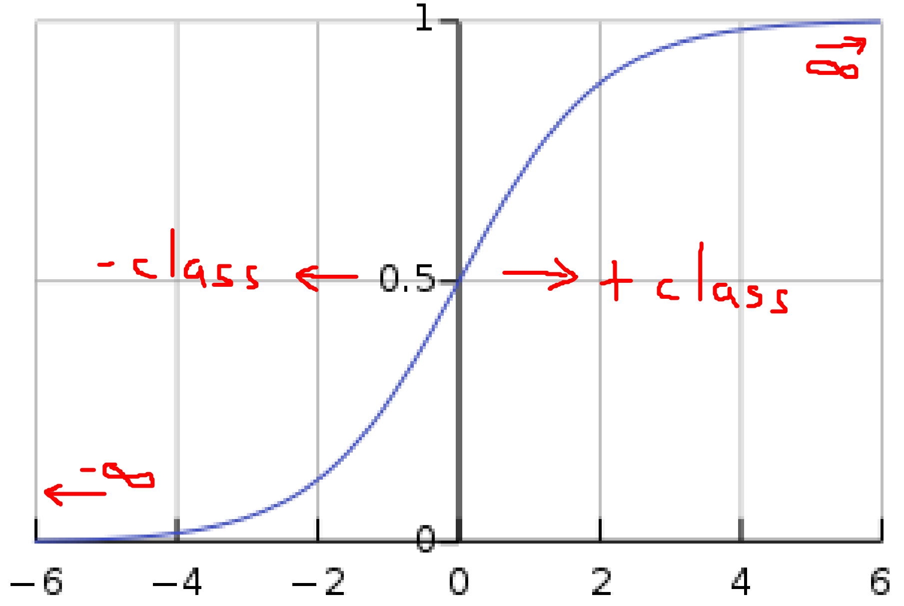
$$0 \leq g(\mathbf{w}^\intercal \mathbf{x}) \leq 1$$

- Model **predictions/hypotheses**:

$$\hat{y} = g(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\intercal \mathbf{x}}}$$

where $g(z) = \dfrac{1}{1 + e^{-z}} = \dfrac{e^z}{e^z + 1}$

Sigmoid/Logistic Function



Logistic Regression – Logits

- The quantity z input to the sigmoid is known as **logit**
- Logistic regression like linear regression has a linear predictor form $\mathbf{w}^T\tilde{\mathbf{x}}$ with an **augmented** input vector $\tilde{\mathbf{x}}$ to account for the line's bias w_0 as:

$$\hat{y} = g(\mathbf{w}^\intercal \tilde{\mathbf{x}}) = g\left(x_0 w_0 + \sum_{j=1}^{n} x_j w_j\right) = \frac{1}{1 + e^{-\mathbf{w}^\intercal \tilde{\mathbf{x}}}}$$

$$\text{where} \quad \mathbf{w} \in \mathbb{R}^{n+1}, \tilde{\mathbf{x}} \in \mathbb{R}^{n+1}$$

Logistic Regression – Decision Boundary

- In a 0-1 loss setting with $g(\mathbf{w}^T\mathbf{x})$ outputting $p(y = 1 \mid \mathbf{x}; \boldsymbol{\theta})$, our optimal decision rule is to predict $y = 1$ iff:

