CSCI 5525 Machine Learning Fall 2019

Lecture 4: Linear Classification (Part 1)

Feb 3rd 2020

Lecturer: Steven Wu *Scribe: Steven Wu*

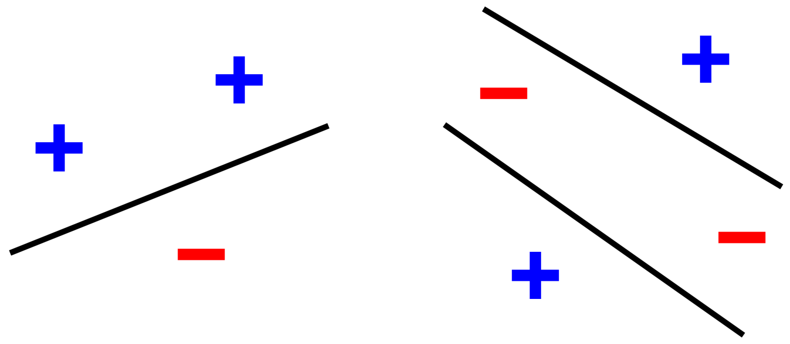
Now let's study classification, where the label space $\mathcal{Y}$ is discrete. We will mostly focus on binary classification, where $\mathcal{Y} = \{\pm 1\}$.[1] We consider *linear predictor* $\hat{f}$ parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^d$ such that on each feature vector $x \in \mathbb{R}^d$,

$$\hat{f}(x) = \text{sign}(\mathbf{w}^\intercal x)$$

A natural loss function is the 0-1 loss: $\ell(y, \hat{y}) = \mathbf{1}[y \neq \hat{y}]$. Given training data $(x_1, y_1) \ldots, (x_n, y_n)$, the ERM problem is then defined as

$$\arg\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[y \neq \hat{y}] \tag{1}$$

First, is it always possible to minimize the empirical risk down to 0 (by perfectly matching all of the labels in the data)? Well, it depends on whether the data points are linearly separable or not.



(a) Linearly Separable (b) Not linearly Separable

Figure 1: Illustration of linear separability from Wikipedia.

Feature Transformation Similar to linear regression, we can also enrich the linear predictor for classification by transforming the features. This can potentially turn linearly inseparable data into linearly separable data. For example, consider the following "XOR" dataset. The four data points are not linearly separable. (Why?)

[1] Sometimes we might also use $\mathcal{Y} = \{0, 1\}$.

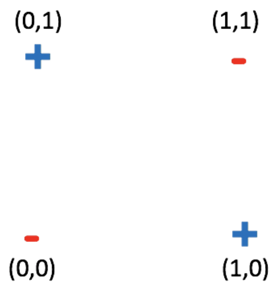


Figure 2: XOR dataset is not linearly separable.

We can then consider the following polynomial expansion:

$$\phi(x) = (1, x_1, x_2, x_1 x_2)$$

Then the predictor $\hat{f}(x) = -1 + 2x_1 + 2x_2 - 3.5x_1x_2$ that is linear in $\phi(x)$ can perfectly separate the dataset.

In particular, by applyining non-linear transformation on the "raw" features, we obtain non-linear decision boundary even with linear predictors.

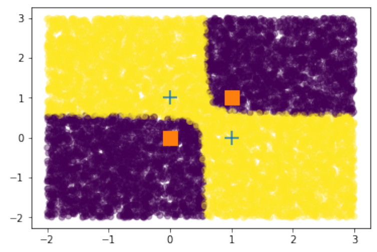


Figure 3: Visualization of the decision boundary of the linear mapping $\hat{f}$ on the transformed feature $\phi(x)$. We can then classify the XOR dataset perfectly.

Hardness of ERM The ERM objective in (1) can be further re-written as the following optimization problem:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[\text{sign}(\mathbf{w}^\intercal x_i) \neq y_i] \tag{2}$$

It looks simple, but it is actually a NP-hard problem. It means that if you have an algorithm for solving this problem with running time poly(d, n), then you could also obtain a polynomial-time algorithm for hard problems like traveling salesman problem, which is impossible unless NP $=$ P. You might want to relax the problem a bit, and hope for efficient algorithms for all possible instances, but it turns out that even the following problem is NP-Hard: suppose that you are given a dataset $\{(x_i, y_i)\}_i$ such that there exists a linear predictor that correctly labels 99% of the examples, find a linear predictor that correctly labels 51% of the examples [1].

