Lecture 08: Logistic Regression

Introduction to Machine Learning [25737]

Sajjad Amini

Sharif University of Technology

Contents

References

Except explicitly cited, the reference for the material in slides is:

- Murphy, K. P. (2022). *Probabilistic machine learning: an introduction.* MIT press.

Section 1

Approach Definitions

Approach Definitions

Logistic Regression

Logistic regression is discriminative classification model $p(y|\boldsymbol{x};\boldsymbol{\theta})$ (supervised learning) where:

$\boldsymbol{x} \in \mathbb{R}^D$	Fixed dimension input vector
$y \in \{1, \ldots, C\}$	Class label
$\boldsymbol{\theta}$	Model parameters

Based on the value of C, we have:

$C = 2$	Binary logistic regression (BLR)
$C > 2$	Milticlass logistic regression (MLR)

Section 2

Binary Logistic Regression

Binary Logistic Regression

Model

Model:

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = \text{Ber}(y|\sigma(\boldsymbol{w}^T\boldsymbol{x} + b))$$

where:

$\sigma(\cdot)$	Sigmoid function
$\boldsymbol{w}$	Weight vector
b	Bias value
$\boldsymbol{\theta} = [b; \boldsymbol{w}]$	Model parameters

Label Set

Define *logit* $a = \boldsymbol{w}^T\boldsymbol{x} + b$.

- If $y \in \{0, 1\}$ then $\begin{cases} p(y=1|\boldsymbol{x};\boldsymbol{\theta}) = \sigma(a) \\ p(y=0|\boldsymbol{x};\boldsymbol{\theta}) = 1 - \sigma(a) = \sigma(-a) \end{cases}$
- If $\tilde{y} \in \{-1, 1\}$ then $p(\tilde{y}|\boldsymbol{x};\boldsymbol{\theta}) = \sigma(\tilde{y}a)$

Decision Boundary

Decision Boundary for Binary Classification

Assume we decide based on l_{01} loss. Decision boundary corresponds to the point $\boldsymbol{x}^\star \in \mathbb{R}^D$ where $p(y = 1|\boldsymbol{x} = \boldsymbol{x}^\star; \boldsymbol{\theta}) = 0.5$.

Decision Boundary

We want to find function $g(\boldsymbol{x})$ that outputs 1 if $y = 1$ is more probable and 0 otherwise. Thus:

$$g(\boldsymbol{x}) = \mathbb{I}(p(y=1|\boldsymbol{x};\boldsymbol{\theta}) > p(y=0|\boldsymbol{x};\boldsymbol{\theta})) = \mathbb{I}\left(\log \frac{p(y=1|\boldsymbol{x};\boldsymbol{\theta})}{p(y=0|\boldsymbol{x};\boldsymbol{\theta})} > 0\right) = \mathbb{I}(a > 0)$$

Thus decision boundary is:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = b + \langle \boldsymbol{w}, \boldsymbol{x} \rangle = 0$$

Decision Boundary

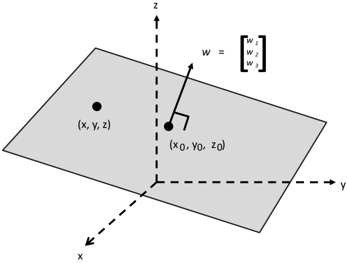
Decision Boundary Characterization

We know point on the hyperplane must satisfy $\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}_0) = 0$ where $\boldsymbol{x}_0$ is a vector on the hyper plane and $\boldsymbol{w}$ is normal vector. Thus:
Decision boundary is a hyperplane with normal vector $\boldsymbol{w}$ and $b = -\langle \boldsymbol{w}, \boldsymbol{x}_0 \rangle$

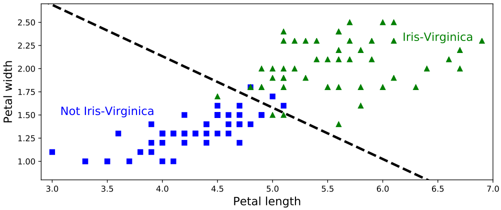
Linearly Separable

If we can perfectly separate the training samples of a binary classification problem using a hyperplane, then the problem is known as linearly separable.