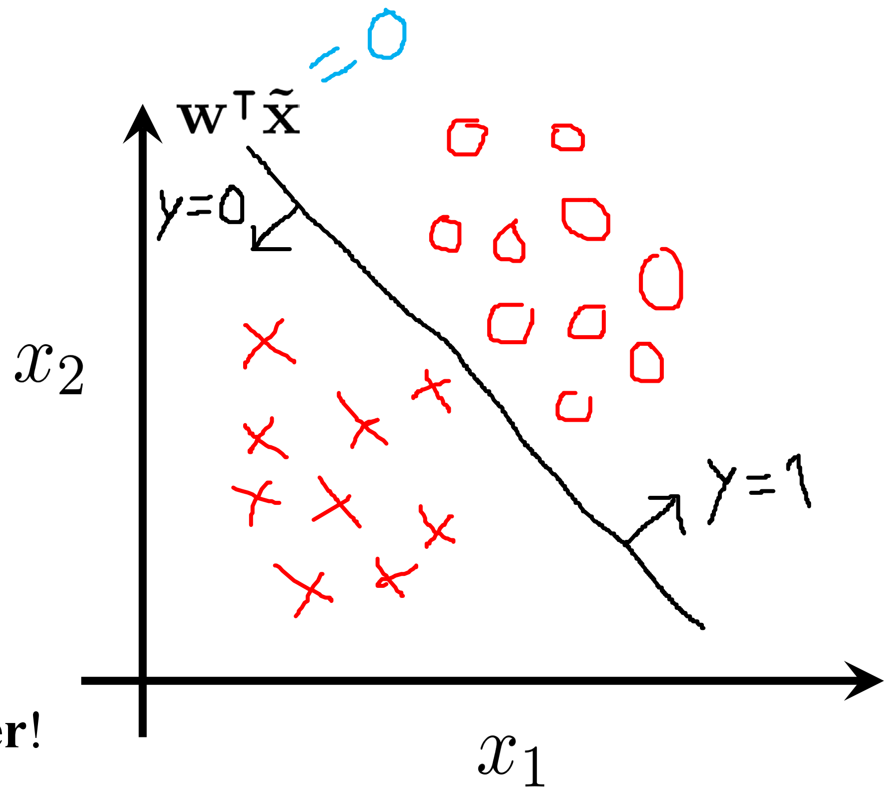
$$p(y = 1 \mid \mathbf{x}; \boldsymbol{\theta}) > p(y = 0 \mid \mathbf{x}; \boldsymbol{\theta})$$
$$g(\mathbf{w}^\intercal\tilde{\mathbf{x}}) > 1 - g(\mathbf{w}^\intercal\tilde{\mathbf{x}})$$

- Rearrange and take logs:

$$\log g(\mathbf{w}^\intercal\tilde{\mathbf{x}}) - \log\left(1 - g(\mathbf{w}^\intercal\tilde{\mathbf{x}})\right) > 0$$

- Decision boundary:

