

Notes from *Learning From Data*

David Lago

July 19, 2014

1 The learning problem

Components of learning:

1. The **input** $\mathbf{x} \in \mathcal{X}$
2. The **output** $y \in \mathcal{Y}$
3. The *unknown* **target function** $f: \mathcal{X} \rightarrow \mathcal{Y}$
4. The **training dataset** $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, where $y_i = f(\mathbf{x}_i) \forall i = 1, \dots, N$
5. The **hypothesis set** \mathcal{H} , which is a set of functions $h: \mathcal{X} \rightarrow \mathcal{Y}$
6. The **learning algorithm** \mathcal{A} , which based on the training dataset \mathcal{D} chooses from among all $h \in \mathcal{H}$ the one that best approximates f
7. The **final hypothesis** $g \approx f$, as chosen by \mathcal{A}

We are going to call \mathcal{A} and \mathcal{H} our **learning model**, since they are the only two components we have control over.

One simple learning model is the **perceptron model**, whose hypothesis functions are of the form:

$$h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^d w_i x_i + b\right), \quad (1)$$

where

- x_1, \dots, x_d are the components of vector \mathbf{x}
- $y \in \mathcal{Y} = \{+1, -1\}$
- b is the threshold ($y = +1 \Rightarrow \sum_{i=1}^d w_i x_i > -b$)

The **perceptron learning algorithm**, or PLA, will look for different h 's by varying the weights vector \mathbf{w} and the bias b . To simplify equation 1, we can add b to the weights vector as w_0 , so that $\mathbf{w} = [w_0, w_1, \dots, w_d]^T$, and add $x_0 = 1$ to \mathbf{x} so that now $\mathbf{x} = [1, x_1, \dots, x_d]^T$ (x and w are *column vectors*). With these changes, the perceptron hypothesis equation can be reduced to:

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x}). \quad (2)$$

PLA is an iterative algorithm. Its steps are:

1. Initialize \mathbf{w} (any value, $\mathbf{w} = [0, \dots, 0]^T$ for example)
2. Calculate $h(\mathbf{x})$ with current \mathbf{w}
3. Compare $h(\mathbf{x}_i)$ with y_i for all $i = 1, \dots, N$
4. If no mismatches, return current h as our g
5. Otherwise, pick one of the misclassified \mathbf{x} 's and update the weights vector:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y(t)\mathbf{x}(t).$$

Exercise 1.3

Show that $y(t)\mathbf{w}^T(t)\mathbf{x}(t) < 0$ for a misclassified point:

$$\begin{aligned} y(t)\mathbf{w}^T(t)\mathbf{x}(t) &= -\text{sign}(\overbrace{\mathbf{w}^T(t)\mathbf{x}(t)}^a) \overbrace{\mathbf{w}^T(t)\mathbf{x}(t)}^a \\ &= -\text{sign}(a)a \\ &< 0. \end{aligned}$$

Show that $y(t)\mathbf{w}^T(t+1)\mathbf{x}(t) > y(t)\mathbf{w}^T(t)\mathbf{x}(t)$:

$$\begin{aligned} y(t)\mathbf{w}^T(t+1)\mathbf{x}(t) &> y(t)\mathbf{w}^T(t)\mathbf{x}(t) \\ y(t)(\mathbf{w}^T(t) + y(t)\mathbf{x}^T(t))\mathbf{x}(t) &> y(t)\mathbf{w}^T(t)\mathbf{x}(t) \\ y(t)\mathbf{w}^T(t)\mathbf{x}(t) + \underbrace{y^2(t)}_{>0} \underbrace{\mathbf{x}^T(t)\mathbf{x}(t)}_{>0} &> y(t)\mathbf{w}^T(t)\mathbf{x}(t) \\ &> 0. \end{aligned}$$

Show that $\mathbf{w}(t+1)$ is an improvement over $\mathbf{w}(t)$:

In order to show this, we have to prove that $h_{\mathbf{w}(t+1)}(\mathbf{x}_i) = y_i = -h_{\mathbf{w}(t)}(\mathbf{x}_i)$ for the misclassified \mathbf{x}_i :

$$\begin{aligned} h_{\mathbf{w}(t+1)}(\mathbf{x}_i) &= \text{sign}(\mathbf{w}^T(t+1)\mathbf{x}(t)) \\ &= \text{sign}((\mathbf{w}^T(t) + y(t)\mathbf{x}^T(t))\mathbf{x}(t)) \\ &= \text{sign}(\overbrace{\mathbf{w}^T(t)\mathbf{x}(t)}^{y(t)} + y(t)\mathbf{x}^T(t)\mathbf{x}(t)) \\ &= \text{sign}(y(t)\overbrace{(1 + \mathbf{x}^T(t)\mathbf{x}(t))}^{>0}) \\ &= \text{sign}(y(t)) \\ &= y(t). \end{aligned}$$

Exercise 1.4

The solution to this exercise has been implemented in Octave. Figures 1, 2 and 3 contain the source code:

Figure 1: ex_1_4.m

```
% This exercise creates a random target function f. Generates 20 samples, and then
    applies
% the perceptron algorithm to predict

% Generate 20 random points between -10 and 10
x=[ones(20,1) 20.*rand(20,2)-10];

% Initial w for f
w0 = [1; 2; 4];

% Obtain their f(x)
for i = 1:20
    y(i) = f(x(i,:),w0);
end
y = y';

% Plot our sample
plot(x(:,2)(y>0),x(:,3)(y>0),'ro');
hold on;
plot(x(:,2)(y<0),x(:,3)(y<0),'bx');

% Plot target f in black
axis=-10:0.1:10;
fline=(-w0(1)/w0(3))- (w0(2)/w0(3))*axis;
plot(axis,fline,'k');

% Use perceptron to calculate g
w = perceptron(x,y);

% Plot g in green
gline=(-w(1)/w(3))- (w(2)/w(3))*axis;
plot(axis,gline,'g');
legend('+1', '-1', 'f', 'g','location','north');
legend('boxoff');
hold off;
```

Figure 2: perceptron.m

```
function w = perceptron(x,y)

    % Initialize w to all zeros
    m = size(x)(1);
    n = size(x)(2);
    w=zeros(n,1);

    found = true;
    iter = 0;
    maxiter = 1000;

    while (found) % Repeat while misclassified points exist (or maxiter reached)

        found = false;
        iter++;
        if (iter == maxiter)
            break;
        end

        % Calculate output with current w
        y2 = sign(w'*x')';

        % See if there is any misclassified point
        for i=1:m
            if (y(i)~=y2(i))
                found = true;

                % Adjust weights
                w=w+(y(i)*x(i,:))';

                break;
            end
        end
    end

end
```

Figure 3: f.m

```
function y = f(x,w)

    y = sign(w'*x);
```