Decision Boundary



(a) Decision boundary in 3D space



(b) Decision boundary for Iris-Virginica flower

Feature Transformation

Nonlinear Decision Boundary

Assume $\boldsymbol{\phi}(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ represents a feature transformer. As and example consider: $\boldsymbol{\phi}(x_1, x_2) = [1, x_1^2, x_2^2]$. Let $\boldsymbol{w} = [-R^2, 1, 1]$. Then decision boundary is:

$$\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle = 0$$

which represents a circle (nonlinear decision boundary).

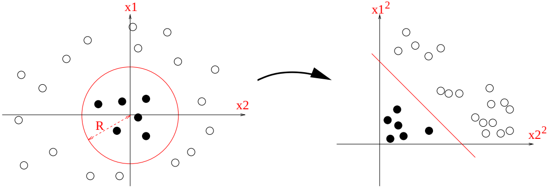


Figure: Nonlinear decision boundary for BLR

MLE

Reformulating logit

$$a = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \langle [b, \boldsymbol{w}], [1, \boldsymbol{x}] \rangle, \begin{cases} [b; \boldsymbol{w}] : \text{Augmented weight vector} \\ [1; \boldsymbol{x}] : \text{Augmented input feature} \end{cases}$$

NLL

Assume $\mu_n = \sigma(a_n)$ and $y \in \{0, 1\}$, then:

$$\begin{aligned} \text{NLL}(\boldsymbol{w}) &= -\frac{1}{N} \log p(\mathcal{D}|\boldsymbol{w}) = -\frac{1}{N} \log \prod_{n=1}^{N} \text{Ber}(y_n|\mu_n) \\ &= -\frac{1}{N} \sum_{n=1}^{N} [y_n \log \mu_n + (1 - y_n) \log(1 - \mu_n)] = \frac{1}{N} \sum_{n=1}^{N} \mathbb{H}(y_n, \mu_n) \end{aligned}$$

Derivatives

Gradient vector

$$\boldsymbol{g}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \operatorname{NLL}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} (\mu_n - y_n)\boldsymbol{x}_n = \frac{1}{N}(\mathbf{1}_N^T(\operatorname{diag}(\boldsymbol{\mu} - \boldsymbol{y})\boldsymbol{X}))^T$$

where $\boldsymbol{X} = \begin{bmatrix} - & \boldsymbol{x}_1^T & - \\ \vdots & & \\ - & \boldsymbol{x}_N^T & - \end{bmatrix}$, $\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_N \end{bmatrix}$, $\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$.

Hessian Matrix

$$\boldsymbol{H}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \nabla_{\boldsymbol{w}}^T \operatorname{NLL}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} (\mu_n(1 - \mu_n)\boldsymbol{x}_n)\boldsymbol{x}_n^T = \frac{1}{N}\boldsymbol{X}^T \boldsymbol{S} \boldsymbol{X}$$

where $\boldsymbol{S} = \operatorname{diag}(\mu_1(1 - \mu_1), \ldots, \mu_N(1 - \mu_N))$.

Characterization of NLL($\boldsymbol{w}$)

$\boldsymbol{H}(\boldsymbol{w})$ is PD

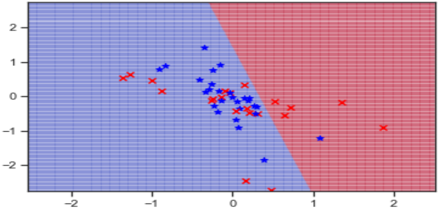
$$\forall \boldsymbol{v} : \boldsymbol{v}^T \boldsymbol{H}(\boldsymbol{w})\boldsymbol{v} = \frac{1}{N}\boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{S}\boldsymbol{X}\boldsymbol{v} = \frac{1}{N}(\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{X}\boldsymbol{v})^T(\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{X}\boldsymbol{v}) = \frac{1}{N}\|\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{X}\boldsymbol{v}\|_2^2 > 0$$

provided $N(\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{X}) = \{\boldsymbol{0}\}$

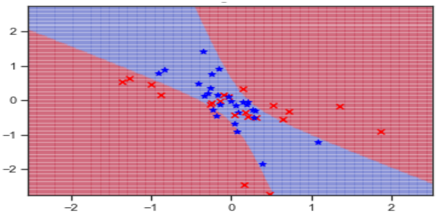
Global minimizer

Thus NLL($\boldsymbol{w}$) is twice differentiable and its hessian matrix is PSD. Thus NLL($\boldsymbol{w}$) is convex and stationary point $\boldsymbol{w}^\star$ ($\boldsymbol{g}(\boldsymbol{w}^\star)$) is the global minimizer.

