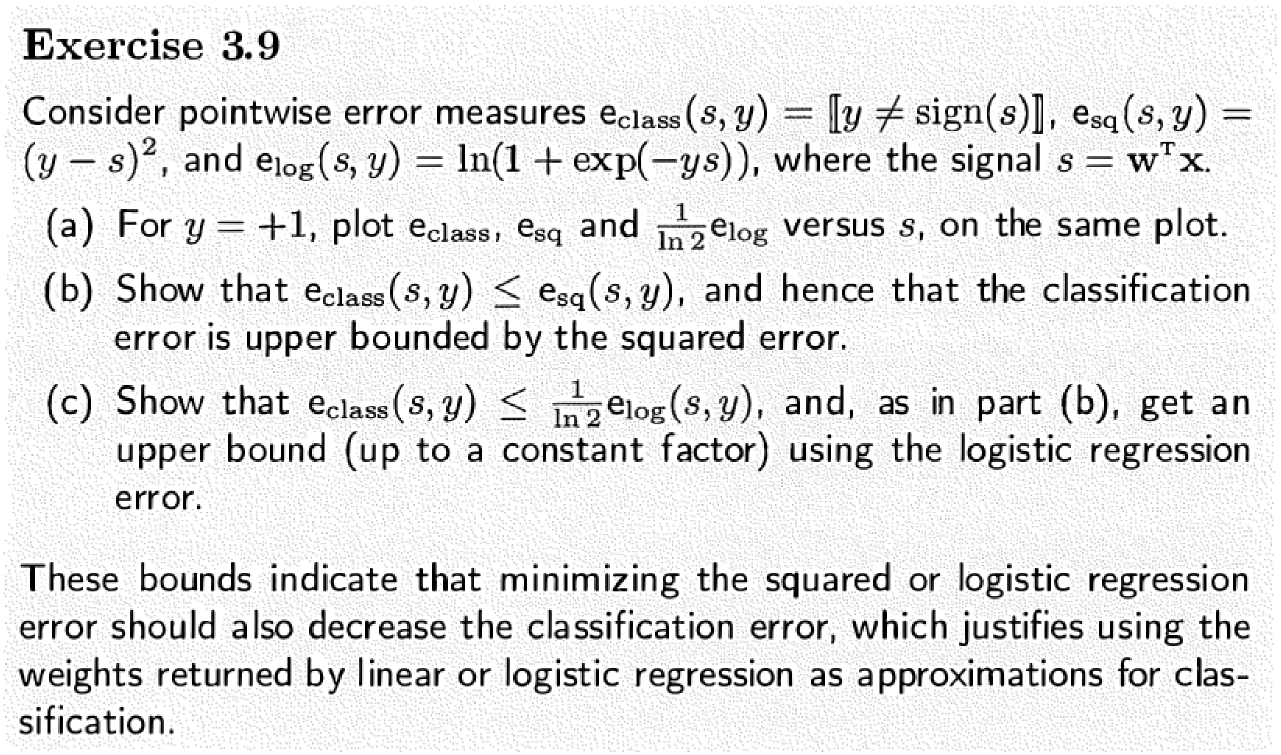
Linear classification

Machine Learning II
2021-2022 - UMONS
Souhaib Ben Taieb

1

Do Exercise 3.9 in LFD.



