Multinomial Regression (MNR)

Tuesday, October 27, 2020 5:05 PM

Multi-Nomial Regression (MNR): Intro

- What we have seen so far:

  QDA/LDA classifiers: probabilistic classifiers

  Based on the assumption that $\underline{x}|y \sim \mathcal{N}_D(\underline{\mu}_y, \Gamma_y)$

  These classifiers are Probabilistic Generative Models

- MNR belongs to the Probabilistic Discriminative Model family

  It makes no attempt to model $\underline{x}$

  For QDA/LDA we modeled $p(\underline{x}, y) = p(\underline{x}|y)\,p(y)$

  $p(\underline{x}|y) = \mathcal{N}_D(\underline{x}|\underline{\mu}_y, \Gamma_y)$

  $y \sim \text{Cat}(\underline{p})$ ← vector of class prior probabilities

  Parenthesis: Categorical Distribution.

  $$y \sim \text{Cat}(\underline{p}) \overset{\text{def}}{\iff} \mathbb{P}\{y=k\} = \begin{cases} p_k & k=1,\dots,C \\ 0 & \text{otherwise} \end{cases}$$

  $$\iff \mathbb{P}\{y=k\} = \prod_{t=1}^{C} p_t^{[\![t=k]\!]}$$

  Finally, $P(y|\underline{x}) = \dfrac{P(\underline{x}|y)P(y)}{P(\underline{x})}$ ← distr. of the mixture.

  For MNR, it models $p(y|\underline{x})$ directly

  $$P(y=k|\underline{x}) = \frac{e^{h_k(\underline{x})}}{\sum_{k'=1}^{C} e^{h_{k'}(\underline{x})}}$$

  e.g. $C=2$ $\quad P(y=1|\underline{x}) = \dfrac{e^{h_1(\underline{x})}}{e^{h_1(\underline{x})} + e^{h_2(\underline{x})}}$

  $$P(y=2|\underline{x}) = \frac{e^{h_2(\underline{x})}}{e^{h_1(\underline{x})} + e^{h_2(\underline{x})}} =$$

  h_k : scoring/score function of class k.

h_k : scoring/score function of class k.

Interpretation: the higher the value of $h_k(\underline{x})$ is, the more the evidence/likelihood that $\underline{x}$ was drawn from class k

$$P(y=2|\underline{x}) = \frac{\quad}{e^{h_1(\underline{x})} + e^{h_2(\underline{x})}} = 1 - P(y=1|\underline{x})$$

$$\begin{bmatrix} P(y=1|\underline{x}) \\ P(y=2|\underline{x}) \\ \vdots \\ P(y=C|\underline{x}) \end{bmatrix} = \underline{\text{softmax}}(\underline{h}(\underline{x})) = \begin{bmatrix} \text{softmax}_1(\underline{h}(\underline{x})) \\ \vdots \\ \text{softmax}_C(\underline{h}(\underline{x})) \end{bmatrix}$$

where $\text{softmax}_k(\underline{h}(\underline{x})) \triangleq \dfrac{e^{h_k(\underline{x})}}{\sum_{k'} e^{h_{k'}(\underline{x})}}$

MNR would predict $\hat{y} = \arg\max_i P\{y=k|\underline{x}\}$

Furthermore, MNR use the following form of score functions:

$$h_k(\underline{x}) = \underline{w}_k^T \underline{x} \qquad k=1,\dots,C$$

MLPs used as classifiers: $h_k(\underline{x})$ is a much more complex, non-linear function of $\underline{x}$, with many more model parameters

Example: MNR for $C=3$

$$f_1(\underline{x}) = \frac{e^{\underline{w}_1^T\underline{x}}}{e^{\underline{w}_1^T\underline{x}} + e^{\underline{w}_2^T\underline{x}} + e^{\underline{w}_3^T\underline{x}}}, \text{ which models/estimates } P(y=1|\underline{x})$$

$$f_2(\underline{x}) = \frac{e^{\underline{w}_2^T\underline{x}}}{e^{\underline{w}_1^T\underline{x}} + e^{\underline{w}_2^T\underline{x}} + e^{\underline{w}_3^T\underline{x}}} \quad \text{--''--} \quad P(y=2|\underline{x})$$

$$f_3(\underline{x}) = \frac{e^{\underline{w}_3^T\underline{x}}}{e^{\underline{w}_1^T\underline{x}} + e^{\underline{w}_2^T\underline{x}} + e^{\underline{w}_3^T\underline{x}}} \quad \text{--''--} \quad P(y=3|\underline{x})$$

f_k: model's (softmax's) k^{th} output.

