## Artificial Intelligence II: further notes on machine learning

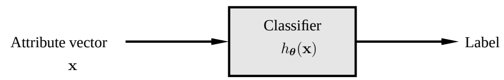
We now look at several issues that need to be considered when *applying machine learning algorithms in practice*:

- We often have more examples from some classes than from others.
- The *obvious* measure of performance is not always the *best*.
- Much as we'd love to have an optimal method for *finding hyperparameters*, we don't have one, and it's *unlikely that we ever will*.
- We need to exercise care if we want to claim that one approach is superior to another.



## Supervised learning

As usual, we want to design a *classifier*.




It should take an attribute vector

$$\mathbf{x}^T = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}$$

and label it.

We now denote a classifier by $h_{\boldsymbol{\theta}}(\mathbf{x})$ where

$$\boldsymbol{\theta}^T = \begin{pmatrix} \mathbf{w} & \mathbf{p} \end{pmatrix}$$

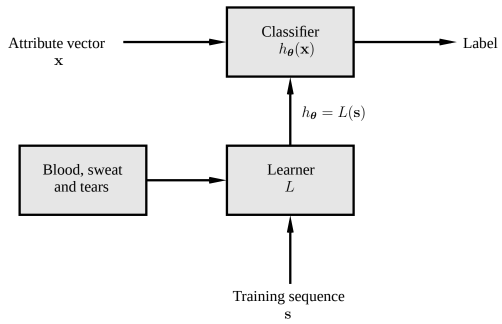
denotes any weights $\mathbf{w}$ and (hyper)parameters $\mathbf{p}$.

To keep the discussion and notation simple we assume a *classification problem* with *two classes* labelled $+1$ *(positive examples)* and $-1$ *(negative examples)*.



## Supervised learning

Previously, the learning algorithm was a box labelled $L$.



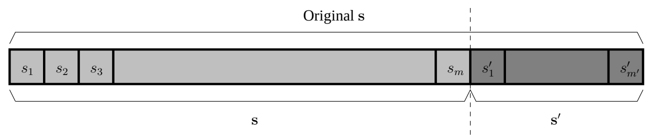

Unfortunately that turns out not to be enough, so a new box has been added.



## Measuring performance

How do you assess the performance of your classifier?

1. That is, *after training*, how do you know how well you've done?
2. In general, the only way to do this is to divide your examples into a smaller *training set* $\mathbf{s}$ of $m$ examples and a *test set* $\mathbf{s}'$ of $m'$ examples.




The *GOLDEN RULE*: *data used to assess performance must NEVER have been seen during training.*

This might seem obvious, but it was a major flaw in a lot of early work.

## Measuring performance

How do we choose $m$ and $m'$? Trial and error!

Assume the training is complete, and we have a classifier $h_{\boldsymbol{\theta}}$ obtained using only $\mathbf{s}$. How do we use $\mathbf{s}'$ to assess our method's performance?

The obvious way is to see how many examples in $\mathbf{s}'$ the classifier classifies correctly:

$$\hat{\text{er}}_{\mathbf{s}'}(h_{\boldsymbol{\theta}}) = \frac{1}{m'} \sum_{i=1}^{m'} \mathbb{I}(h_{\boldsymbol{\theta}}(\mathbf{x}_i') \neq y_i')$$

where

$$\mathbf{s}' = \begin{pmatrix} (\mathbf{x}_1', y_1') & (\mathbf{x}_2', y_2') & \cdots & (\mathbf{x}_{m'}', y_{m'}') \end{pmatrix}^T$$

and

$$\mathbb{I}(z) = \begin{cases} 1 & \text{if } z = \text{true} \\ 0 & \text{if } z = \text{false} \end{cases}.$$

This is just an estimate of the *probability of error* and is often called the *accuracy*.



## Unbalanced data

Unfortunately it is often the case that we have *unbalanced data* and this can make such a measure misleading. For example:

> If the data is naturally such that *almost all examples are negative* (medical diagnosis for instance) then simply *classifying everything as negative* gives a high performance using this measure.

We need more subtle measures.

For a classifier $h$ and any set $\mathbf{s}$ of size $m$ containing $m^+$ positive examples and $m^-$ negative examples...



## Unbalanced data

Define

1. The *true positives*
$$P^+ = \{(\mathbf{x}, +1) \in \mathbf{s} | h(\mathbf{x}) = +1\}, \text{ and } p^+ = |P^+|$$
2. The *false positives*
$$P^- = \{(\mathbf{x}, -1) \in \mathbf{s} | h(\mathbf{x}) = +1\}, \text{ and } p^- = |P^-|$$
3. The *true negatives*
$$N^+ = \{(\mathbf{x}, -1) \in \mathbf{s} | h(\mathbf{x}) = -1\}, \text{ and } n^+ = |N^+|$$
4. The *false negatives*
$$N^- = \{(\mathbf{x}, +1) \in \mathbf{s} | h(\mathbf{x}) = -1\}, \text{ and } n^- = |N^-|$$

Thus $\hat{\text{er}}_{\mathbf{s}}(h) = (p^+ + n^+)/m$.

