IntroML - Lecture Notes Week 4

Ruben Schenk, ruben.schenk@inf.ethz.ch

March 24, 2022

1 Classification

1.1 Classification Problem

We start by comparing regression and classification:

- **Regression:** Labeling $x \to y \in Y$ where Y is a real value in $\mathbb{R}$
- **Classification:** Labeling $x \to y \in Y$ where Y is a finite, discrete set, e.g. $\{1, 2, ..., K\}$ or $\{-1, +1\}$.

For classification, inputs x and outputs y may look as follows:

X	**Y**
Documents	Sentiment (among K)
Images	Object (among K)
Emails	Spam (yes or no)
Medical data	At risk (yes or no)

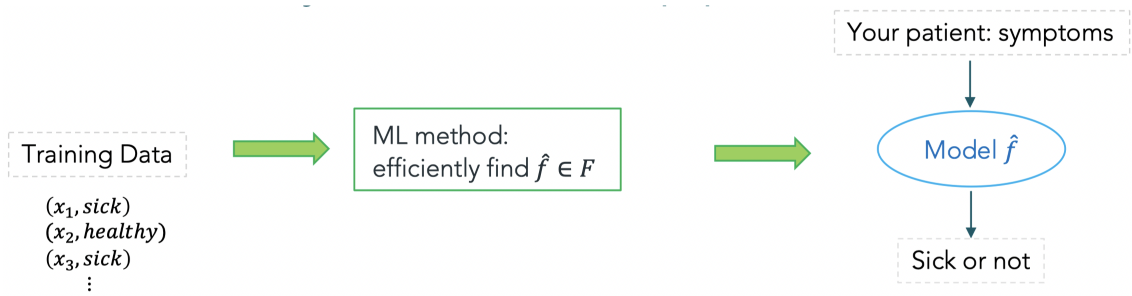
Table 1: Examples of classifications.

We distinguish the first two and the last two examples into *multiclass classification* and *binary classification.*

1.2 Losses for Classification

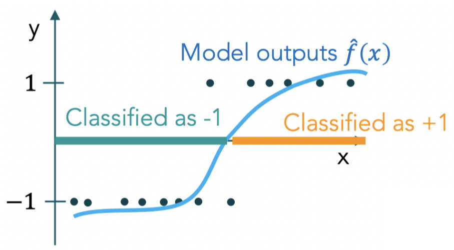
1.2.1 Evaluation

Again, we first introduce the **ML (Binary) Classification Pipeline:**



Binary classification is done using a function $\hat{f} : \mathbb{R}^d \to \mathbb{R}$:

- First, assign a number to the labels, for example "sick" = class 1, "healthy" = class 0
- One could choose these values at random w.l.o.g., but for binary classification it's often convenient to use +1 and -1
- Then, the predicted class is $\hat{y} = \text{sign}\hat{f}(x)$
- A *good model* predicts $\hat{y} = y$, where y is the true label



We have already seen that the **good model** in regression is determined by, given some data that follows the model $y = f^*(x) + \epsilon = \langle w^*, x \rangle + \epsilon$ for some f^*, the average prediction (generalization) error $R(\hat{f}) := \mathbb{E}_{x,\,y} l(\hat{f}(x),\, y) = \mathbb{E}_{x,\,y}(\hat{f}(x),\, y)^2$.

How does the good model look like for classification? For data that follows the model, i.e. $y = \epsilon \text{sign} f^*(x)$ with $\epsilon \in \{-1,\, +1\}$ for some f^*, we care about the **average classification (generalized) error:**

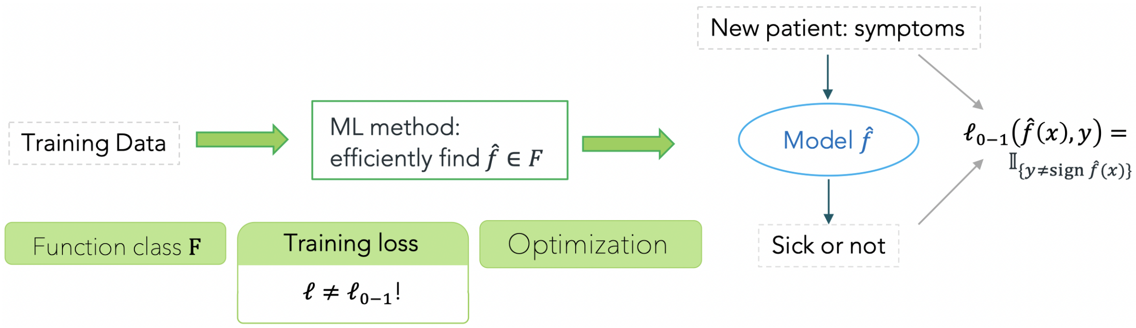
$$R(\hat{f}) := \mathbb{P}_{x,\,y}[y \neq \text{sign}\hat{f}(x)] = \mathbb{E}_{x,\,y}\mathbb{I}_{\{y \neq \text{sign}\hat{f}(x)\}} = \mathbb{E}_{x,\,y} l_{0-1}(\hat{f}(x),\, y),$$

where $l_{0-1}(\hat{f}(x),\, y) = \mathbb{I}_{\{y \neq \text{sign}\hat{f}(x)\}}$ is called the *zero-one loss* and:

$$\mathbb{I}_{\{A\}} = \begin{cases} 1, & A \text{ is true,} \\ 0, & A \text{ is false.} \end{cases}$$

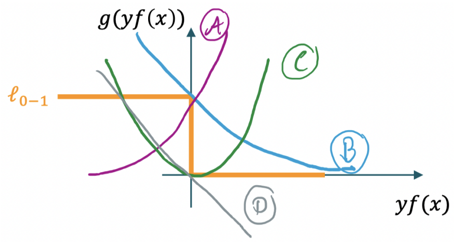
1.2.2 Training and Surrogate Losses

It is important to note that *training loss is not equal to the evaluation loss.* We have a dichotomy betnween pointwise loss we care about when predicting during test time and the pointwise surrogate loss we might use to train our model!