Model parameters: $\underline{w}_1, \underline{w}_2, \underline{w}_3$

one can organize these in a single matrix

$$W = \begin{bmatrix} \underline{w}_1^T \\ \underline{w}_2^T \\ \underline{w}_3^T \end{bmatrix} \in \mathbb{R}^{C \times D}$$

$$W = \begin{bmatrix} \underline{w}_1^T \\ \underline{w}_2^T \\ \underline{w}_3^T \end{bmatrix} \in \mathbb{R}^{C \times D}$$

$$\underline{h}(\underline{x}) = \begin{bmatrix} h_1(\underline{x}) \\ h_2(\underline{x}) \\ h_3(\underline{x}) \end{bmatrix} = \begin{bmatrix} \underline{w}_1^T \underline{x} \\ \underline{w}_2^T \underline{x} \\ \underline{w}_3^T \underline{x} \end{bmatrix} = \overbrace{\begin{bmatrix} \underline{w}_1^T \\ \underline{w}_2^T \\ \underline{w}_3^T \end{bmatrix}}^{\equiv W} \underline{x} = W\underline{x}$$

- Why softmax(·)

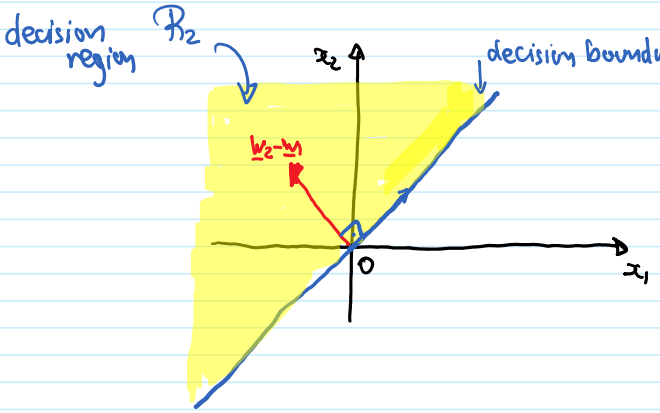
1) $\text{softmax}_k(\underline{h}(\underline{x})) \in (0,1)$
2) $\sum_k \text{softmax}_k(\underline{h}(\underline{x})) = 1$

$\Rightarrow$ softmax(·) is a good function to model a set of probabilities of mutually exclusive event

- Decision Regions/Boundaries (for simplicity, $C=2$)

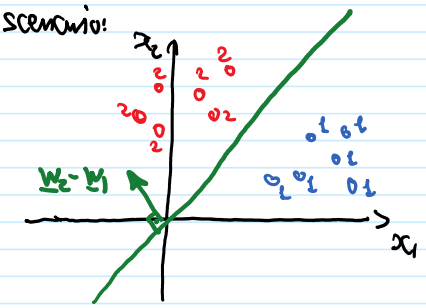
$$\hat{y} = 2 \Leftrightarrow \hat{S}_2(\underline{h}(\underline{x})) > \hat{S}_1(\underline{h}(\underline{x})) \Leftrightarrow$$
$$\Leftrightarrow e^{h_2(\underline{x})} > e^{h_1(\underline{x})} \Leftrightarrow$$
$$\Leftrightarrow h_2(\underline{x}) > h_1(\underline{x}) \Leftrightarrow \dots \Leftrightarrow (\underline{w}_2 - \underline{w}_1)^T \cdot \underline{x} > 0$$

$=$ decision boundary

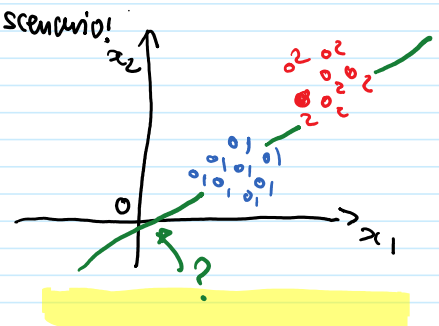


Observations w/ regards to the decis. bound:
- linear (hyper-plane)
- passes through the origin

A good scenario:



A bad scenario:



The work-around: make decision boundaries to by hyperplanes that do not have to pass through the origin.