Exercise 3.9

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

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

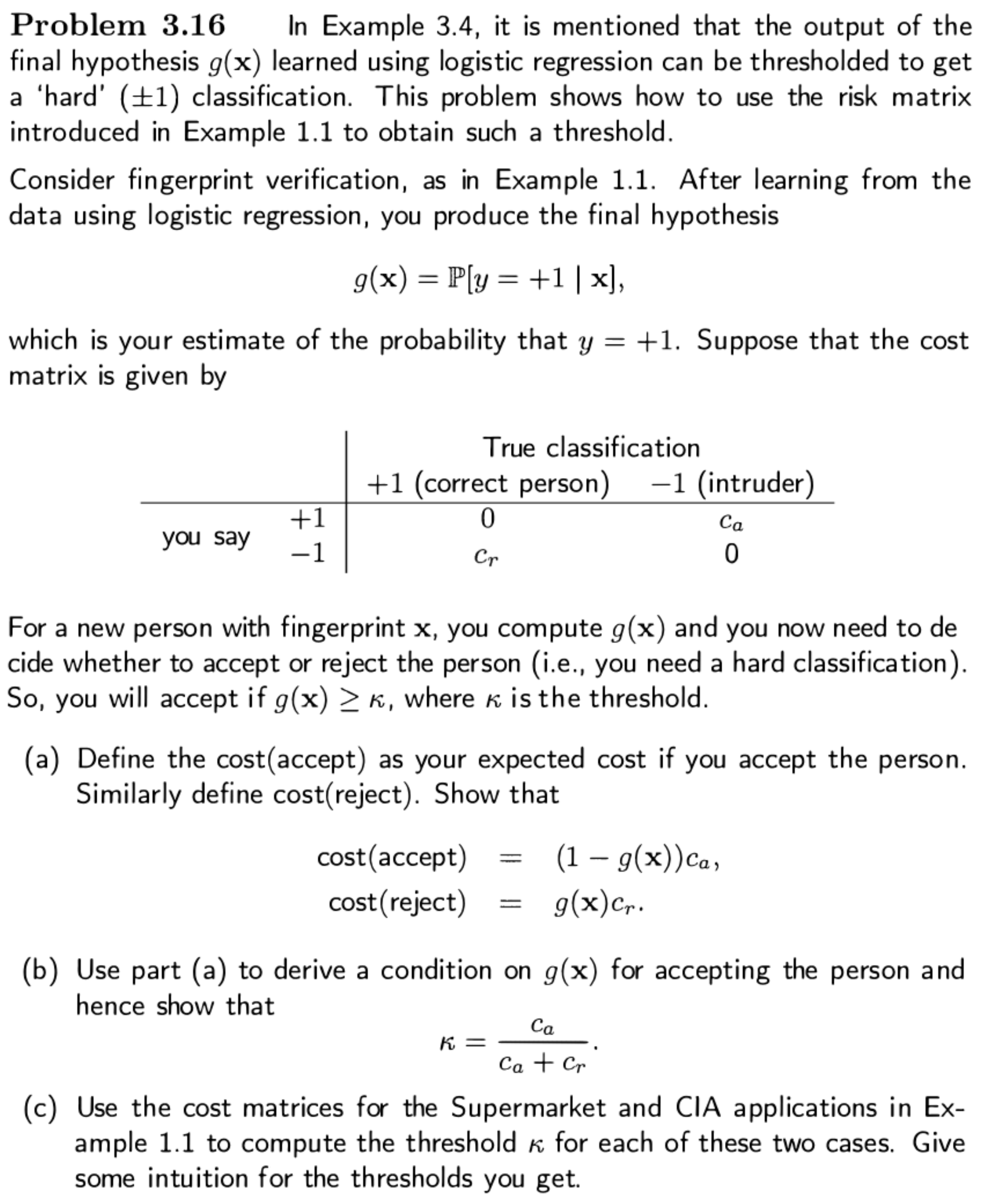
(c) Show that $e_{\text{class}}(s, y) \leq \frac{1}{\ln 2} e_{\log}(s, y)$, and, as in part (b), get an upper bound (up to a constant factor) using the logistic regression error.

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.

Figure 1: Source: Abu-Mostafa et al. Learning from data. AMLbook.

2

Solve Problem 3.16 in LFD.



Problem 3.16 In Example 3.4, it is mentioned that the output of the final hypothesis $g(\mathbf{x})$ learned using logistic regression can be thresholded to get a 'hard' (± 1) classification. This problem shows how to use the risk matrix introduced in Example 1.1 to obtain such a threshold.