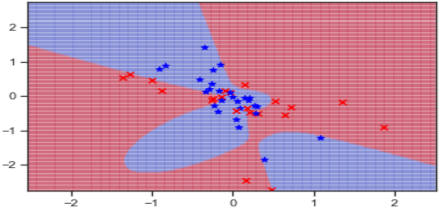
Overfitting Problem



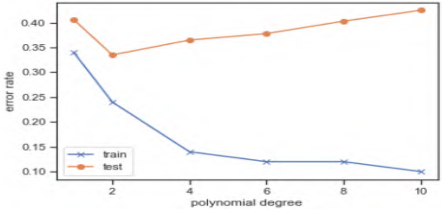
(a) $K = 1$



(b) $K = 2$



(c) $K = 4$



(d) Error vs Complexity

Figure: Overfitting of BLR model when increasing the transformation complexity

MAP Estimation

Weights Amplitude vs Model Complexity

$$K = 1 \Rightarrow \widehat{\boldsymbol{w}}_{mle} = (0.513, 0.119)$$
$$K = 2 \Rightarrow \widehat{\boldsymbol{w}}_{mle} = (2.275, 0.060, 11.842, 15.403, 2.512)$$
$$K = 4 \Rightarrow \widehat{\boldsymbol{w}}_{mle} = (-3.078, \ldots, -9.032, 51.771, 10.250)$$

Overfitting is accompanied by increasing the amplitude of weights.
Solution: One solution is to add a zero-mean Gaussian prior as $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\mathbf{0}, C\boldsymbol{I})$

MAP Estimation

Objective Function

Using MAP estimation we have the following objective function:

$$\text{PNLL}(\boldsymbol{w}) = \text{NLL}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2$$

- The above formulation is called ℓ_2 regularization or *Weight Decay*.

Hyper-parameter Effect

Based on lambda:

- $\lambda \uparrow \Rightarrow$ more penalization $\Rightarrow$ less flexible model
- $\lambda \downarrow \Rightarrow$ less penalization $\Rightarrow$ more flexible model

Derivatives

In this case, the derivatives are calculated as:

$$\begin{aligned}\text{PNLL}(\boldsymbol{w}) &= \text{NLL}(\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{w} \\ \nabla_{\boldsymbol{w}} \text{PNLL}(\boldsymbol{w}) &= \boldsymbol{g}(\boldsymbol{w}) + 2\lambda \boldsymbol{w} \\ \nabla^2_{\boldsymbol{w}} \text{PNLL}(\boldsymbol{w}) &= \boldsymbol{H}(\boldsymbol{w}) + 2\lambda \boldsymbol{I}\end{aligned}$$

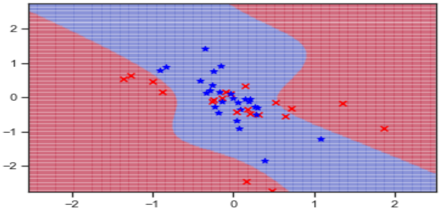
Positive Definiteness of Hessian Matrix

Assume $\lambda > 0$, then:

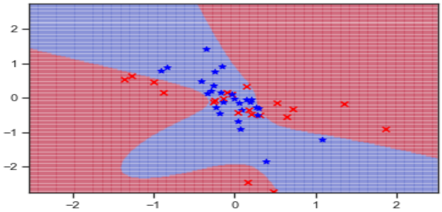
$$\begin{aligned}\forall \boldsymbol{v} : \boldsymbol{v}^T \nabla^2_{\boldsymbol{w}} \text{PNLL}(\boldsymbol{w}) \boldsymbol{v} &= \boldsymbol{v}^T \boldsymbol{H}(\boldsymbol{w}) \boldsymbol{v} + 2\lambda \boldsymbol{v}^T \boldsymbol{I} \boldsymbol{v} = \frac{1}{N} \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{S} \boldsymbol{X} \boldsymbol{v} + 2\lambda \|\boldsymbol{v}\|_2^2 \\ &= \frac{1}{N} \|\boldsymbol{S}^{\frac{1}{2}} \boldsymbol{X} \boldsymbol{v}\|_2^2 + 2\lambda \|\boldsymbol{v}\|_2^2 > 0\end{aligned}$$