Margin The quantity $z_i = y_i(\mathbf{w}^\intercal x_i)$ is called the *margin* of $\mathbf{w}$ on example (x_i, y_i). Let $\ell_{01}(z) = \mathbf{1}[z \le 0]$. Then we can rewrite the objective in (2) as follows

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[y_i(\mathbf{w}^\intercal x_i) \le 0] = \frac{1}{n}\sum_{i=1}^{n} \ell_{01}(z). \tag{3}$$

1 Relaxing 0-1 Loss

To obtain efficient algorithms, we will replace the zero-one loss ℓ_{01} with other *surrogate loss* functions that are *convex*. Examples include:

- Hinge loss (in SVM): $\ell_{\text{hin}} = \max\{0, 1 - z\}$
- Logisitic loss (in logistic regression): $\ell_{\text{log}} = \ln(1 + \exp(-z))$
- Exponential loss (in AdaBoost): $\ell_{\text{exp}}(z) = \exp(-z)$

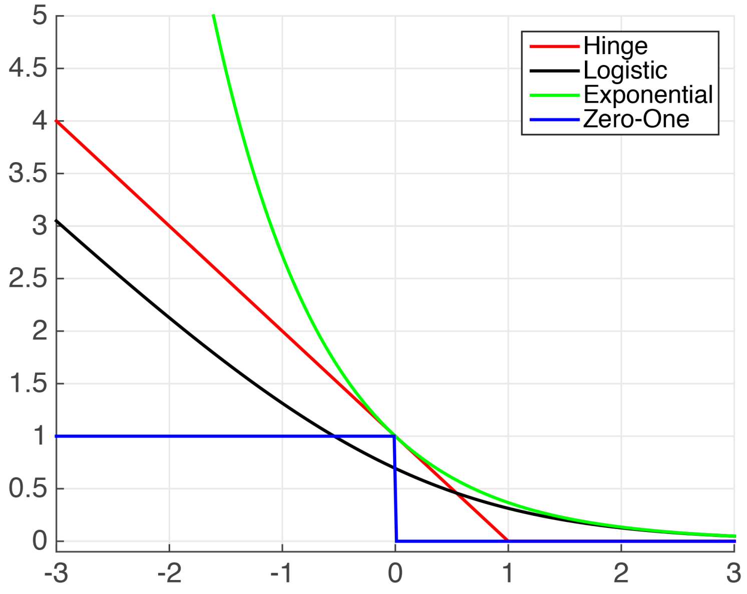


Figure 4: Visualization of the different surrogate losses. Image source.

Unlike the least squares regression solution (that is $A^{+}\mathbf{b}$) from previous lectures, the problem of minimizing the empirical risk $\hat{\mathcal{R}}$ with these loss functions does not admit closed-form solution in general. However, their convexity structure does allow us to apply convex optimization methods.

2 Gradient Descent

Given a function $F\colon \mathbb{R}^d \to \mathbb{R}$, *gradient descent* is an iterative method that udpates the parameter as follows:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta_t \nabla_{\mathbf{w}} F(\mathbf{w}^t)$$

where $\mathbf{w}^0$ is the initial point and η_t is the *learning rate*.

For provable convergence, we will set η_t depending on properties on the function F, including Lipschitz constant. In practice, we try different small values. Note that learning rate is another example of hyperparameters.

For example, in the case of logistic regression, the empirical risk $\hat{\mathcal{R}}_{\log}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell_{\log}(z_i)$ is just another example such function F. Now go ahead and solve homework1.

3 Convexity

The gradient descent method provides provable convergence/optimization guarantee for the class of convex functions. Let us formally define it here.

A set $C \subseteq \mathbb{R}^d$ is convex if all any two points $x_1, x_2 \in C$, the point $\lambda x_1 + (1-\lambda)x_2 \in C$ for any $\lambda \in [0,1]$.

A function $F\colon C \to \mathbb{R}$ is convex if all any two points $x_1, x_2 \in C$, and any $\lambda \in [0,1]$:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$

References

[1] V. Feldman, V. Guruswami, P. Raghavendra, and Y. Wu. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590, 2012.