"Trick": 1) augment each input $\underline{x}$ with an extra "dummy" feature 1

2) augment each weight vector with an extra bias / intercept parameter

E.g. Use as input $\tilde{\underline{x}} \triangleq \begin{bmatrix} \underline{x} \\ 1 \end{bmatrix} \in \mathbb{R}^{D+1}$ instead of just $\underline{x}$ &.

use weight vectors $\tilde{\underline{w}}_k \triangleq \begin{bmatrix} \underline{w}_k \\ b_k \end{bmatrix} \in \mathbb{R}^{D+1}$ instead of just $\underline{w}_k$

Here's why it works (e.g. $C=2$)

$$\hat{y}=2 \Leftrightarrow (\tilde{\underline{w}}_2 - \tilde{\underline{w}}_1)^T \tilde{\underline{x}} > 0 \Leftrightarrow \begin{bmatrix} \underline{w}_2 - \underline{w}_1 \\ b_2 - b_1 \end{bmatrix}^T \begin{bmatrix} \underline{x} \\ 1 \end{bmatrix} > 0 \Leftrightarrow$$

$$\Leftrightarrow \underbrace{(\underline{w}_2 - \underline{w}_1)^T \underline{x} + (b_2 - b_1)}_{=0} > 0$$

eqn of a hyper-plane in general position.

- Translation Invariance of decision rule.

From previous example,

$$\hat{y}=2 \Leftrightarrow (\tilde{\underline{w}}_2 - \tilde{\underline{w}}_1)^T \tilde{\underline{x}} > 0$$

Consider the weight translations: $\tilde{\underline{w}}_1' \triangleq \tilde{\underline{w}}_1 + \tilde{\underline{w}}_0$ ← arbitrary vector

$\tilde{\underline{w}}_2' \triangleq \tilde{\underline{w}}_2 + \tilde{\underline{w}}_0$

$$(\tilde{\underline{w}}_2' - \tilde{\underline{w}}_1')\tilde{\underline{x}} > 0 \Leftrightarrow (\tilde{\underline{w}}_2 - \tilde{\underline{w}}_1)\tilde{\underline{x}} > 0 \Leftrightarrow \hat{y}=2.$$

In general ($C \geq 2$), $\begin{bmatrix} \tilde{\underline{w}}_1^T + \tilde{\underline{w}}_0^T \\ \vdots \\ \tilde{\underline{w}}_C^T + \tilde{\underline{w}}_0^T \end{bmatrix}$ and $\begin{bmatrix} \tilde{\underline{w}}_1^T \\ \vdots \\ \tilde{\underline{w}}_C^T \end{bmatrix}$ produce the

same decision regions and, hence, label predictions.

Training MNR models

Goal: to train MNR classifier whose empirical 0/1 loss (sample misclassif. rate) is minimum.

$$\mathcal{E}_{0/1} \triangleq \frac{1}{N}\sum_{n=1}^{N} l_{0/1}(\hat{y}_n, y_n) \quad ①$$

$$l_{0/1}(\hat{y}(\underline{x}), y) \triangleq [\![\, h_y(\underline{x}) - \max_{k\neq y} h_k(\underline{x}) < 0 \,]\!] = \begin{cases} 0 & \text{if no prediction error} \\ 1 & \text{if } \underline{x} \text{ was misclassified} \end{cases} \quad ②$$

$$\hat{y} = \arg\max_k \hat{P}(l=k|\underline{x}) = \arg\max_k \text{softmax}_k(\underline{h}(\underline{x})) =$$

$$= \arg\max_k e^{h_k(\underline{x})} = \arg\max_k h_k(\underline{x})$$

if $h_y(\underline{x})$ is the largest of the $h_k(\underline{x})$'s : prediction is correct.
—11— is not —11— . : wrong prediction.

$$①② \Rightarrow \mathcal{E}_{0/1} = \frac{1}{N}\sum_{n=1}^{N} \underbrace{[\![\, h_{y_n}(\underline{x}_n) - \max_{k\neq y_n} h_k(\underline{x}_n) < 0 \,]\!]}_{\text{discontinuous} \Rightarrow \text{non-differentiable.}}$$

▶ Cross-Entropy (CE) loss & Average CE (ACE)