$\nabla^2_{\boldsymbol{w}} \text{PNLL}(\boldsymbol{w})$ is always PD.

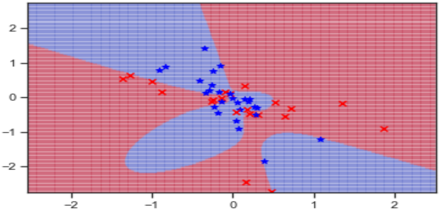
Weight Decay Result



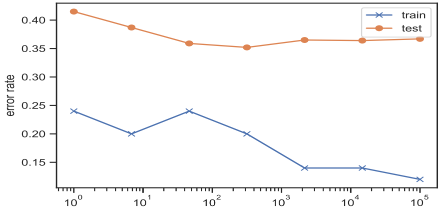
(a) $C = 1$



(b) $C = 316$



(c) $C = 100000$



(d) Error vs inverse regularization

Figure: The effect of weight decay in BLR model performance

Standardization

Reason for Standardization

For MAP estimation, we use $\mathcal{N}(\boldsymbol{w}|\mathbf{0}, \lambda^{-1}\boldsymbol{I})$ prior for weights. This prior implicitly assumes the input features to be similar in magnitude. To assure this, we can use the following methods:

- Individual normalization:
  $$\widehat{x}_{nd} = \frac{x_{nd} - \widehat{\mu}_d}{\widehat{\sigma}_d}, \begin{cases} \widehat{\mu}_d = \frac{1}{N}\sum_{n=1}^{N} x_{nd} \\ \widehat{\sigma}_d^2 = \frac{1}{N}\sum_{n=1}^{N} (x_{nd} - \widehat{\mu}_d)^2 \end{cases}, d = 1, \dots, D$$
- Min-max scaling:
  $$\widehat{x}_{nd} = \frac{x_{nd} - m_d}{M_d - m_d}, \begin{cases} m_d = \min_n x_{nd} \\ M_d = \max_n x_{nd} \end{cases}, d = 1, \dots, D$$
- Data whitening using eigenvectors

Section 3

Multinomial Logistic Regression

Multinomial Logistic Regression

Model

Model:

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = \mathrm{Cat}(y|\mathcal{S}(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}))$$

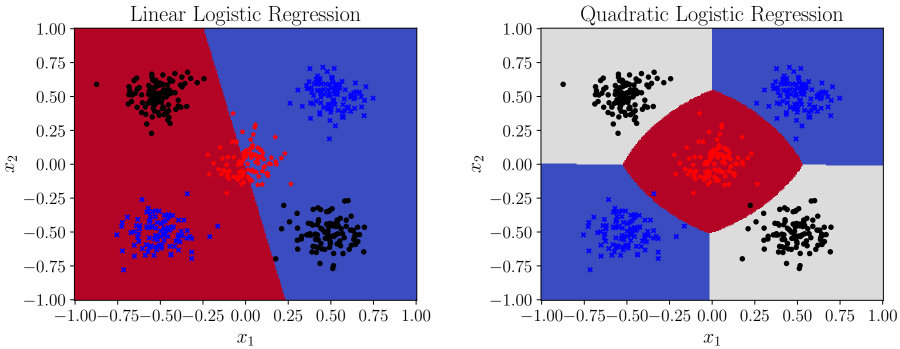
where:

$\mathcal{S}(\cdot)$	Softmax function
$\boldsymbol{W} \in \mathbb{R}^{C\times D}$	Weight matrix
$\boldsymbol{b} \in \mathbb{R}^{C}$	Bias vector
$\boldsymbol{\theta}(\boldsymbol{W},\boldsymbol{b})$	Model parameters
$\boldsymbol{a} = \boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}$	logits vector

