

I2ML :: BASICS

Data

$\mathcal{X} \subset \mathbb{R}^p$: p -dimensional **feature / input space**
Usually we assume $\mathcal{X} \equiv \mathbb{R}^p$, but sometimes, dimensions may be bounded (e.g., for categorical or non-negative features.)

$\mathcal{Y} \subset \mathbb{R}^g$: **target space**
e.g.: $\mathcal{Y} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$, $\mathcal{Y} = \{-1, +1\}$, $\mathcal{Y} = \{1, \dots, g\}$ with g classes

$\mathbf{x} = (x_1, \dots, x_p)^T \in \mathcal{X}$: **feature vector**

$y \in \mathcal{Y}$: **target / label / output**

$\mathbb{D}_n = (\mathcal{X} \times \mathcal{Y})^n \subset \mathbb{D}$: **set of all finite data sets of size n**

$\mathbb{D} = \bigcup_{n \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^n$: **set of all finite data sets**

$\mathcal{D} = ((\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})) \in \mathbb{D}_n$: **data set** with n observations

$\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}} \subset \mathcal{D}$: **data for training and testing**
(often: $\mathcal{D} = \mathcal{D}_{\text{train}} \dot{\cup} \mathcal{D}_{\text{test}}$)

$(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$: i -th **observation** or **instance**

$\mathbf{o}_k^{(i)} = (\overbrace{0, 0, \dots, \mathbf{1}}^{k-1}, \overbrace{0, 0, \dots}^{n-k}) \in \{0, 1\}^n$: **class vector** for i -th observation of class k

$\mathbb{P}_{\mathbf{xy}}$: **joint probability distribution on $\mathcal{X} \times \mathcal{Y}$**

$\pi_k = \mathbb{P}(y = k)$: **prior probability** for class k
In case of binary labels we might abbreviate: $\pi = \mathbb{P}(y = 1)$.

Model and Learner

Model / hypothesis: $f : \mathcal{X} \rightarrow \mathbb{R}^g$, $\mathbf{x} \mapsto f(\mathbf{x})$ is a function that maps feature vectors to predictions, often parametrized by $\boldsymbol{\theta} \in \Theta$ (then we write $f_{\boldsymbol{\theta}}$, or, equivalently, $f(\mathbf{x} \mid \boldsymbol{\theta})$).

$\Theta \subset \mathbb{R}^d$: **parameter space**

$\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_d) \in \Theta$: model **parameters**
Some models may traditionally use different symbols.

$\mathcal{H} = \{f : \mathcal{X} \rightarrow \mathbb{R}^g \mid f \text{ belongs to a certain functional family} \}$: **hypothesis space** – set of functions to which we restrict learning

Learner $\mathcal{I} : \mathbb{D} \times \Lambda \rightarrow \mathcal{H}$ takes a training set $\mathcal{D}_{\text{train}} \in \mathbb{D}$ and produces a model $f : \mathcal{X} \rightarrow \mathbb{R}^g$, its hyperparameters set to $\boldsymbol{\lambda} \in \Lambda$.
For a parametrized model this can be adapted to $\mathcal{I} : \mathbb{D} \times \Lambda \rightarrow \Theta$

$\Lambda = \Lambda_1 \times \Lambda_2 \times \dots \times \Lambda_\ell \subset \mathbb{R}^\ell$, where $\Lambda_j = (a_j, b_j)$, $a_j, b_j \in \mathbb{R}$, $j = 1, 2, \dots, \ell$: **hyperparameter space**

$\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_\ell) \in \Lambda$: **model hyperparameters**

$\pi_k(\mathbf{x}) = \mathbb{P}(y = k \mid \mathbf{x}) \in [0, 1]$: **posterior probability** for class k , given \mathbf{x} (in a binary case we might abbreviate: $\pi(\mathbf{x}) = \mathbb{P}(y = 1 \mid \mathbf{x})$).

$h(\mathbf{x}) : \mathbb{R}^g \rightarrow \mathcal{Y}$: **prediction function** for classification that maps class scores / posterior probabilities to discrete classes

$\epsilon = y - f(\mathbf{x})$ or $\epsilon^{(i)} = y^{(i)} - f(\mathbf{x}^{(i)})$: (i -th) **residual** in regression

$y f(\mathbf{x})$ or $y^{(i)} f(\mathbf{x}^{(i)})$: **margin** for (i -th) observation in binary classification

$\hat{y}, \hat{f}, \hat{h}, \hat{\pi}_k(\mathbf{x}), \hat{\pi}(\mathbf{x})$ and $\hat{\boldsymbol{\theta}}$
The hat symbol denotes **learned** functions and parameters.

