Supervised Learning[1]

David Barber

University College London

[1] These slides accompany the book *Bayesian Reasoning and Machine Learning*. The book and demos can be downloaded from `www.cs.ucl.ac.uk/staff/D.Barber/brml`. Feedback and corrections are also available on the site. Feel free to adapt these slides for your own purposes, but please include a link the above website.

Utility and Loss

- Given an input x^*, the optimal prediction depends on how costly making an error is. This is quantified using a loss function L (or utility $U = -L$).
- In forming a *decision function* $c(x^*)$ that will produce a class label for the new input x^*, we don't know the true class, only our surrogate for this, the predictive distribution $p(c|x^*)$.
- If $U(c^{true}, c^{pred})$ represents the utility of making a decision c^{pred} when the truth is c^{true}, the expected utility is

$$U(c(x^*)) = \sum_{c^{true}} U(c^{true}, c(x^*))p(c^{true}|x^*)$$

For a parametric decision function $c(x^*|\theta)$ we would have (summing over datapoints)

$$U(\theta) = \sum_n \sum_{c^{true}} U(c^{true}, c(x_n^*|\theta))p(c^{true}|x_n^*)$$

One finds the decision (or parameter θ) that maximises the expected utility.

The different philosophies

- In the above maximal expected utility framework, we need to define the utility, U, the class probability $p(c|x)$ and the function $c(x^*)$.
- However, in practice we typically don't know the correct model underlying the data – all we have is a dataset of examples $\mathcal{D} = \{(x^n, c^n), n = 1, \ldots, N\}$ (and possibly our domain knowledge).
- There are different philosophies for how to determine $p(c|x)$ and how to parameterise the decision function $c(x^*)$.
- Broadly, there are two main philosophies: Empirical Risk (empirical distribution for $p(c|x)$ and parametric $c(x^*|\theta)$) and Bayesian Decision (non-trivial $p(c|x)$ and unconstrained $c(x^*)$).
- We will discuss the benefits/drawbacks of each.
- Generally, there is no 'correct' approach and it's useful to potentially consider both, depending on the problem.

Zero-one loss/utility

A ‘count the correct predictions’ measure of prediction performance is based on the zero-one utility:

$$U(c^{true}, c^*) = \begin{cases} 1 \text{ if } c^* = c^{true} \\ 0 \text{ if } c^* \neq c^{true} \end{cases}$$

For the two class case, we then have the expected utility

$$U(c(x^*)) = \begin{cases} p(c^{true} = 1|x^*) \text{ for } c(x^*) = 1 \\ p(c^{true} = 2|x^*) \text{ for } c(x^*) = 2 \end{cases}$$

Hence, in order to have the highest expected utility, the (unconstrained) decision function $c(x^*)$ should correspond to selecting the highest class probability $p(c|x^*)$:

$$c(x^*) = \begin{cases} 1 & \text{if} \quad p(c = 1|x^*) > 0.5 \\ 2 & \text{if} \quad p(c = 2|x^*) > 0.5 \end{cases}$$

In the case of a tie, either class is selected at random with equal probability.

General utility functions

- One can readily generalise this to multiple-class situations using a *utility matrix* with elements

  $$U_{ij} = U(c^{true} = i, c^{pred} = j)$$

  where the i, j element of the matrix contains the utility of predicting class j when the true class is i.
- Then the expected utility is

  $$U(c^* = j) = \sum_i U_{ij} p(c^{true} = i | x^*)$$

  and one then finds the class c^* that maximises this expected utility.

Asymmetric Utility

- In some applications the utility matrix is highly non-symmetric.
- Consider a medical scenario in which we are asked to predict whether or not the patient has cancer $\mathrm{dom}(c) = \{\text{cancer}, \text{benign}\}$.
- If the true class is cancer yet we predict benign, this could have terrible consequences for the patient. On the other hand, if the class is benign yet we predict cancer, this may be less disastrous for the patient.
- Such asymmetric utilities can favour conservative decisions – in the cancer case, we would be more inclined to decide the sample is cancerous than benign, even if the predictive probability of the two classes is equal.

Squared loss/utilty

- In regression problems, for a real-valued prediction y^{pred} and truth y^{true}, a common loss function is the squared loss

$$L(y^{true}, y^{pred}) = \left(y^{true} - y^{pred}\right)^2$$

- The above decision framework then follows through, replacing summation with integration for the continuous variables.
- For an output prediction given an input x,

$$L(y^{pred}) = \int \left(y^{true} - y^{pred}\right)^2 p(y^{true}|x) dy^{true}$$

In the unconstrained case, this loss is minimized by setting

$$y^{pred} = \int y^{true} p(y^{true}|x) dy^{true}$$

Other loss functions

- It's worth noting that some regression problems have non-squared loss (for example the summed absolute loss) in which case optimal predictor won't necessarily be simply the average of the prediction distribution.
- It's very common to train models using the squared loss. However, if the task performance is measured by some other loss, it often makes more sense to train the model using the correct loss. One often sees users train a model using one kind of loss, but evaluate it using a very different loss – seems a bad idea in general.
- It's also worth noting that the loss can heavily effect how easy it is to train a model. For example, in classification, using the logistic regression model $p(c = 1|x) = \sigma(\theta^\mathsf{T}\mathbf{x})$ with a log-loss

$$U(\theta) = \frac{1}{N}\sum_n \sum_{c^{true}} p(c^{true}|n)\log\sigma((2c^{true} - 1)\theta^\mathsf{T}\mathbf{x}^n)$$

  gives a convex optimisation problem for θ. Using a squared loss gives a non-convex problem.