Augmented Formulation

$$\boldsymbol{a} = \boldsymbol{W}\boldsymbol{x}+\boldsymbol{b} = [\boldsymbol{b},\boldsymbol{W}]\times[1;\boldsymbol{x}] \begin{cases}[\boldsymbol{b},\boldsymbol{W}] : \text{Augmented weight vector}\\ [1;\boldsymbol{x}] : \text{Augmented input feature}\end{cases}$$

Feature Transformation



(a) Original features (b) Transformed features

Figure: Using feature transformation $\boldsymbol{\phi}(\boldsymbol{x}) = [1; x_1; x_2; x_1^2; x_2^2; x_1 x_2]$ for reaching nonlinear decision boundary

MLE

NLL

$$\mathrm{NLL}(\boldsymbol{W}) = -\frac{1}{N}\log p(\mathcal{D}|\boldsymbol{W}) = -\frac{1}{N}\log\prod_{n=1}^{N}\prod_{c=1}^{C}\mu_{nc}^{y_{nc}} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{nc}\log\mu_{nc}$$
$$= \frac{1}{N}\sum_{n=1}^{N}\mathbb{H}(\boldsymbol{y}_n, \boldsymbol{\mu}_n)$$

where:

$$\mu_{nc} = p(y_{nc} = 1|\boldsymbol{x}_n, \boldsymbol{\theta}) = \mathcal{S}(f(\boldsymbol{x}_n; \boldsymbol{\theta}))_c$$

and $\boldsymbol{y}_n$ is one-hot encoding of the label.

Derivatives

Gradient vector

Assume arbitrary input sample $\boldsymbol{x}_n$ and row $\boldsymbol{w}_j$ in $\boldsymbol{W}$ matrix, then:

$$\begin{cases} \nabla_{\boldsymbol{w}_j} \mathrm{NLL}_n = -\sum_c \frac{\partial}{\partial \mu_{nc}} \left[y_{nc} \log \mu_{nc} \right] = -\sum_c \frac{y_{nc}}{\mu_{nc}} \frac{\partial \mu_{nc}}{\partial a_{nj}} \frac{\partial a_{nj}}{\partial \boldsymbol{w}_j} \\ \frac{\partial \mu_{nc}}{\partial a_{nj}} = \mu_{nc}(\delta_{jc} - \mu_{nj}) \\ \frac{\partial a_{nj}}{\partial \boldsymbol{w}_j} = \boldsymbol{x}_j \end{cases}$$

$$\Rightarrow \nabla_{\boldsymbol{w}_j} \underset{n}{\mathrm{NLL}} = (\mu_{nj} - y_{nj}) \boldsymbol{x}_n$$

$$\Rightarrow \boldsymbol{g}(\boldsymbol{W}) = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n (\boldsymbol{\mu}_n - \boldsymbol{y}_n)^T$$

Derivatives

Hessian Matrix

$$\boldsymbol{H}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\text{diag}(\boldsymbol{\mu}_n) - \boldsymbol{\mu}_n\boldsymbol{\mu}_n^T) \otimes (\boldsymbol{x}_n\boldsymbol{x}_n^T)$$

where

$$\begin{cases} \boldsymbol{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \vdots & a_{m,n} \end{bmatrix} \in \mathbb{R}^{m\times n} \\ \boldsymbol{B} \in \mathbb{R}^{p\times q} \end{cases} \Rightarrow \boldsymbol{A}\otimes\boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{pm\times qn}$$

and Hessian matrix is PD. Thus for 3 features and 2 classes problem, we have:

$$\boldsymbol{H}(\boldsymbol{w}) = \frac{1}{N}\sum_{n}\begin{bmatrix} \mu_{n1} - \mu_{n1}^2 & -\mu_{n1}\mu_{n2} \\ -\mu_{n1}\mu_{n2} & \mu_{n2} - \mu_{n2}^2 \end{bmatrix} \otimes \begin{bmatrix} x_{n1}x_{n1} & x_{n1}x_{n2} & x_{n1}x_{n3} \\ x_{n2}x_{n1} & x_{n2}x_{n2} & x_{n2}x_{n3} \\ x_{n3}x_{n1} & x_{n3}x_{n2} & x_{n3}x_{n3} \end{bmatrix}$$