Instead of the 0/1 loss, we are going to use this surrogate loss:

$$l_{CE}(\underline{t}, \underline{s}(\underline{x})) = -\sum_{k=1}^{C} t_k \ln(s_k(\underline{x})) = -\ln s_y(\underline{x})$$

$\underline{t}$: 1-hot encoding of $\underline{x}$'s label y; $\underline{s}(\underline{x})$: softmax function evaluated using $\underline{x}$

$$\underline{s}(\underline{x}) \triangleq \frac{1}{\sum_{k'=1}^{C} e^{h_{k'}(\underline{x})}} \begin{bmatrix} e^{h_1(\underline{x})} \\ e^{h_2(\underline{x})} \\ \vdots \\ e^{h_C(\underline{x})} \end{bmatrix}$$

Assume that $C=2$, $y=1$

↳ $s_1(\underline{x}) > s_2(\underline{x}) \Leftrightarrow h_1(\underline{x}) > h_2(\underline{x}) \Rightarrow \hat{y}=1$ (no error)

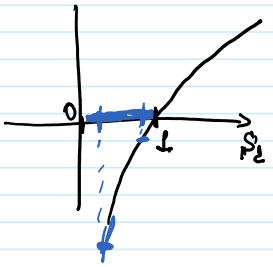
Then, $l_{CE}([\begin{smallmatrix}1\\0\end{smallmatrix}], \underline{s}(\underline{x})) = -1\cdot\ln(s_1(\underline{x})) - 0\cdot\ln(s_2(\underline{x})) = -\ln(\overbrace{s_1(\underline{x})}^{\approx 1}) \approx 0$

Then, $l_{CE}([1,0], \underline{\hat{s}}(\underline{x})) = -1 \cdot \ln(\hat{s}_1(\underline{x})) - 0 \cdot \ln(\hat{s}_2(\underline{x})) = -\ln(\hat{s}_1(\underline{x})) \approx 0$ small ($\hat{s}_1(\underline{x}) \approx 1$)

$\hat{s}_2(\underline{x}) > \hat{s}_1(\underline{x}) \Leftrightarrow h_2(\underline{x}) > h_1(\underline{x}) \Rightarrow \hat{y} = 2$ (mistake)

Then, $l_{CE}([0,1], \underline{\hat{s}}(\underline{x})) = -1 \cdot \ln(\hat{s}_1(\underline{x})) - 0 \cdot \ln(\hat{s}_2(\underline{x})) = -\ln(\hat{s}_1(\underline{x}))$ large. ($\hat{s}_1(\underline{x}) \approx 0$)

$$ACE = \frac{1}{N} \sum_{n=1}^{N} l_{CE}(\underline{t}_n, \underline{\hat{s}}(\underline{x}_n))$$ ← this is differentiable.



Comment:

It turns out (details omitted for the moment) that minimizing ACE depresses the empirical 0/1 loss (it does not necessarily minimize it)

$$\frac{dACE(\tilde{W})}{d\tilde{W}} = \frac{1}{N} \tilde{X}^T (\hat{S} - T) \in \mathbb{R}^{(D+1) \times C}$$

$\tilde{X}^T = [\underline{\tilde{x}}_1 \cdots \underline{\tilde{x}}_N] \in \mathbb{R}^{(D+1) \times N}$

$\hat{S} = \begin{bmatrix} \underline{\hat{s}}^T(\underline{x}_1) \\ \vdots \\ \underline{\hat{s}}^T(\underline{x}_N) \end{bmatrix} \in \mathbb{R}^{N \times C}$

$T = \begin{bmatrix} \underline{t}_1^T \\ \vdots \\ \underline{t}_N^T \end{bmatrix} \in \mathbb{R}^{N \times C}$

Gradient matrix. $\in \mathbb{R}^{(D+1) \times C}$

$$\triangleq \begin{bmatrix} \left(\frac{dACE}{d\underline{\tilde{w}}_1}\right)^T \\ \\ \left(\frac{dACE}{d\underline{\tilde{w}}_C}\right)^T \end{bmatrix}$$

Comment: setting the gradient matrix to $\underline{0}$ and trying to solve for $\{\underline{\tilde{w}}_k\}_{k=1}^{C}$ does not produce a closed-form solution

Only choice left: use a numerical minimization approach: Gradient Descent (with backtracking)

$$\tilde{W}^{new} \leftarrow \tilde{W}^{old} - \eta \left.\frac{d\,ACE}{d\tilde{W}}\right|_{\tilde{W}=\tilde{W}^{old}} =$$

$$= \tilde{W}^{old} - \eta \frac{1}{N} \tilde{X}^T (S - T)$$

↑ computed using the weights in $\tilde{W}^{old}$.