This allows us to define more discriminating measures of performance.



## Performance measures

Some standard performance measures:

1. Precision $\frac{p^+}{p^+ + p^-}$.
2. Recall $\frac{p^+}{p^+ + n^-}$.
3. Sensitivity $\frac{p^+}{p^+ + n^-}$.
4. Specificity $\frac{n^+}{n^+ + p^-}$.
5. False positive rate $\frac{p^-}{p^- + n^+}$.
6. Positive predictive value $\frac{p^+}{p^+ + p^-}$.
7. Negative predictive value $\frac{n^+}{n^+ + n^-}$.
8. False discovery rate $\frac{p^-}{p^- + p^+}$.

In addition, plotting sensitivity (true positive rate) against the false positive rate while a parameter is varied gives the *receiver operating characteristic (ROC)* curve.

## Performance measures

The following specifically take account of unbalanced data:

1. Matthews Correlation Coefficient (MCC)

$$\mathrm{MCC} = \frac{p^+n^+ - p^-n^-}{\sqrt{(p^+ + p^-)(n^+ + n^-)(p^+ + n^-)(n^+ + p^-)}}$$

2. F1 score

$$\mathrm{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

When data is unbalanced these are preferred over the accuracy.



## Validation and crossvalidation

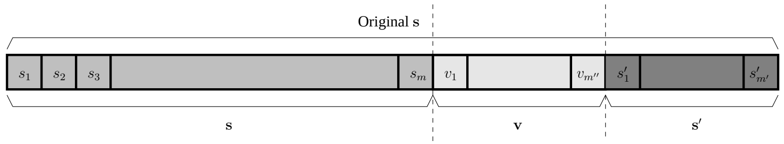
The next question: how do we choose hyperparameters?

Answer: *try different values and see which values give the best (estimated) performance*.

There is however a problem:

If I use my test set $\mathbf{s}'$ to find good hyperparameters, *then I can't use it to get a final measure of performance*. (See the Golden Rule above.)

Solution 1: make a further division of the complete set of examples to obtain a third, *validation* set:






## Validation and crossvalidation

Now, to choose the value of a hyperparameter $p$:

For some range of values $p_1, p_2, \ldots, p_n$

1. Run the training algorithm using training data $\mathbf{s}$ and with the hyperparameter set to $p_i$.
2. Assess the resulting $h_{\boldsymbol{\theta}}$ by computing a suitable measure (for example accuracy, MCC or F1) using $\mathbf{v}$.

Finally, select the $h_{\boldsymbol{\theta}}$ with maximum estimated performance and assess its *actual* performance using $\mathbf{s}'$.



## Validation and crossvalidation

This was originally used in a similar way when deciding the best point at which to *stop training* a neural network.


The figure shows the typical scenario.

## Crossvalidation

The method of *crossvalidation* takes this a step further.

We our complete set into training set s and testing set s′ as before.

But now instead of further subdividing s just once we divide it into $n$ *folds* $\mathrm{s}^{(i)}$ each having $m/n$ examples.


Typically $n = 10$ although other values are also used, for example if $n = m$ we have *leave-one-out* cross-validation.



## Crossvalidation

Let $\mathrm{s}_{-i}$ denote the set obtained from s by *removing* $\mathrm{s}^{(i)}$.

Let $\hat{\mathrm{er}}_{\mathbf{s}^{(i)}}(h)$ denote any suitable error measure, such as accuracy, MCC or F1, computed for $h$ using fold $i$.

Let $L_{\mathbf{s}_{-i},\mathbf{p}}$ be the classifier obtained by running learning algorithm $L$ on examples $\mathrm{s}_{-i}$ using hyperparameters $p$.

Then,

$$\frac{1}{n}\sum_{i=1}^{n} \hat{\mathrm{er}}_{\mathbf{s}^{(i)}}(L_{\mathbf{s}_{-i},\mathbf{p}})$$

is the $n$*-fold crossvalidation error estimate*.

So for example, let $\mathrm{s}_j^{(i)}$ denote the $j$th example in the $i$th fold. Then using accuracy as the error estimate we have

$$\frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m/n} \mathbb{I}(L_{\mathbf{s}_{-i},\mathbf{p}}(\mathbf{x}_j^{(i)}) \neq y_j^{(i)})$$



## Crossvalidation

Two further points:

1. What if the data are unbalanced? *Stratified crossvalidation* chooses folds such that the proportion of positive examples in each fold matches that in s.
2. Hyperparameter choice can be done just as above, using a basic search.