$$\sum_{j=0}^{n} x_j w_j > 0$$ **Linear classifier!**



Nonlinear Classification

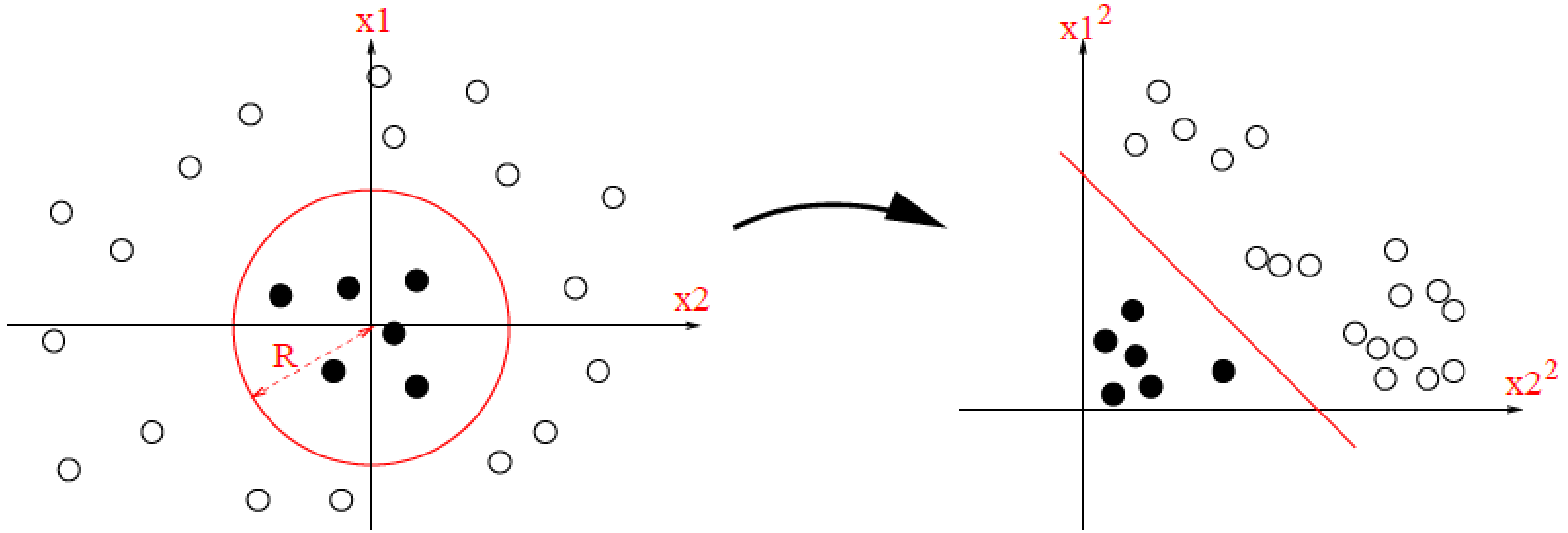


Figure 10.3: Illustration of how we can transform a quadratic decision boundary into a linear one by transforming the features from $\boldsymbol{x} = (x_1, x_2)$ to $\phi(\boldsymbol{x}) = (x_1^2, x_2^2)$. Used with kind permission of Jean-Philippe Vert.

Transform features $\phi(\mathbf{x})$, *e.g.* $\phi(x_1, x_2) = [1, x_1^2, x_2^2]$ with $\mathbf{w} = [-R^2, 1, 1]$, such that the decision boundary $\mathbf{w}^\intercal \phi(\mathbf{x}) = x_1^2 + x_2^2 - R^2$ is a circle with radius R.

Sources – Kevin Murphy, "Probabilistic Machine Learning: An Introduction", 2022

MLE Algorithm

Summary of MLE framework for parameter estimation:

1. Choose parametric model for $p(D \mid \boldsymbol{\theta})$ and define PMF/PDF

2. Write out log-likelihood, $\text{LL}(\boldsymbol{\theta})$ or $\text{NLL}(\boldsymbol{\theta})$, and express as argmax/min for optimization

3. Use an optimization algorithm, e.g., GD or SGD to calculate argmax and derive $\hat{\boldsymbol{\theta}}_{MLE}$

Logistic Regression – Class-Conditional Likelihood

- Observe that our **binary** probabilistic classifier corresponds to:

$$p(y \,|\, \mathbf{x}; \boldsymbol{\theta}) = \mathrm{Ber}\left(y \,|\, g(\mathbf{w}^\intercal \tilde{\mathbf{x}})\right)$$
$$= g(\mathbf{w}^\intercal \tilde{\mathbf{x}})^y (1 - g(\mathbf{w}^\intercal \tilde{\mathbf{x}}))^{(1-y)}$$

1

- Thus **class-conditional likelihood** for all N training samples is:

$$\mathrm{LL}(\boldsymbol{\theta}) = \frac{1}{N} \log p(\mathcal{D} \,|\, \boldsymbol{\theta}) = \frac{1}{N} \log \prod_{i=1}^{N} \left[g(\mathbf{w}^\intercal \tilde{\mathbf{x}}^{(i)})^{y^{(i)}} (1 - g(\mathbf{w}^\intercal \tilde{\mathbf{x}}^{(i)}))^{(1-y^{(i)})} \right]$$

2

$$= \frac{1}{N} \sum_{i=1}^{N} \log \left[g(\mathbf{w}^\intercal \tilde{\mathbf{x}}^{(i)})^{y^{(i)}} (1 - g(\mathbf{w}^\intercal \tilde{\mathbf{x}}^{(i)}))^{(1-y^{(i)})} \right]$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left[y^{(i)} \log g(\mathbf{w}^\intercal \tilde{\mathbf{x}}^{(i)}) + (1 - y^{(i)}) \log(1 - g(\mathbf{w}^\intercal \tilde{\mathbf{x}}^{(i)})) \right]$$

Derivative of Sigmoid

- **Optimization problem:** Minimize negative class-conditional log likelihood

② $$\arg\min_{\theta} \mathrm{NLL}(\boldsymbol{\theta}) = \arg\min_{\theta} -\frac{1}{N}\sum_{i=1}^{N} \log p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta})$$