Consider fingerprint verification, as in Example 1.1. After learning from the data using logistic regression, you produce the final hypothesis

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

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 compute $g(\mathbf{x})$ and you now need to de cide 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

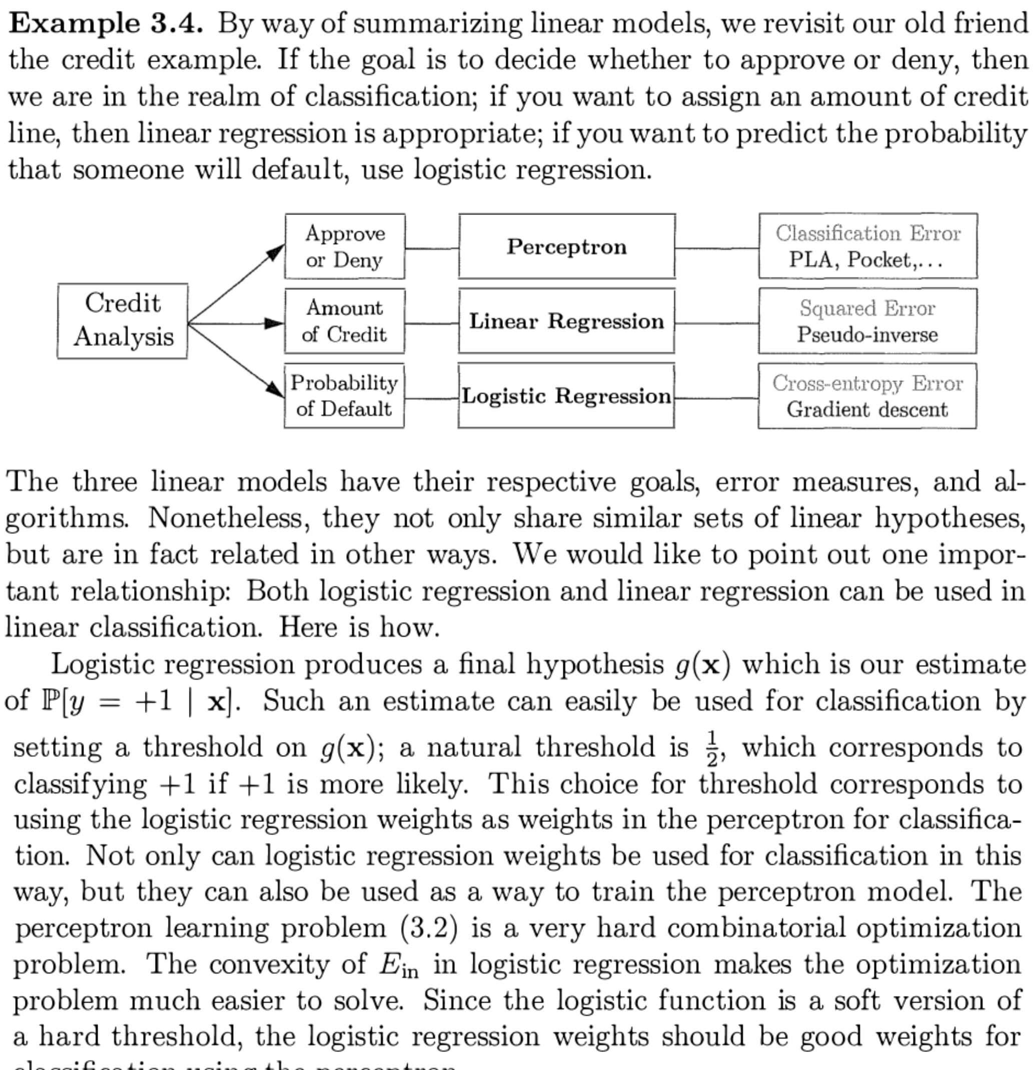
$$\begin{aligned} \text{cost(accept)} &= (1 - g(\mathbf{x}))c_a, \\ \text{cost(reject)} &= g(\mathbf{x})c_r. \end{aligned}$$

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

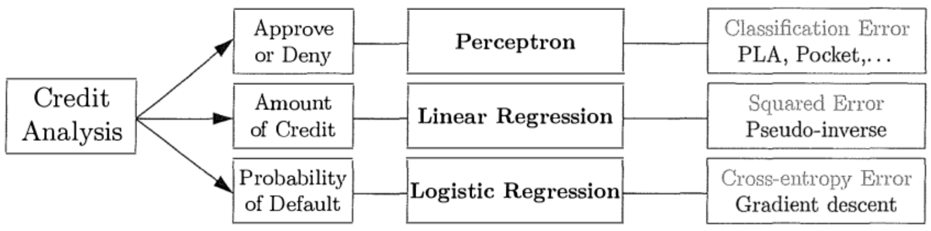
$$\kappa = \frac{c_a}{c_a + c_r}.$$

(c) Use the cost matrices for the Supermarket and CIA applications in Example 1.1 to compute the threshold κ for each of these two cases. Give some intuition for the thresholds you get.