Loss and Risk

$L : \mathcal{Y} \times \mathbb{R}^g \rightarrow \mathbb{R}$: **loss function**
Quantifies "quality" of prediction $f(\mathbf{x})$ (or $\pi_k(\mathbf{x})$) for single \mathbf{x} .

$\mathcal{R}_{\text{emp}} : \mathcal{H} \rightarrow \mathbb{R}$: **empirical risk**
The ability of a model f to reproduce the association between \mathbf{x} and y that is present in the data \mathcal{D} can be measured by the summed loss:

$$\mathcal{R}_{\text{emp}}(f) = \sum_{i=1}^n L(y^{(i)}, f(\mathbf{x}^{(i)}))$$

Learning then amounts to **empirical risk minimization** – figuring out which model \hat{f} has the smallest summed loss.
Since f is usually defined by **parameters $\boldsymbol{\theta}$** , this becomes:

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \Theta} \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \arg \min_{\boldsymbol{\theta} \in \Theta} \sum_{i=1}^n L(y^{(i)}, f(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})),$$

where $\mathcal{R}_{\text{emp}} : \Theta \rightarrow \mathbb{R}$.

Components of Learning

Learning = Hypothesis space + Risk + Optimization
 $= \mathcal{H} + \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) + \arg \min_{\boldsymbol{\theta} \in \Theta} \mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$

Regression Losses

L2 loss / squared error:

- $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$ or $L(y, f(\mathbf{x})) = 0.5(y - f(\mathbf{x}))^2$
- Convex and differentiable
- Tries to reduce large residuals (loss scaling quadratically)
- **Optimal constant model:** $\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n y^{(i)} = \bar{y}$

L1 loss / absolute error:

- $L(y, f(\mathbf{x})) = |y - f(\mathbf{x})|$
- Convex and more robust
- Non-differentiable for $y = f(\mathbf{x})$, optimization becomes harder
- **Optimal constant model:** $\hat{f}(\mathbf{x}) = \text{med}(y^{(i)})$

Classification Losses

Brier score (binary case)
 $L(y, \pi(\mathbf{x})) = (\pi(\mathbf{x}) - y)^2$ for $\mathcal{Y} = \{0, 1\}$

Log-loss / Bernoulli loss / binomial loss (binary case)
For $\mathcal{Y} = \{0, 1\}$: $L(y, \pi(\mathbf{x})) = -y \log(\pi(\mathbf{x})) - (1 - y) \log(1 - \pi(\mathbf{x}))$
For $\mathcal{Y} = \{-1, +1\}$: $L(y, \pi(\mathbf{x})) =$
For $\mathcal{Y} = \{0, 1\}$: $L(y, f(\mathbf{x})) = -y \cdot f(\mathbf{x}) + \log(1 + \exp(f(\mathbf{x})))$
For $\mathcal{Y} = \{-1, +1\}$: $L(y, f(\mathbf{x})) = \log(1 + \exp(-y \cdot f(\mathbf{x})))$

Brier score (multi-class case)
 $L(y, \pi(\mathbf{x})) = \sum_{k=1}^g (\pi_k(\mathbf{x}) - o_k)^2$

Log-loss (multi-class case)
 $L(y, \pi(\mathbf{x})) = - \sum_{k=1}^g o_k \log(\pi_k(\mathbf{x}))$
 $L(y, f(\mathbf{x})) =$

Classification

Classification usually means to construct g **discriminant functions**: $f_1(\mathbf{x}), \dots, f_g(\mathbf{x})$, so that we choose our class as
 $h(\mathbf{x}) = \arg \max_{k \in \{1, \dots, g\}} f_k(\mathbf{x})$

Linear Classifier: functions $f_k(\mathbf{x})$ can be specified as linear functions

Binary classification: If only 2 classes ($\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, +1\}$) exist, we can use a single discriminant function $f(\mathbf{x}) = f_1(\mathbf{x}) - f_2(\mathbf{x})$.