Regarding CE

Divergence measure for distributions

Kullback-Leibler (KL) divergence

$$D_{KL}(Q\|P) \triangleq \int_{\underline{x}\in\mathbb{R}^D} q(\underline{x}) \ln\left(\frac{q(\underline{x})}{p(\underline{x})}\right) d\underline{x} = \underbrace{\int_{\underline{x}\in\mathbb{R}^D} q(\underline{x}) \ln q(\underline{x})\, d\underline{x}}_{\equiv -Ent(Q)} - \underbrace{\int_{\underline{x}\in\mathbb{R}^D} q(\underline{x}) \ln p(\underline{x})\, d\underline{x}}_{= CE(Q,P)}$$

↑ distributions with PDFs $q(\underline{x})$ & $p(\underline{x})$

If $Q \equiv P$ ($\Leftrightarrow q(\underline{x}) = p(\underline{x})$), then $D_{KL}(Q\|P) = 0$

One can show that $D_{KL}(Q\|P) \geq 0$

$D_{KL}(Q\|P) \neq D_{KL}(P\|Q)$ in general

Comments:

$$Ent(Q) \triangleq -\int_{\underline{x}\in\mathbb{R}^D} q(\underline{x}) \ln q(\underline{x})\, d\underline{x}$$

Convention: $0 \cdot \ln 0 = 0$

measure of "spread" of the distribution

$$CE(Q,P) = -\int_{\underline{x}\in\mathbb{R}^D} q(\underline{x}) \ln p(\underline{x})\, d\underline{x} \qquad CE(Q,Q) = Ent(Q)$$

$$D_{KL}(Q\|P) = CE(Q,P) - Ent(Q) \leftarrow$$

The idea is: know distribution parametrized by $\underline{\theta}$

The idea is:

$$\underline{\theta}^* = \arg\min_{\underline{\theta}} D_{KL}(Q \,||\, P_{\underline{\theta}}) = \arg\min_{\underline{\theta}} CE(Q, P_{\underline{\theta}})$$

Q & $P_{\underline{\theta}}$ distribution about some $\underline{x}$ or $(\underline{x}, y)$

Q: unknown distribution from which we can sample data

$P_{\underline{\theta}}$: know distribution parametrized by $\underline{\theta}$

$$= \arg\min_{\underline{\theta}} -\int_{\underline{x}\in\mathbb{R}^D} q(\underline{x}) \ln p_{\underline{\theta}}(\underline{x})\, d\underline{x} = \arg\min_{\underline{\theta}} \int_{\underline{x}\in\mathbb{R}^D} q(\underline{x}) \left[-\ln p_{\underline{\theta}}(\underline{x})\right] d\underline{x}$$

$$\rightarrow \mathbb{E}_Q\{-\ln p(\underline{x}|\underline{\theta})\} \approx$$

$$\approx \frac{1}{N}\sum_n \left[-\ln p(\underline{x}_n|\underline{\theta})\right] \propto$$

where $\underline{x}_n \sim Q$

$$\propto -\sum_n \ln p(\underline{x}_n|\underline{\theta})$$

negative log-likelihood given the assumed model and the data $\mathcal{D} = \{\underline{x}_n \sim Q\}_{n=1}^N$

Big takeaway:

Minimizing the neg. log. likelihood ($\equiv$ CE) for a chosen model ($P(\underline{x}|\underline{\theta})$) and data drawn from an unknown distribution (which we would like to model) is equivalent to minimizing the KL divergence of our model to the unknown distribution

The likelihood for MNR classification is

$$\mathcal{L}(\underline{\theta}|\mathcal{D}) = \binom{N}{N_1,\dots,N_C} \prod_{n=1}^N P(y=y_n|\underline{x}_n)$$

$$\prod_{k=1}^C \left[P(y=k|\underline{x}_n)\right]^{\equiv t_{n,k} \equiv [\![y_n=k]\!]}$$

$$= S_k(\underline{x}_n)$$

typically, ignored

$$\binom{N}{N_1,\dots,N_C} = \frac{N!}{N_1!\,N_2!\cdots N_C!}$$

$$\Rightarrow \quad -\ln\mathcal{L}(\underline{\theta}|\mathcal{D}) = N \cdot ACE(\underline{\theta})$$