Figure 2: Source: Abu-Mostafa et al. Learning from data. AMLbook.



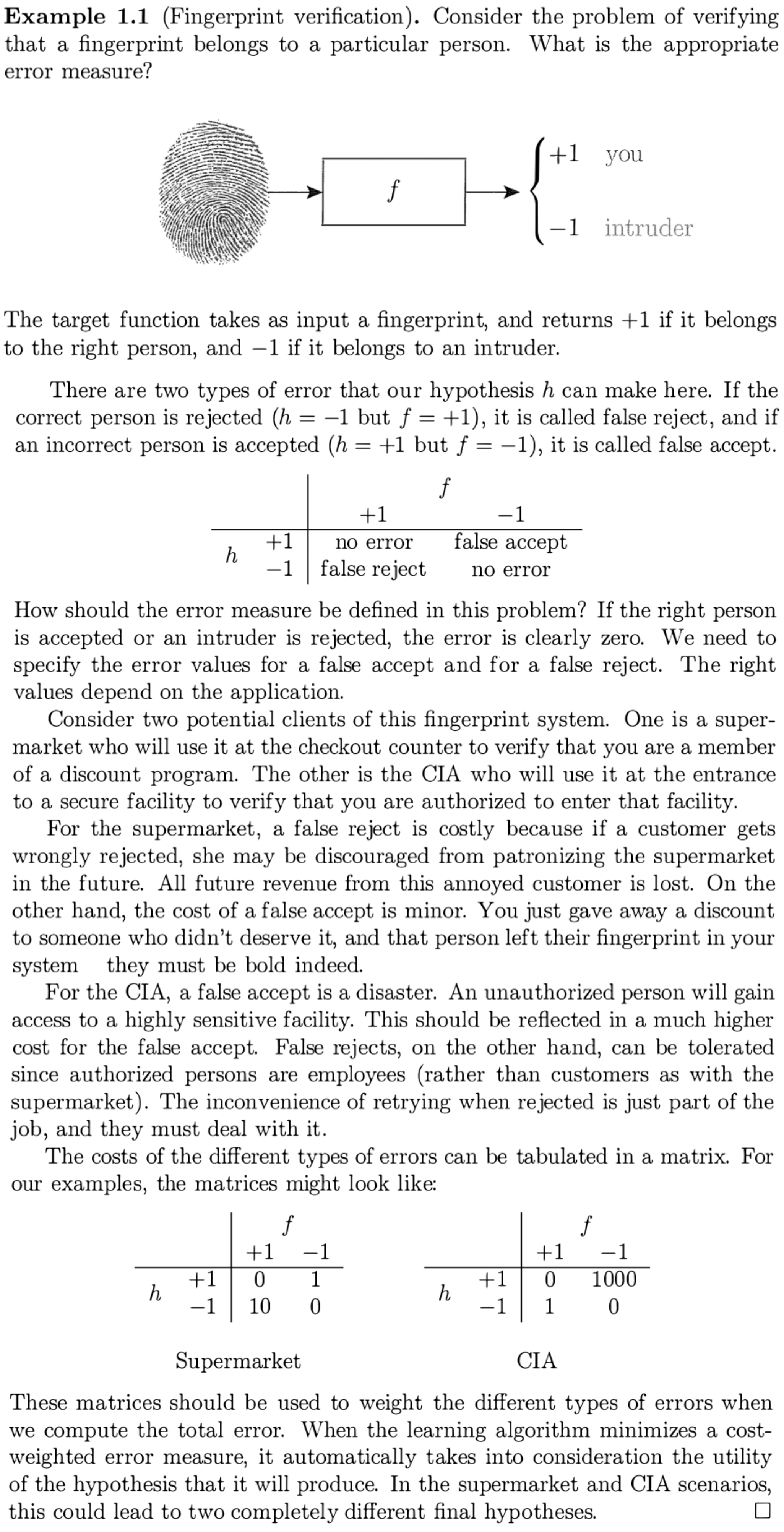
Example 3.4. By way of summarizing linear models, we revisit our old friend the credit example. If the goal is to decide whether to approve or deny, then we are in the realm of classification; if you want to assign an amount of credit line, then linear regression is appropriate; if you want to predict the probability that someone will default, use logistic regression.



