Linear classification and optimisation

Machine Learning II (2023-2024)
UMONS

Background

Gradient. Consider a function $r(z)$ where $\boldsymbol{z} = [z_1, z_2, \ldots, z_d]^T$ is a d-vector. The gradient vector of this function is given by the partial derivatives with respect to each of the independent variables,

$$\nabla r(\boldsymbol{z}) \equiv g(\boldsymbol{z}) \equiv \begin{bmatrix} \frac{\partial r}{\partial z_1}(\boldsymbol{z}) \\ \frac{\partial r}{\partial z_2}(\boldsymbol{z}) \\ \vdots \\ \frac{\partial r}{\partial z_d}(\boldsymbol{z}) \end{bmatrix}.$$

The gradient is the generalization of the concept of derivative, which captures the local rate of change in the value of a function in multiple directions. It is very important to remember that the gradient of a function is only defined if the function is real-valued, that is, if it returns a scalar value. If the gradient exists at every point, the function is said to be differentiable. If each entry of the gradient is continuous, we say the function is once continuously differentiable.

Hessian. While the gradient of a function of d variables (i.e. the "first derivative") is an d-vector, the "second derivative" of an d-variable function is defined by d^2 partial derivatives (the derivatives of the d first partial derivatives with respect to the d variables):

$$\frac{\partial r}{\partial z_i}\left(\frac{\partial r}{\partial z_j}\right) = \frac{\partial^2 r}{\partial z_i \partial z_j}, \quad i \neq j \qquad \text{and} \qquad \frac{\partial r}{\partial z_i}\left(\frac{\partial r}{\partial z_i}\right) = \frac{\partial^2 r}{\partial^2 z_i}, \quad i = j$$

where $i, j = 1, \ldots, d$.

If r is single-valued and the partial derivatives $\frac{\partial r}{\partial z_i}$, $\frac{\partial r}{\partial z_j}$ and $\frac{\partial^2 r}{\partial z_i \partial z_j}$ are continuous, then $\frac{\partial^2 r}{\partial z_i \partial z_j}$ exists and $\frac{\partial^2 r}{\partial z_i \partial z_j} = \frac{\partial^2 r}{\partial z_j \partial z_i}$. Therefore the second-order partial derivates can be represented by a square symmetric matrix called **Hessian** matrix:

$$\nabla^2 r(\boldsymbol{z}) \equiv H(\boldsymbol{z}) \equiv \begin{pmatrix} \frac{\partial^2 r}{\partial^2 z_1}(\boldsymbol{z}) & \cdots & \frac{\partial^2 r}{\partial z_1 \partial z_d}(\boldsymbol{z}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 r}{\partial z_d \partial z_1}(\boldsymbol{z}) & \cdots & \frac{\partial^2 r}{\partial^2 z_d}(\boldsymbol{z}) \end{pmatrix},$$

which contains $d(d+1)/2$ independent elements.

If a function has a Hessian matrix at every point, then the function is called twice differentiable. If each entry of the Hessian is continuous, then the function is called twice continuously differentiable.

L-smooth function. A continuously differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if $\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|$.

Suppose f is twice continuously differentiable convex function.

f is L-smooth $\iff$ $x \mapsto \frac{L}{2}\|x\|^2 - f(x)$ is convex $\iff$ $\nabla^2 f(x) \preceq LI$, where I is the identity matrix.

1 Exercise 1

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a differentiable function $f : \mathbb{R}^m \to \mathbb{R}$. Let $h : \mathbb{R}^n \to \mathbb{R}$ defined by $h(x) = f(Ax)$. Find $\nabla h(x)$ and $\nabla^2 h(x)$.

2 Exercise 2

Consider pointwise error measures $e_{class}(s, y) = [\![y \neq sign(s)]\!]$, $e_{sq}(s, y) = (y - s)^2$, and $e_{log}(s, y) = \ln(1 + \exp(-ys))$, where the signal $s = \mathbf{w}^T\mathbf{x}$.

(a) For $y = +1$, plot e_{class}, e_{sq} and $\frac{1}{\ln 2}e_{log}$ versus s, on the same plot.

(b) Show that $e_{class}(s, y) \leq e_{sq}(s, y)$, and hence that the classification error is upper bounded by the squared error.

(c) Show that $e_{class} \leq \frac{1}{\ln 2}e_{log}(s, y)$, and, as in part (b), get an upper bound (up to a constant factor) using the logistic regression error.

These bounds indicate that minimizing the squared or logistic regression error should also decrease the classification error, which justifies using the weights returned by linear or logistic regression as approximations for classification.

3 Exercise 3

The output of the final hypothesis $g(\mathbf{x})$ learned by a probabilistic classifier can be thresholded to get a 'hard' (± 1) classification. The problem shows how to use a risk matrix to obtain such a threshold.