Another big takeaway is: ACE is proportional to the neg.-log-likelihood of the MNR classifier

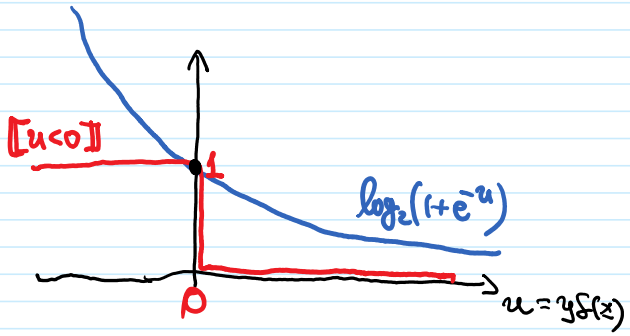
Relationship between 0/1 loss & CE loss

For simplicity, choose $C=2$

Assume that we have observed $(\underline{x}, y)$ $\quad y=\pm 1$ (convenience).

$\hat{y} = \arg\max_{k=\pm 1} h_k(\underline{x})$ $\qquad$ $\hat{y} = \text{sign}\underbrace{\left(h_1(\underline{x}) - h_{-1}(\underline{x})\right)}_{\delta(\underline{x}) \triangleq}$ ✓

$$\ell_{0/1}(y,\hat{y}) = [\![y \neq \hat{y}]\!] = [\![y\hat{y} < 0]\!] = [\![y\delta(\underline{x}) < 0]\!] \leq \log_2\left(1 + e^{-y\delta(\underline{x})}\right) =$$



$$= \log_2\left(1 + e^{-y\delta(\underline{x})}\right) = -\log_2\left(\frac{1}{1+e^{-y\delta(\underline{x})}}\right) \quad (1)$$

$$\frac{1}{1+e^{-y\delta(\underline{x})}} = \frac{e^{t_+\delta(\underline{x})}}{e^{t_+\delta(\underline{x})} + e^{t_-\delta(\underline{x})}} = \frac{e^{t_+\delta(\underline{x})}}{\left[1+e^{\delta(\underline{x})}\right]^{t_+ + t_-}} = \left(\frac{e^{\delta(\underline{x})}}{1+e^{\delta(\underline{x})}}\right)^{t_+}\left(\frac{1}{1+e^{\delta(\underline{x})}}\right)^{t_-} =$$

($t_+ + t_- = 1$)

$t_+ = 1$ if $y=1$

$t_- = 1$ if $y=-1$

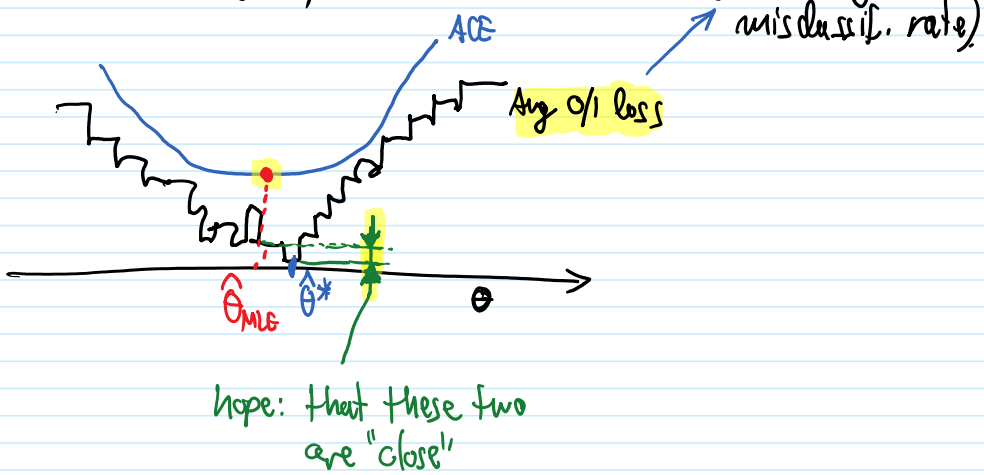
$$= \left(\frac{e^{h_+(\underline{x})}}{e^{h_+(\underline{x})} + e^{h_-(\underline{x})}}\right)^{t_+}\left(\frac{e^{h_-(\underline{x})}}{e^{h_+(\underline{x})} + e^{h_-(\underline{x})}}\right)^{t_-} \quad (2)$$