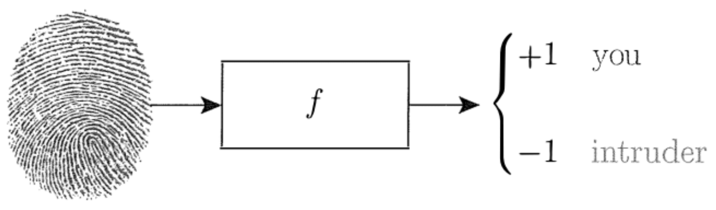
The three linear models have their respective goals, error measures, and algorithms. Nonetheless, they not only share similar sets of linear hypotheses, but are in fact related in other ways. We would like to point out one important relationship: Both logistic regression and linear regression can be used in linear classification. Here is how.

Logistic regression produces a final hypothesis $g(\mathbf{x})$ which is our estimate of $\mathbb{P}[y = +1 \mid \mathbf{x}]$. Such an estimate can easily be used for classification by setting a threshold on $g(\mathbf{x})$; a natural threshold is $\frac{1}{2}$, which corresponds to classifying $+1$ if $+1$ is more likely. This choice for threshold corresponds to using the logistic regression weights as weights in the perceptron for classification. Not only can logistic regression weights be used for classification in this way, but they can also be used as a way to train the perceptron model. The perceptron learning problem (3.2) is a very hard combinatorial optimization problem. The convexity of E_{in} in logistic regression makes the optimization problem much easier to solve. Since the logistic function is a soft version of a hard threshold, the logistic regression weights should be good weights for classification using the perceptron.

Figure 3: Source: Abu-Mostafa et al. Learning from data. AMLbook.



Example 1.1 (Fingerprint verification)**.** Consider the problem of verifying that a fingerprint belongs to a particular person. What is the appropriate error measure?



The target function takes as input a fingerprint, and returns $+1$ if it belongs to the right person, and -1 if it belongs to an intruder.

There are two types of error that our hypothesis h can make here. If the correct person is rejected ($h = -1$ but $f = +1$), it is called false reject, and if an incorrect person is accepted ($h = +1$ but $f = -1$), it is called false accept.

		f	
		$+1$	-1
h	$+1$	no error	false accept
	-1	false reject	no error

How should the error measure be defined in this problem? If the right person is accepted or an intruder is rejected, the error is clearly zero. We need to specify the error values for a false accept and for a false reject. The right values depend on the application.

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. All future revenue from this annoyed customer is lost. 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, and that person left their fingerprint in your system they must be bold indeed.

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 (rather than customers as with the supermarket). The inconvenience of retrying when rejected is just part of the job, and they must deal with it.