What happens however if we have multiple hyperparameters?

1. We can search over all combinations of values for specified ranges of each parameter.
2. This is the *standard method in choosing parameters for support vector machines (SVMs)*.
3. With SVMs it is generally limited to the case of only two hyperparameters.
4. Larger numbers quickly become infeasible.



## Comparing classifiers

Imagine I have compared the *Bloggs Classificator 2000* and the *CleverCorp Discriminotron* and found that:

1. Bloggs Classificator 2000 has estimated accuracy 0.981 on the test set.
2. CleverCorp Discriminotron has estimated accuracy 0.982 on the test set.

Can I claim that CleverCorp Discriminotron is the better classifier?

Answer:

NO! NO! NO! NO! NO! NO! NO! NO! NO!!!!!!!!!!!!!!

NO!!!!!!!

Note for next year: include photo of grumpy-looking cat.





From *Mathematical Methods for Computer Science*:

The *Central Limit Theorem*: If we have independent identically distributed (iid) random variables $X_1, X_2, \dots, X_n$ with mean

$$\mathbb{E}[X] = \mu$$

and standard deviation

$$\mathbb{E}\left[(X-\mu)^2\right] = \sigma^2$$

then as $n \to \infty$

$$\frac{\hat{X}_n - \mu}{\sigma/\sqrt{n}} \to N(0,1)$$

where

$$\hat{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$





We have tables of values $z_p$ such that if $x \sim N(0,1)$ then

$$\Pr(-z_p \leq x \leq z_p) > p.$$

Rearranging this using the equation from the previous slide we have that with probability $p$

$$\mu \in \left[\hat{X}_n \pm \frac{z_p \sigma}{\sqrt{n}}\right].$$

We don't know $\sigma$ but it can be estimated using

$$\sigma^2 \simeq \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \hat{X}_n\right)^2.$$

Alternatively, when $X$ takes only values $0$ or $1$

$$\sigma^2 = \mathbb{E}\left[(X-\mu)^2\right] = \mathbb{E}\left[X^2\right] - \mu^2 = \mu(1-\mu) \simeq \hat{X}_n(1-\hat{X}_n).$$





Now say I have classifiers $h_1$ (Bloggs Classificator 2000) and $h_2$ (CleverCorp Discriminotron) and I want to know something about the quantity

$$d = \text{er}(h_1) - \text{er}(h_2)$$

where

$$\text{er}(h) = \mathbb{E}\left[\mathbb{I}(h(\mathbf{x}) \neq y)\right]$$

is the *actual probability of error* for $h$.

Earlier, we *estimated* $\text{er}(h)$ using the *accuracy*

$$\hat{\text{er}}_{\mathbf{s}}(h) = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(h(\mathbf{x}_i) \neq y_i)$$

for a test set $\mathbf{s}$.

Say I estimate $d$ using

$$d \simeq \hat{\text{er}}_{\mathbf{s}_1}(h_1) - \hat{\text{er}}_{\mathbf{s}_2}(h_2)$$

where $\mathbf{s}_1$ and $\mathbf{s}_2$ are two independent test sets.

## Comparing classifiers

Notice:

1. The estimate of $d$ is a sum of random variables, and we can apply the central limit theorem.
2. Our estimate is *unbiased*
$$\mathbb{E}\left[\hat{\text{er}}_{\mathbf{s}_1}(h_1) - \hat{\text{er}}_{\mathbf{s}_2}(h_2)\right] = d.$$
3. The two parts of the estimate $\hat{\text{er}}_{\mathbf{s}_1}(h_1)$ and $\hat{\text{er}}_{\mathbf{s}_2}(h_2)$ are each sums of random variables.
4. The variance of the estimate is the sum of the variances of $\hat{\text{er}}_{\mathbf{s}_1}(h_1)$ and $\hat{\text{er}}_{\mathbf{s}_2}(h_2)$
5. *We can calculate a confidence interval for our estimate*.

In fact, if we are using a split into training set $\mathbf{s}$ and test set $\mathbf{s}'$ we can generally obtain $h_1$ and $h_2$ using $\mathbf{s}$ and use the estimate
$$d \simeq \hat{\text{er}}_{\mathbf{s}'}(h_1) - \hat{\text{er}}_{\mathbf{s}'}(h_2)$$



## Comparing classifiers

And finally:

1. We would typically demand a 95% confidence interval before claiming one classifier is better than another.
2. Don't assume this is the end of the story: statistical testing of this kind is a LARGE subject.
3. For example, we haven't taken account of the fact that $h_1$ and $h_2$ *also depend on the training set*.
4. To do this we need the *paired $t$-test*.