$= \hat{p}_+(\underline{x})$ $\qquad$ $= \hat{p}_-(\underline{x})$

$(1) \overset{(2)}{\Rightarrow}$ $\ell_{0/1}(y,\hat{y}) \leq -t_+ \log_2 \hat{p}_+(\underline{x}) - t_- \log_2 \hat{p}_-(\underline{x}) \equiv \ell_{CE}(\underline{t}, \hat{\underline{p}}(\underline{x}))$

($t_+ = t_1$, $\hat{p}_+ = \hat{p}_1$, $t_- = t_2$, $\hat{p}_- = \hat{p}_2$)

Big takeaway: CE loss upper-bounds the 0/1 loss (ACE)



● Regularization: squared L_2 penalty term.

Useful if # training samples N << dimensionality D of the feature space.

to avoid overfitting.

$$RACE(\tilde{W}) = ACE(\tilde{W}) + \nu \|W\|_F^2 \quad \text{vs.} \quad \|\tilde{W}\|_F^2 \qquad (P)$$

penalty parameter

regularized ACE

≥ 0

$$\|W\|_F^2 = \text{trace}\{W^T W\} = \sum_{k=1}^{C} \|\underline{w}_k\|_2^2$$

intercept weights are excluded

$$\tilde{W} = \begin{bmatrix} \underline{w}_1 & \cdots & \underline{w}_C \\ b_1 & & b_C \end{bmatrix} = \begin{bmatrix} W \\ \underline{b}^T \end{bmatrix}$$

$\tilde{\underline{w}}_1$, $\tilde{\underline{w}}_C$, W

intercept weights.

$$(P) \Rightarrow \frac{dRACE(\tilde{W})}{d\tilde{W}} = \frac{dACE(\tilde{W})}{d\tilde{W}} + 2\nu \begin{bmatrix} W \\ \underline{0}_C^T \end{bmatrix} = \frac{1}{N}\left(\tilde{X}^T(S-T) + \nu \begin{bmatrix} W \\ \underline{0}_C^T \end{bmatrix}\right)$$

$\nu \to \frac{1}{2N}\nu$

$\tilde{X}^T(S-T)$

$$\nabla_{\tilde{W}} \frac{1}{N} \tilde{X}^T (S - T)$$

$$\frac{d\,\|W\|_F^2}{d\,\tilde{W}} = \begin{bmatrix} \frac{d\,\|W\|_F^2}{dW} \\ \hline \frac{d\,\|W\|_F^2}{d\,\underline{b}^T} \end{bmatrix} = \begin{bmatrix} 2W \\ \hline \underline{0}_C^T \end{bmatrix}$$

$1 \times C$

$$\underline{b} \triangleq \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_C \end{bmatrix}$$

Pseudocode:

For various values of $\gamma \geq 0$ (usually, $\gamma = 10^e$ where $e = -4, -3, \ldots, 3$)

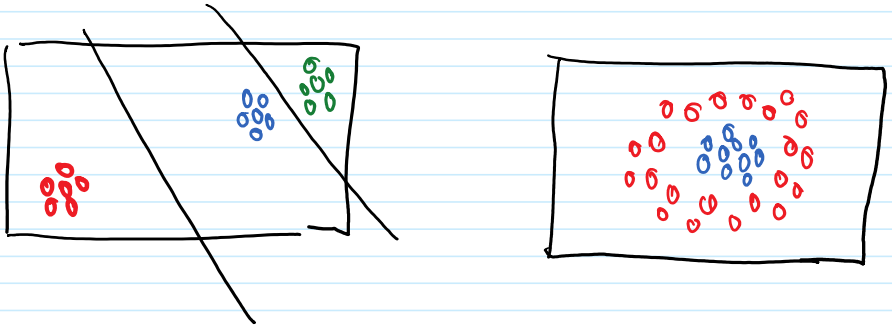
- Train the MNR classifier for this value of γ. (using training set)
  - ↳ Gradient Descent.

$$\tilde{W}^{new} \leftarrow \tilde{W}^{old} - \eta \left. \frac{d\,RACE(\tilde{W})}{d\tilde{W}} \right|_{\tilde{W} = \tilde{W}^{old}}$$

- Compute the performance of the resulting MNR classifier on a hold-out (validation) set.