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

		f	
		$+1$	-1
h	$+1$	0	1
	-1	10	0

Supermarket

		f	
		$+1$	-1
h	$+1$	0	1000
	-1	1	0

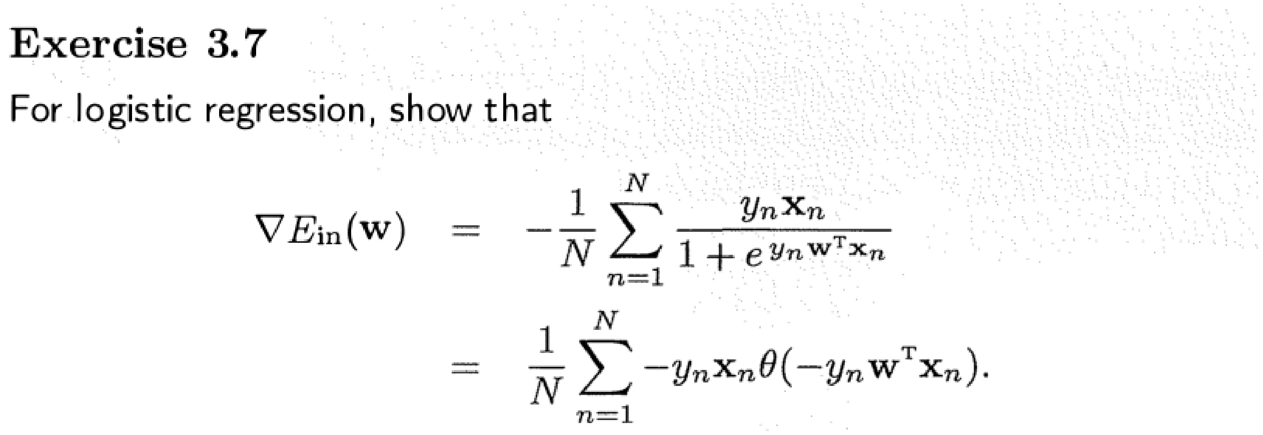
CIA

These matrices should be used to weight the different types of errors when we compute the total error. When the learning algorithm minimizes a cost-weighted error measure, it automatically takes into consideration the utility of the hypothesis that it will produce. In the supermarket and CIA scenarios, this could lead to two completely different final hypotheses. □

Figure 4: Source: Abu-Mostafa et al. Learning from data. AMLbook.

3

Do Exercise 3.7 in LFD.



Exercise 3.7

For logistic regression, show that

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

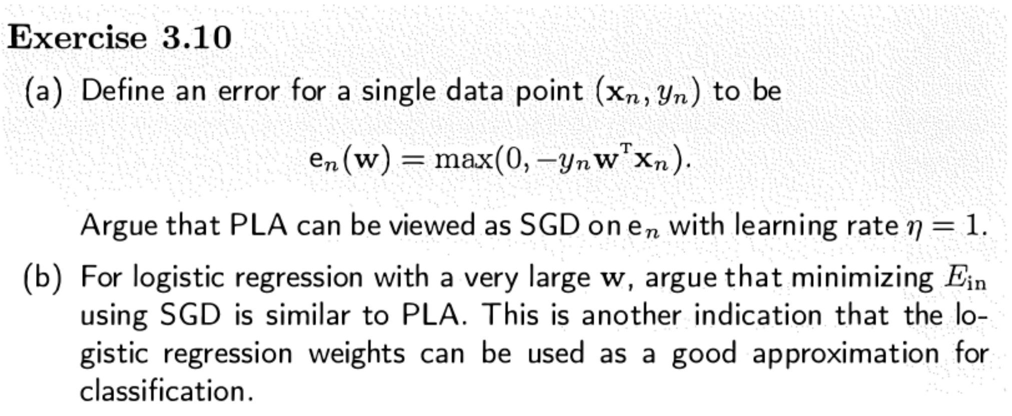
Figure 5: Source: Abu-Mostafa et al. Learning from data. AMLbook.

4

For logistic regression, show that the Newton's weight update can be rewritten as the solution of a weighted least square problem. Start by computing the Hessian of $E_{\text{in}}(\mathbf{w})$.

5

Do Exercise 3.10 in LFD.