MAP Estimation

Objective Function

Using MAP estimation we have the following objective function:

$$\text{PNLL}(\boldsymbol{w}) = \sum_{n=1}^{N} \mathbb{H}(\boldsymbol{y}_n, \boldsymbol{\mu}_n) + \lambda \sum_{c=1}^{C} \|\boldsymbol{w}_c\|_2^2$$

Zero Sum Property

At the stationary point for the above regularized formulation, we have:

$$-\sum_{n=1}^{N} \boldsymbol{x}_n(\boldsymbol{\mu}_n - \boldsymbol{y}_n)^T + 2\lambda \boldsymbol{W}^T = \boldsymbol{0}$$

$$\text{For column } j \Rightarrow 2\lambda \sum_c w_{cj} = \sum_n \sum_c (y_{nc} - \mu_{nc}) x_{nj} = \sum_n (1-1) x_{nj} = 0$$

$$\Rightarrow \sum_c w_{cj} = 0$$

Un-identifiability

Un-identifiability

Assume we have a trained MLR model where the posterior probabilities can be computed as:

$$p(y = c|\boldsymbol{x}, \boldsymbol{W}) = \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x})}{\sum_{k=1}^{C} \exp(\boldsymbol{w}_k^T \boldsymbol{x})}$$

If we add a constant vector $\boldsymbol{v}$ to all rows of $\boldsymbol{W}$, then we have:

$$p(y = c|\boldsymbol{x}, \boldsymbol{W} + \mathbf{1}\boldsymbol{v}^T) = \frac{\exp((\boldsymbol{w}_c + \boldsymbol{v})^T \boldsymbol{x})}{\sum_{k=1}^{C} \exp((\boldsymbol{w}_k + \boldsymbol{v})^T \boldsymbol{x})} = \frac{\exp(\boldsymbol{v}) \exp(\boldsymbol{w}_c^T \boldsymbol{x})}{\exp(\boldsymbol{v}) \sum_{k=1}^{C} \exp(\boldsymbol{w}_k^T \boldsymbol{x})}$$
$$= p(y = c|\boldsymbol{x}, \boldsymbol{W})$$

Thus $\boldsymbol{W} + \mathbf{1}\boldsymbol{v}^T$ is also the maximum likelihood estimation and problem is known as un-indentifiability.

Identifiability and MAP

Identifiability and MAP

The MAP estimation can solve un-identifiability because:

- Assume weight matrix $\boldsymbol{W}$, then due to zero sum property we have:

$$\sum_{c=1}^{C} w_{cj} = 0, j = 1, 2, \ldots, D + 1$$

- Assume weight matrix $\boldsymbol{Z} = \boldsymbol{W} + \mathbf{1}\boldsymbol{v}^T$, then due to zero sum property we have:

$$\sum_{c=1}^{C} z_{cj} = 0, j = 1, 2, \ldots, D + 1$$

Thus:

$$\sum_{c=1}^{C} z_{cj} = Cv_j + \sum_{c=1}^{C} w_{cj} = 0 \Rightarrow v_j = 0, j = 1, \ldots, D + 1 \Rightarrow \boldsymbol{v} = \mathbf{0}$$

Section 4

Bayesian Logistic Regression

Bayesian BLR

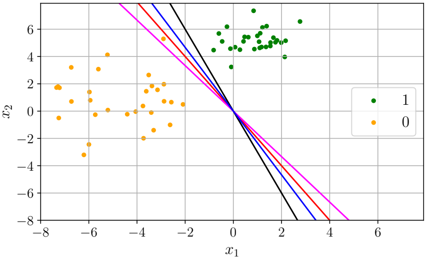
Laplace Approximation to Posterior

Assume we have a BLR model. Using laplace approximation to posterior, we have:

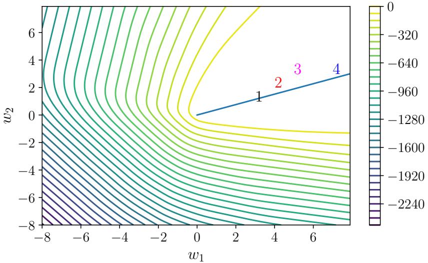
$$p(\boldsymbol{w}|\mathcal{D}) \sim \mathcal{N}(\boldsymbol{w}\|\widehat{\boldsymbol{w}}, \boldsymbol{H}^{-1})$$

where:

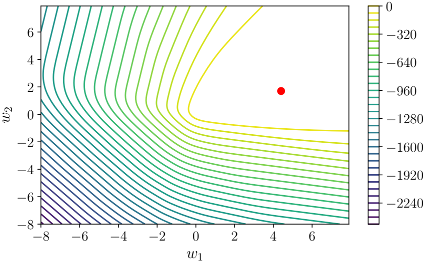
- For MLE we have: $\begin{cases} \widehat{\boldsymbol{w}} = \operatorname{argmin}_{\boldsymbol{w}} \mathrm{NLL}(\boldsymbol{w}) \\ \boldsymbol{H} = \frac{1}{N}\boldsymbol{X}^T\boldsymbol{S}(\widehat{\boldsymbol{w}})\boldsymbol{X} \end{cases}$
- For MAP we have: $\begin{cases} \widehat{\boldsymbol{w}} = \operatorname{argmin}_{\boldsymbol{w}} \left[\mathrm{NLL}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2\right] \\ \boldsymbol{H} = \frac{1}{N}\boldsymbol{X}^T\boldsymbol{S}(\widehat{\boldsymbol{w}})\boldsymbol{X} + 2\lambda\boldsymbol{I} \end{cases}$



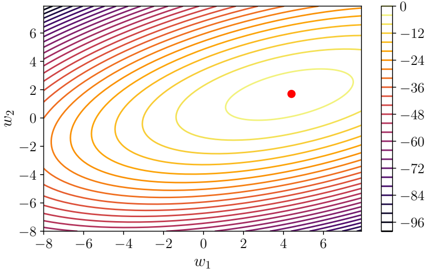
(a) Dataset with four model



(b) Log-likelihood contour plot



(c) Posterior contour plot ($\mathcal{N}(\boldsymbol{w}|\mathbf{0}, 100\boldsymbol{I})$)



(d) Laplace approximation contour plot

Approximating the Posterior Predictive

Posterior Predictive Distribution (PPD)

Posterior predictive distribution is defined as:

$$p(y|\boldsymbol{x}, \mathcal{D}) = \int p(y|\boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{w}|\mathcal{D}) d\boldsymbol{w}$$

Point approximate to PPD

In this approach, we ignore the uncertainty in parameters by assuming:

$$p(\boldsymbol{w}|\mathcal{D}) = \delta(\boldsymbol{w} - \widehat{\boldsymbol{w}}), \begin{cases} \widehat{\boldsymbol{w}} = \widehat{\boldsymbol{w}}_{mle} \\ \widehat{\boldsymbol{w}} = \widehat{\boldsymbol{w}}_{map} \end{cases}$$

And then approximate PPD as:

$$p(y|\boldsymbol{x}, \mathcal{D}) \simeq \int p(y|\boldsymbol{x}, \boldsymbol{w}) \delta(\boldsymbol{w} - \widehat{\boldsymbol{w}}) d\boldsymbol{w} = p(y|\boldsymbol{x}, \widehat{\boldsymbol{w}})$$

Challenge: Ignoring uncertainty

Approximating the Posterior Predictive

Monte Carlo approximate to PPD

In this approach, we draw S sample from the posterior as $\boldsymbol{w}_s \sim p(\boldsymbol{w}|\mathcal{D})$, and then approximate it as:

$$p(\boldsymbol{w}|\mathcal{D}) \simeq \frac{1}{S}\sum_{s=1}^{S}\delta(\boldsymbol{w}-\boldsymbol{w}_s)$$

Then PPD can be approximated as:

$$p(y|\boldsymbol{x},\mathcal{D}) \simeq \frac{1}{S}\sum_{s=1}^{S}\int p(y|\boldsymbol{x},\boldsymbol{w})\delta(\boldsymbol{w}-\boldsymbol{w}_s)d\boldsymbol{w} = \frac{1}{S}\sum_{s=1}^{S}p(y|\boldsymbol{x},\boldsymbol{w}_s)$$