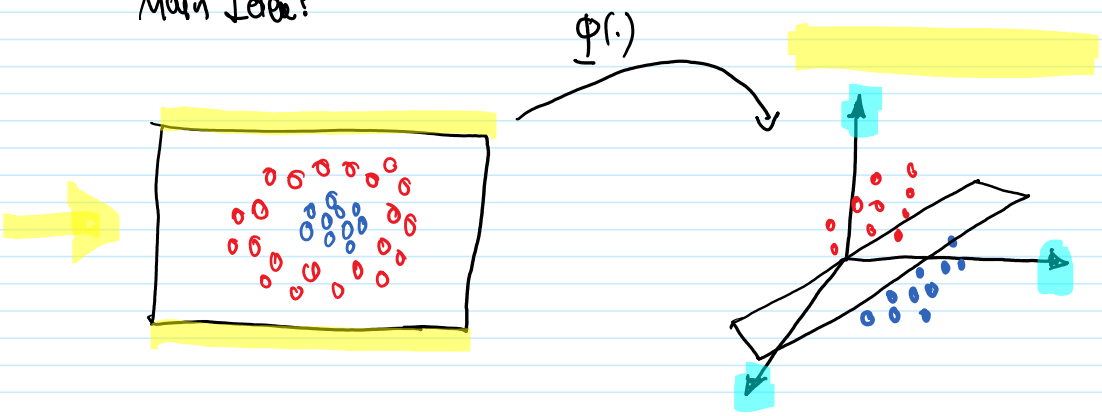
Pick the model ($\Leftrightarrow$ best value of γ) that has the best validation performance ("champion model")

MNR w/ non-linear transformation of features



For the 2nd example, use a suitable non-linear transformation of the features.

Main Idea:



How to find a suitable $\underline{\Phi}(\cdot)$?

- Learned from the data
- "Handcrafted" $\underline{\Phi}(\cdot)$
- "Canned" $\underline{\Phi}(\cdot)$
  - Radial Basis Function mapping:

$$\underline{\Phi}(\underline{x}) = \begin{bmatrix} e^{-\frac{\|\underline{x}-\underline{x}_1\|_2^2}{\delta^2}} \\ \vdots \\ e^{-\frac{\|\underline{x}-\underline{x}_N\|_2^2}{\delta^2}} \end{bmatrix} \in \mathbb{R}^N$$

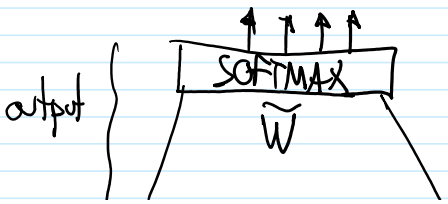
training samples

where δ is a (hyper) parameter

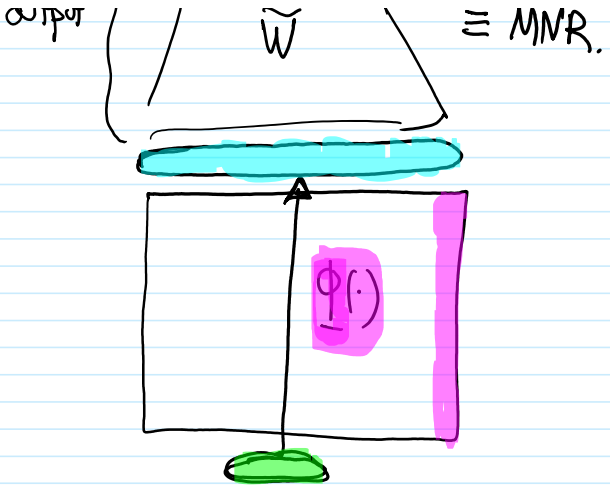
depends on the particular training set.

$D^{new} \equiv N$ ← may lead to potential overfitting.

Train an MNR classifier using the transformed features $\{\underline{\Phi}(\underline{x}_1), \ldots, \underline{\Phi}(\underline{x}_N)\}$.



$\equiv$ MNR.


output
$\tilde{W}$
$\equiv$ MNR.
$\phi(\cdot)$