Exercise 3.10

(a) Define an error for a single data point $(\mathbf{x}_n, y_n)$ to be

$$e_n(\mathbf{w}) = \max(0, -y_n \mathbf{w}^{\mathrm{T}} \mathbf{x}_n).$$

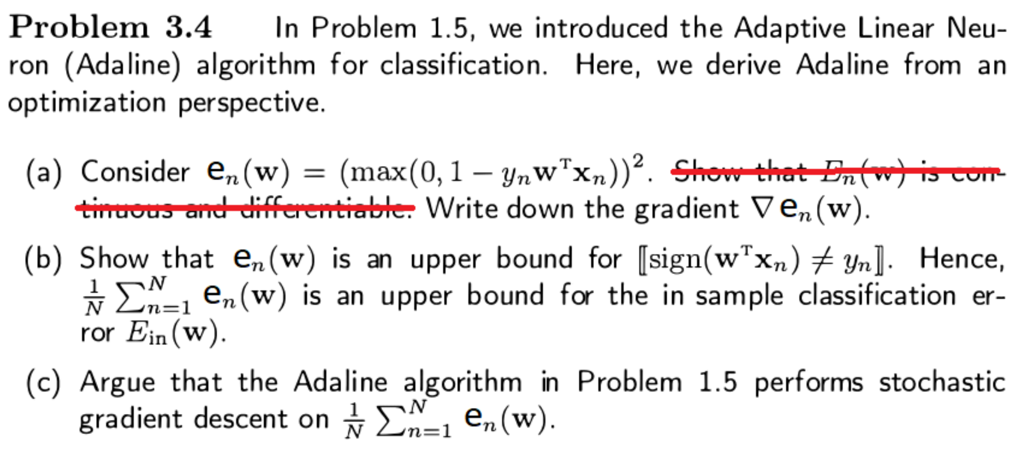
Argue that PLA can be viewed as SGD on e_n with learning rate $\eta = 1$.

(b) For logistic regression with a very large $\mathbf{w}$, argue that minimizing E_{in} using SGD is similar to PLA. This is another indication that the logistic regression weights can be used as a good approximation for classification.

Figure 6: Source: Abu-Mostafa et al. Learning from data. AMLbook.

6

Solve Problem 3.4 in LFD.



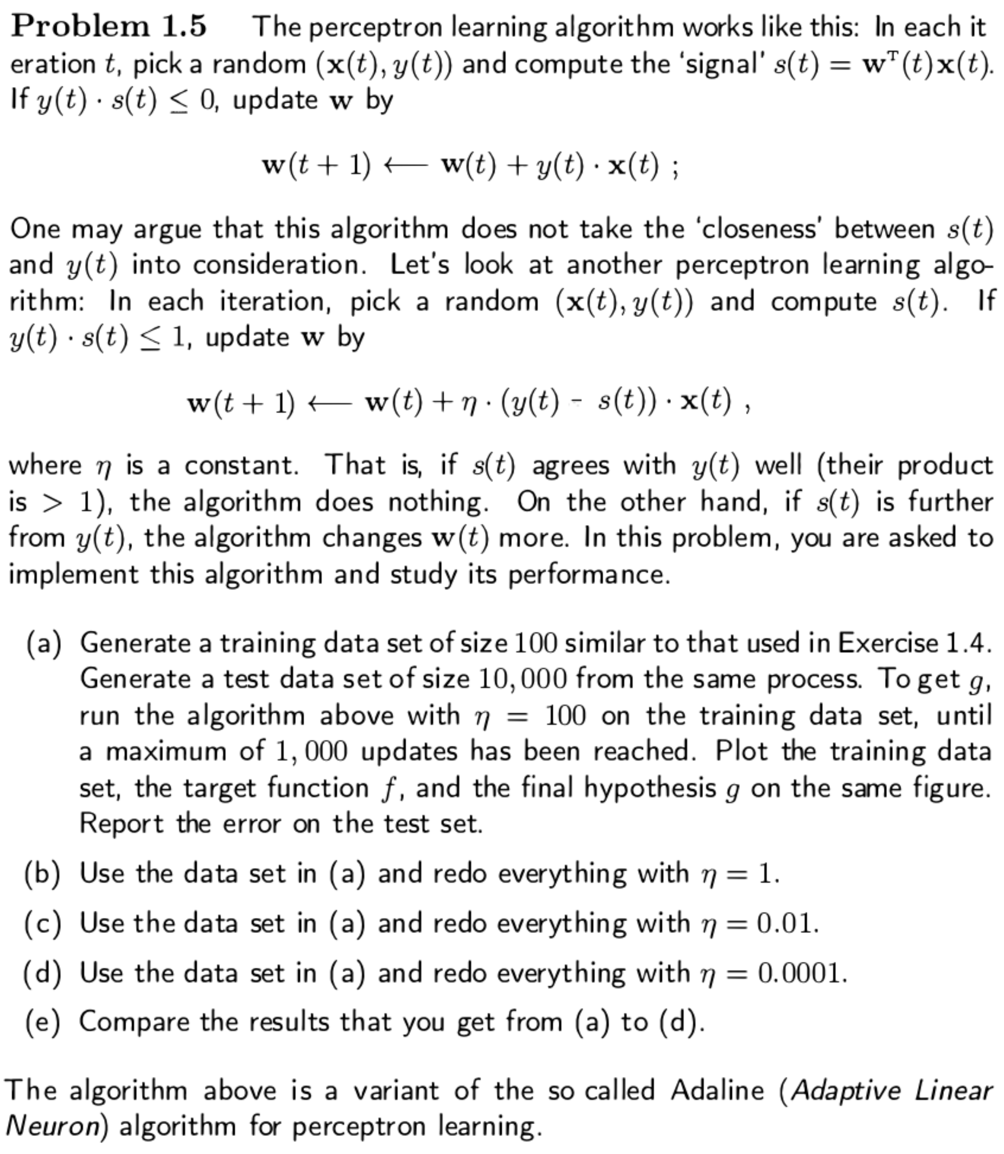
Problem 3.4 In Problem 1.5, we introduced the Adaptive Linear Neuron (Adaline) algorithm for classification. Here, we derive Adaline from an optimization perspective.

