Supervised Learning :: CHEAT SHEET

Multiclass Classification

Multiclass classification with $g > 2$ classes

$$\mathcal{D} \subset (\mathcal{X} \times \mathcal{Y})^n, \mathcal{Y} = \{1, ..., g\}$$

Goal is to find a model $f : \mathcal{X} \to \mathbb{R}^g$, where g is the number of classes, that minimizes the expected loss over random variables $(\mathbf{x}, y) \sim \mathbb{P}_{xy}$

$$\mathcal{R}(f) = \mathbb{E}_{xy}[L(y, f(\mathbf{x}))] = \mathbb{E}_x \left[\sum_{k \in \mathcal{Y}} L(k, f(\mathbf{x})) \mathbb{P}(y = k | \mathbf{x} = \mathbf{x}) \right]$$

The optimal model for a loss function $L(y, f(\mathbf{x}))$ is

$$\hat{f}(\mathbf{x}) = \arg\min_{f \in \mathcal{H}} \sum_{k \in \mathcal{Y}} L(k, f(\mathbf{x})) \mathbb{P}(y = k | \mathbf{x} = \mathbf{x})$$

ERM: $\hat{f} = \arg\min_{f \in \mathcal{H}} \mathcal{R}_{\text{emp}}(f) = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right)$

One-Hot Encoding

$$\text{with} \quad \mathbb{1}_{\{y=k\}} = \begin{cases} 1 & \text{if } y = k \\ 0 & \text{otherwise} \end{cases}$$

Notations

- Vectors of scores
$$f(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_g(\mathbf{x}))$$
- Vectors of probabilities
$$\pi(\mathbf{x}) = (\pi_1(\mathbf{x}), ..., \pi_g(\mathbf{x}))$$
- Hard labels
$$h(\mathbf{x}) = k, k \in \{1, 2, ..., g\}$$

Loss Functions

- 0-1 Loss
$$L(y, h(\mathbf{x})) = \mathbb{1}_{\{y \neq h(\mathbf{x})\}}$$
- Brier Score
$$L(y, \pi(x)) = \sum_{k=1}^{g} \left(\mathbb{1}_{\{y=k\}} - \pi_k(\mathbf{x})\right)^2$$
- Logarithmic Loss
$$L(y, \pi(x)) = -\sum_{k=1}^{g} \mathbb{1}_{\{y=k\}} \log\left(\pi_k(\mathbf{x})\right)$$

Softmax Regression

Softmax regression gives us a **linear classifier**.
We have g linear discriminant functions

$$f_k(\mathbf{x}) = \boldsymbol{\theta}_k^\top \mathbf{x}, \quad k = 1, 2, ..., g,$$

each indicating the confidence in class k.

The g score functions are transformed into g probability functions by the **softmax** function $s : \mathbb{R}^g \to [0, 1]^g$

$$\pi_k(\mathbf{x}) = s(f(\mathbf{x}))_k = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x})}{\sum_{j=1}^{g} \exp(\boldsymbol{\theta}_j^\top \mathbf{x})},$$

The probabilities are well-defined: $\sum \pi_k(\mathbf{x}) = 1$ and $\pi_k(\mathbf{x}) \in [0, 1]$ for all k.

Use the multiclass **logarithmic loss**

$$L(y, \pi(\mathbf{x})) = -\sum_{k=1}^{g} \mathbb{1}_{\{y=k\}} \log\left(\pi_k(\mathbf{x})\right).$$

The softmax function is
- a smooth approximation of the arg max operation:
$s((1, 1000, 2)^T) \approx (0, 1, 0)^T$ (picks out 2nd element).
- invariant to constant offsets in the input:

$$s(f(\mathbf{x}) + \mathbf{c}) = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x} + c)}{\sum_{j=1}^{g} \exp(\boldsymbol{\theta}_j^\top \mathbf{x} + c)} = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x}) \cdot \exp(c)}{\sum_{j=1}^{g} \exp(\boldsymbol{\theta}_j^\top \mathbf{x}) \cdot \exp(c)} = s(f(\mathbf{x}))$$

Binary Reduction

Computational effort for one-vs-one is much higher than for one-vs-rest.

One-vs-Rest

Create g binary subproblems, where in each the k-th original class is encoded as $+1$, and all other classes (the **rest**) as -1.
Applying all classifiers to a sample $\mathbf{x} \in \mathcal{X}$ and predicting the label k for which the corresponding classifier reports the highest confidence:

$$\hat{y} = \arg\max_{k \in \{1,2,...,g\}} \hat{f}_k(\mathbf{x}).$$

One-vs-One

Create $\frac{g(g-1)}{2}$ binary sub-problems, where each $\mathcal{D}_{k,\tilde{k}} \subset \mathcal{D}$ only considers observations from a class-pair $y^{(i)} \in \{k, \tilde{k}\}$, other observations are omitted.
Label prediction is done via **majority voting**. We predict the label of a new $\mathbf{x}$ with all classifiers and select the class that occurred most often.

Error-Correcting Output Codes (ECOC)

Codebooks: The k-th column defines how classes of all observations are encoded in the binary subproblem / for binary classifier $f_k(\mathbf{x})$. Entry (m, i) takes values $\in \{-1, 0, +1\}$
- if 0, observations of class $y^{(i)} = m$ are ignored.
- if 1, observations of class $y^{(i)} = m$ are encoded as 1.
- if -1, observations of class $y^{(i)} = m$ are encoded as -1.

Class	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$	$f_4(\mathbf{x})$	$f_5(\mathbf{x})$	$f_6(\mathbf{x})$	$f_7(\mathbf{x})$	$f_8(\mathbf{x})$
1	-1	-1	-1	-1	1	1	1	1
2	-1	- 1	1	1	-1	-1	1	1
3	-1	1	-1	1	-1	1	-1	1

We want to maximize distances between rows, and want the distances between columns to not be too small (identical columns) or too high (complementary columns):
- Row separation: each codeword should be well-separated in Hamming distance from each of the other codewords.
- Column separation: columns should be uncorrelated

Train L binary classifiers: only few classes $g \leq 11$, exhaustive search can be performed.
Randomized hill-climbing algorithm:
- g codewords of length L are randomly drawn.
- Any pair of such random strings will be separated by a Hamming distance that is binomially distributed with mean $\frac{L}{2}$.
- Iteratively improves the code: algorithm repeatedly finds the pair of rows closest together and the pair of columns that have the most extreme distance.
- Then computes the four codeword bits where these rows and columns intersect and changes them to improve the row and column separations.
- When the procedure reaches a local maximum, the algorithm randomly chooses pairs of rows and columns and tries to improve their separation.