- Will be useful to have **derivative** of sigmoid $g(z)$ for optimization

$$\frac{d}{dz}\left[\frac{1}{1+e^{-z}}\right] = \frac{1}{(1+e^{-z})^2} \cdot e^{-z}$$
$$= \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}}\right)$$
$$= g(z)(1-g(z))$$

MLE for Logistic Regression

$$\operatorname*{arg\,min}_{\theta} \mathrm{NLL}(\boldsymbol{\theta}) = \operatorname*{arg\,min}_{\theta} -\frac{1}{N} \log \prod_{i=1}^{N} p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta})$$

- Find critical point where $\frac{dNLL(\boldsymbol{\theta})}{d\mathbf{w}} = 0$, so first derive gradient of NLL($\boldsymbol{\theta}$):

$$\frac{d\mathrm{NLL}(\boldsymbol{\theta})}{d\mathbf{w}} = -\frac{1}{N} \sum_{i=1}^{N} \frac{d}{d\mathbf{w}} \left[y^{(i)} \log g(\mathbf{w}^{\intercal}\tilde{\mathbf{x}}^{(i)}) + (1 - y^{(i)}) \log(1 - g(\mathbf{w}^{\intercal}\tilde{\mathbf{x}}^{(i)})) \right]$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left[\left(y^{(i)}(1 - g(\mathbf{w}^{\intercal}\tilde{\mathbf{x}}^{(i)})) - (1 - y^{(i)}) g(\mathbf{w}^{\intercal}\tilde{\mathbf{x}}^{(i)}) \right) \tilde{\mathbf{x}}^{(i)} \right]$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left[\left(y^{(i)} - g(\mathbf{w}^{\intercal}\tilde{\mathbf{x}}^{(i)}) \right) \tilde{\mathbf{x}}^{(i)} \right]$$

Gradient Descent for Logistic Regression

1. Initialize $\boldsymbol{\theta}^{(0)}$

2. Repeat until convergence:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla \mathrm{NLL}(\boldsymbol{\theta}^{(t)})$$

$$= \boldsymbol{\theta}^{(t)} - \alpha \left(\frac{1}{N} \sum_{i=1}^{N} \left[\left(g(\mathbf{w}^\intercal \tilde{\mathbf{x}}^{(i)}) - y^{(i)} \right) \tilde{\mathbf{x}}^{(i)} \right] \right)$$

3. When $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$, then we have derived $\mathbf{w}^* = \hat{\boldsymbol{\theta}}_{MLE}$ using GD

Coding Break

Multinomial Logistic Regression

- Can extend logistic regression case where $C > 2$, with model:

$$p(y \mid \mathbf{x}; \boldsymbol{\theta}) = \mathrm{Cat}\left(y \mid S(\mathbf{W}^\intercal \tilde{\mathbf{x}})\right)$$

$$\mathrm{Cat}\left(y \mid \boldsymbol{\theta}\right) = \prod_{c=1}^{C} \theta_c^{y=c} \quad \text{i.e.} \quad p(y = c \mid \boldsymbol{\theta}) = \theta_c$$

- With θ_c as the probability of observing class c
- $\boldsymbol{\theta} = \mathbf{W} \in \mathbb{R}^{C \times (n+1)}$ is the weights matrix, assuming bias vector included
- **Softmax** function $S(\cdot)$ produces probability vector from logits $\mathbf{a}$:

$$S(\mathbf{a}) = \left[\frac{e^{a_1}}{\sum_{c'=1}^{C} e^{a'_c}}, \cdots, \frac{e^{a_C}}{\sum_{c'=1}^{C} e^{a'_c}}\right]$$

Concluding Remarks

- Many learning algorithms have probabilistic interpretations
- Code:

https://github.com/mazrk7/EECE5644_IntroMLPR_LectureCode/blob/main/notebooks/linear_classification/logistic_regression_gd.ipynb

- Beware of overfitting; next we tackle **regularization** and **model selection**!
- Questions?