Challenge: Sampling the posterior at the text time

Approximating the Posterior Predictive

Probit Approximation

Assume $\Phi(a)$ to be normal Gaussian CDF. Then this method uses two following relations:

- $\sigma(a) \simeq \Phi(\lambda a), \lambda^2 = \frac{\pi}{8}$
- $\int \Phi(\lambda a)\mathcal{N}(a|m,\nu)da = \Phi\left(\frac{\lambda m}{(1+\lambda^2\nu)^{\frac{1}{2}}}\right) \simeq \sigma(\kappa(\nu)m), \kappa(\nu) \triangleq (1+\pi\nu/8)^{-\frac{1}{2}}$

Thus if we define $a = \boldsymbol{x}^T\boldsymbol{w}$, then we can rewrite PPD as:

$$p(y|\boldsymbol{x},\boldsymbol{w}) = \int p(y|a)p(a|\mathcal{D})da$$

then:

$$p(y=1|\boldsymbol{x},\mathcal{D}) \simeq \sigma(\kappa(\nu)m)$$
$$m = \mathbb{E}[a] = \boldsymbol{x}^T\boldsymbol{\mu}$$
$$\nu = \mathbb{V}[a] = \mathbb{V}[\boldsymbol{x}^T\boldsymbol{w}] = \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x}$$

Intuition

Intuition

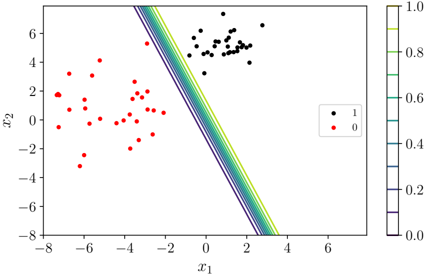
According to probit approximation, the PPD is:

$$p(y = 1|\boldsymbol{x}, \mathcal{D}) \simeq \sigma(\kappa(\nu)m)$$

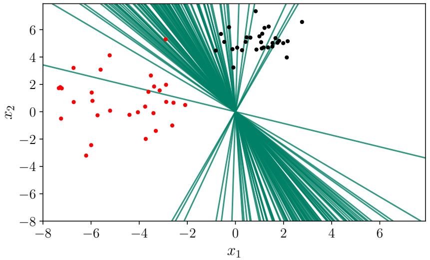
We can conclude two important points:

- Because $0 < \kappa(\nu) < 1$, then $\sigma(\kappa(\nu)m)$ is close to 0.5.
- Bayesian setting does not change the decision boundary because:

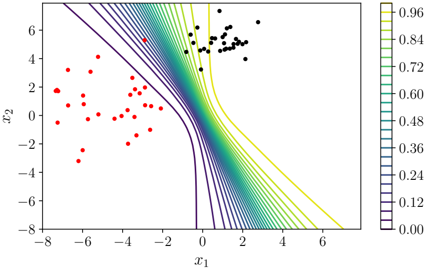
$$p(y = 1|\boldsymbol{x}, \mathcal{D}) = 0.5 \Rightarrow \kappa(\nu)m = 0 \Rightarrow m = 0 \Rightarrow \mathbb{E}[\boldsymbol{w}]^T \boldsymbol{x} = 0$$



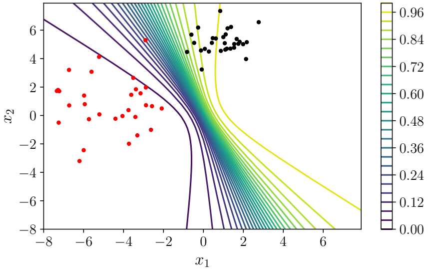
(a) Plug-in approximation $p(y = 1|\boldsymbol{x}, \widehat{\boldsymbol{w}})$



(b) Samples drawn from $p(\boldsymbol{w}|\mathcal{D})$



(c) MC approximation to $p(y = 1|\boldsymbol{x})$



(d) Probit approximation to $p(y = 1|\boldsymbol{x})$