We would like a convex surrogate loss that also only depends on $yf(x)$. In general, we need to penaltize when $y\hat{f}(x)$ is small, or $-y\hat{f}(x)$ is big, that is, the loss function should be of type $l(\hat{f}(x),\, y) = g(y\hat{f}(x))$ for some $g(z)$ that is large when $z < 0$ and vice versa.

Therefore, in the following figure, only B and D would work:



The following functions satisfy that $g(y\hat{f}(x))$ is increasing in $-y\hat{f}(x)$:

- Exponential: $g_{exp}(y\hat{f}(x)) = e^{-y\hat{f}(x)}$
- Logistic: $g_{log}(y\hat{f}(x)) = \log(1 + e^{-y\hat{f}(x)})$
- Hinge: $g_{hinge}(y\hat{f}(x)) = \max(0,\, 1 - y\hat{f}(x))$
- Linear function: $g_{lin}(y\hat{f}(x)) = -y\hat{f}(x)$

1.2.3 Logistic Loss

We can rewrite $\hat{y} = \text{sign}[\hat{f}(x)] = \arg\max_{c\in\{-1,+1\}} c\hat{f}(x)$ since $\text{sign}\hat{f}(x) \cdot \hat{f}(x) \geq \hat{f}(x)$. If we assign 0 instead of -1 for one of the classes, we can define vector $\tilde{f}(x) := (-\hat{f}(x), \hat{f}(x))$ and then $\hat{y} = \arg\max_i(\tilde{f}(x))_{[i]}$.

For any vector $\tilde{f} \in R^K$ we define the **softmax** transformation softmax : $R^K \to R^K$ to vector $\hat{p} \in R^K$ as follows:

$$\hat{p}_{[i]} = (\text{softmax}(\tilde{f}))_{[i]} = \frac{\exp(\tilde{f}_{[i]}/2)}{\sum_{i=1}^K \exp(\tilde{f}_{[i]}/2)}$$

In particular, we can rewrite $\hat{y} = \arg\max_i(\hat{p}(x))_{[i]}$.

Furthermore, note that $\hat{p} = \text{softmax}(\tilde{f})$ is a probability vector, that is, $\hat{p}_{[i]} \geq 0$ and $\sum_{i=0}^{K-1} \hat{p}[i] = 1$! In particular, we can interpret $\hat{p}_{[i]} = \text{Prob}(y = i)$. For binary, we use $\hat{p}_0 = \text{Prob}(y = -1)$.

Using $\hat{p}_0 = \frac{1}{1+\exp(\hat{f}(x))}$ and $\hat{p}_1 = 1 - \hat{p}_0$, we can now derive logistic loss from the 0-1 loss using "probabilistic" perspective of the softmax:

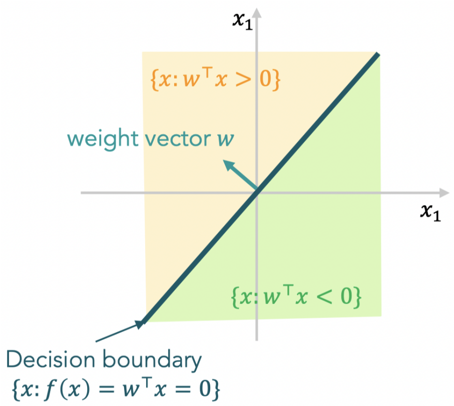
$$l_{log}(\hat{f}(x), y) = \mathbb{I}_{y=-1}\log(1 + e^{\hat{f}(x)}) + \mathbb{I}_{y=+1}\log(1 + e^{-\hat{f}(x)}) = \log(1 + e^{-y\hat{f}(x)})$$

1.3 Linear Classifiers

Linear classifiers are of the form

$$F = \{f : f(x) = w^T x \text{ for some } w \in R^d\}$$

Training and prediction is fairly simple! The decision boundary of a function f is $\{x : f(x) = 0\}$. The prediction and 0-1 eroor only depends on $\frac{w}{||w||}$ and uses $\hat{y} = \text{sign}\hat{f}(x)$, training is done using softmax$(\hat{f})$.



1.3.1 Logistic Regression

Recall the **logstic loss** to be $l_{log}(f(x), y) = \log(1 + e^{-yf(x)})$.

For linear classifiers, the **training loss** is defined as $L(w) = \frac{1}{n}\sum_{i=1}^n \log(1 + e^{-y_i w^T x_i})$ for the training points $\{(x_i, y_i)\}_{i=1}^n$.

1.3.2 Margin and Support Vector Machines

1.3.3 Multi-Class Classification

1.4 Non-Linear Classifiers