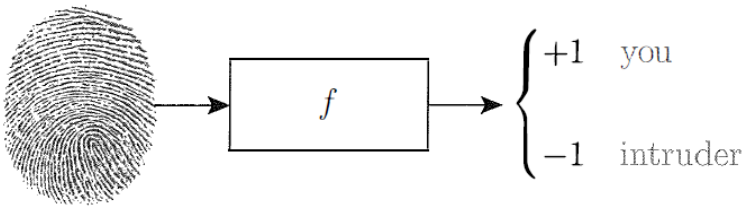

Figure 1: Fingerprint verification (Abu-Mostafa et al., 2012)

Consider fingerprint verification problem as shown in Figure 1. After learning by a probabilistic classifier from the data, you produce the final hypothesis

$$g(\mathbf{x}) = \mathbb{P}[y = +1|\mathbf{x}],$$

which is your estimate of the probability that $y = +1$. Suppose that the cost matrix is given by

		True classification	
		$+1$ (correct person)	-1 (intruder)
you say	$+1$	0	c_a
	-1	c_r	0

For a new person with fingerprint $\mathbf{x}$, you can compute $g(\mathbf{x})$ and you now need to decide whether to accept or reject the person (i.e., you need a hard classification). So, you will accept if $g(\mathbf{x}) \geq \kappa$ where κ is the threshold.

(a) Define the cost(accept) as your expected cost if you accept the person. Similarly define cost(reject). Show that

$$\begin{aligned} \text{cost(accept)} &= (1 - g(\mathbf{x}))c_a, \\ \text{cost(reject)} &= g(\mathbf{x})c_r. \end{aligned}$$

(b) Use part (a) to derive a condition on $g(\mathbf{x})$ for accepting the person and hence show that

$$\kappa = \frac{c_a}{c_a + c_r}$$

(c) Now, consider two potential clients of this fingerprint system. One is a supermarket who will use it at the checkout counter to verify that you are a member of a discount program. The other is the CIA who will use it at the entrance to a secure facility to verify that you are authorized to enter that facility.

For the supermarket, a false reject is costly because if a customer gets wrongly rejected, she may be discouraged from patronizing the supermarket in the future. On the other hand, the cost of a false accept is minor. You just gave away a discount to someone who didn't deserve it.

For the CIA, a false accept is a disaster. An unauthorized person will gain access to a highly sensitive facility. This should be reflected in a much higher cost for the false accept. False rejects, on the other hand, can be tolerated since authorized persons are employees.

The costs of the different types of errors can be tabulated in a matrix. For our examples, the matrices might look like:

Table 1: Supermarket

		f	
		$+1$	-1
g	$+1$	0	1
	-1	10	0

Table 2: CIA

		f	
		$+1$	-1
g	$+1$	0	1000
	-1	1	0

Now, compute threshold κ for each of these two cases. Give some intuition for the thresholds you get.

4 Exercise 4

For logistic regression,

$$E_{in}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}).$$

show that

$$\begin{aligned}\nabla E_{\text{in}}(\mathbf{w}) &= -\frac{1}{n}\sum_{i=1}^{N} \frac{y_i \mathbf{x}_i}{1 + e^{y_i \mathbf{w}^T \mathbf{x}_i}} \\ &= \frac{1}{n}\sum_{i=1}^{n} -y_i \mathbf{x}_i \theta(-y_i \mathbf{w}^T \mathbf{x}_i)\end{aligned}$$

and find the Hessian of $E_{in}(\mathbf{w})$.

5 Exercise 5

For logistic regression,

$$E_{in}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}).$$

Prove that E_{in} is an L-smooth function and determine L.

6 Exercise 6

Adaline (Adaptive Linear Neuron) algorithm for classification works like this: In each iteration, pick a random $(\mathbf{x}(t), y(t))$ and compute the 'signal' $s(t) = \mathbf{w}^T(t)\mathbf{x}(t)$. If $y(t) \cdot s(t) \leq 1$, update $\mathbf{w}$ by

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + \eta \cdot (y(t) - s(t)) \cdot \mathbf{x}(t),$$

where η is a constant. That is, if $s(t)$ agrees with $y(t)$ well (their product is > 1), the algorithm does nothing. On the other hand, if $s(t)$ is further from $y(t)$, the algorithm changes $\mathbf{w}(t)$ more. Here, we derive Adaline from an optimisation perspective.

(a) Consider $e_i(\mathbf{w}) = (\max(0, 1 - y_i \mathbf{w}^T \mathbf{x}))^2$. Write down the gradient $\nabla e_i(\mathbf{w})$.

(b) Show that $e_i(\mathbf{w})$ is an upper bound for $[\![sign(\mathbf{w}^T \mathbf{x}_i) \neq y_i]\!]$. Hence, $\frac{1}{n}\sum_{i=1}^{n} e_n(\mathbf{w})$ is an upper bound for the in sample classification error $E_{in}(\mathbf{w})$.

(c) Argue that the Adaline algorithm performs stochastic gradient descent on $\frac{1}{n}\sum_{i=1}^{n} e_i(\mathbf{w})$. Note that the weight update rule for SGD is

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \eta \nabla e_i \mathbf{w}.$$