(a) Consider $\mathrm{e}_n(\mathbf{w}) = (\max(0, 1 - y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n))^2$. ~~Show that $E_n(\mathbf{w})$ is continuous and differentiable.~~ Write down the gradient $\nabla\mathrm{e}_n(\mathbf{w})$.

(b) Show that $\mathrm{e}_n(\mathbf{w})$ is an upper bound for $[\![\mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) \neq y_n]\!]$. Hence, $\frac{1}{N}\sum_{n=1}^{N}\mathrm{e}_n(\mathbf{w})$ is an upper bound for the in sample classification error $E_{\mathrm{in}}(\mathbf{w})$.

(c) Argue that the Adaline algorithm in Problem 1.5 performs stochastic gradient descent on $\frac{1}{N}\sum_{n=1}^{N}\mathrm{e}_n(\mathbf{w})$.

Figure 7: Source: Abu-Mostafa et al. Learning from data. AMLbook.



Problem 1.5 The perceptron learning algorithm works like this: In each it eration t, pick a random $(\mathbf{x}(t), y(t))$ and compute the 'signal' $s(t) = \mathbf{w}^{\mathrm{T}}(t)\mathbf{x}(t)$. If $y(t) \cdot s(t) \leq 0$, update $\mathbf{w}$ by

$$\mathbf{w}(t+1) \longleftarrow \mathbf{w}(t) + y(t) \cdot \mathbf{x}(t) \ ;$$

One may argue that this algorithm does not take the 'closeness' between $s(t)$ and $y(t)$ into consideration. Let's look at another perceptron learning algorithm: In each iteration, pick a random $(\mathbf{x}(t), y(t))$ and compute $s(t)$. If $y(t) \cdot s(t) \leq 1$, update $\mathbf{w}$ by

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

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. In this problem, you are asked to implement this algorithm and study its performance.

(a) Generate a training data set of size 100 similar to that used in Exercise 1.4. Generate a test data set of size $10,000$ from the same process. To get g, run the algorithm above with $\eta = 100$ on the training data set, until a maximum of $1,000$ updates has been reached. Plot the training data set, the target function f, and the final hypothesis g on the same figure. Report the error on the test set.

(b) Use the data set in (a) and redo everything with $\eta = 1$.

(c) Use the data set in (a) and redo everything with $\eta = 0.01$.

(d) Use the data set in (a) and redo everything with $\eta = 0.0001$.

(e) Compare the results that you get from (a) to (d).

The algorithm above is a variant of the so called Adaline (*Adaptive Linear Neuron*) algorithm for perceptron learning.

Figure 8: Source: Abu-Mostafa et al. Learning from data. AMLbook.