Empirical Risk Approach

Using the empirical distribution

- Given a dataset of train data $\mathcal{D} = \{(x^n, c^n), n = 1, \dots, N\}$, a direct approach to not knowing the correct model $p^{true}(c, x)$ is to replace it with the *empirical distribution*

$$p(c, x|\mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} \delta(c, c^n) \delta(x, x^n)$$

- Using this gives the empirical expected utility

$$\langle U(c, c(x)) \rangle_{p(c,x|\mathcal{D})} = \frac{1}{N} \sum_{n} U(c^n, c(x^n))$$

or conversely the *empirical risk*

$$R = \frac{1}{N} \sum_{n} L(c^n, c(x^n))$$

- Assuming the loss is minimal when the correct class is predicted, the optimal decision $c(x)$ for any input in the train set is given by $c(x^n) = c^n$.
- However, for any new x^* not contained in $\mathcal{D}$ then $c(x^*)$ is undefined.

Prediction beyond the train set

- To define the class of a novel input we use a parametric function $c(x|\theta)$. For example for a two class problem $\mathrm{dom}(c) = \{1, 2\}$, a linear decision function is given by

$$c(\mathbf{x}|\theta) = \begin{cases} 1 \text{ if } \boldsymbol{\theta}^\mathsf{T}\mathbf{x} + \theta_0 \geq 0 \\ 2 \text{ if } \boldsymbol{\theta}^\mathsf{T}\mathbf{x} + \theta_0 < 0 \end{cases}$$

If the vector input $\mathbf{x}$ is on the positive side of a hyperplane defined by the vector $\boldsymbol{\theta}$ and bias θ_0, we assign it to class 1, otherwise to class 2. The empirical risk then becomes a function of the parameters $\theta = \{\boldsymbol{\theta}, \theta_0\}$,

$$R(\theta|\mathcal{D}) = \frac{1}{N}\sum_n L(c^n, c(x^n|\theta))$$

The optimal parameters θ are given by minimising the empirical risk with respect to θ,

$$\theta_{opt} = \underset{\theta}{\mathrm{argmin}}\ R(\theta|\mathcal{D})$$

The decision for a new datapoint x^* is then given by $c(x^*|\theta_{opt})$.

Example of the Empirical Risk approach

Let's consider a two class problem in which we have utility

$$U(c^{true} = 1, c^{pred} = 1) = 1, U(c^{true} = 2, c^{pred} = 2) = 1$$

$$U(c^{true} = 1, c^{pred} = 2) = 0, U(c^{true} = 2, c^{pred} = 1) = -20$$

and consider a classifier:

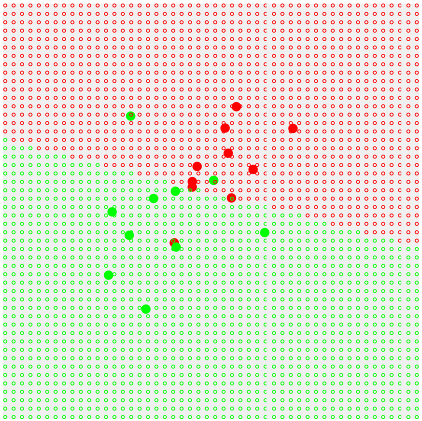
$$c(\mathbf{x}|\theta) = \begin{cases} 1 \text{ if } \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} + \theta_0 \geq 0 \\ 2 \text{ if } \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} + \theta_0 < 0 \end{cases}$$

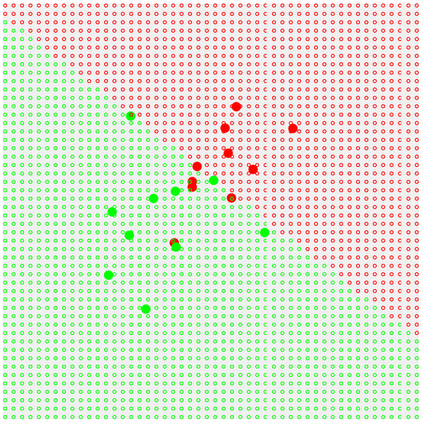
Then the unpenalised utility $U(\theta)$ is

$$\sum_{n|c^n=1} \mathbb{I}\left[\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^n + \theta_0 \geq 0\right] + \sum_{n|c^n=2} \mathbb{I}\left[\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^n + \theta_0 < 0\right] - 20 \sum_{n|c^n=2} \mathbb{I}\left[\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^n + \theta_0 \geq 0\right]$$

Finding θ that maximises this utility is difficult since this is a non-convex objective. Once learned we make a prediction $c(x^*|\theta)$ for a novel x^*. The decision boundary is linear in this particular case.

Empirical Risk Examples


(a) Using an identity utility matrix $U = I$


(b) Using $U(1,1) = U(2,2) = 1, U(2,1) = -20, U(1,2) = 0$

For each utility U we need to retrain the model. Note how changing the utility changes the decision boundary. The larger circle are the training points, and the smaller circles the test points.

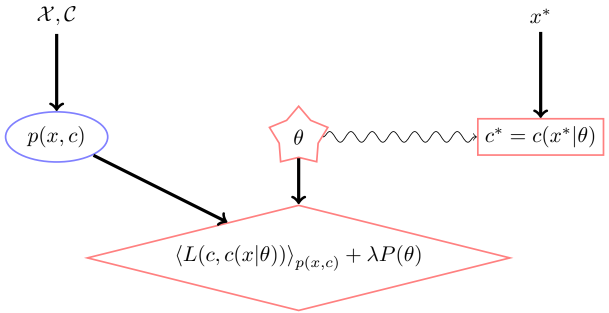
Overfitting

- In this *empirical risk minimisation* approach, as we make the decision function $c(x|\theta)$ more flexible, the empirical risk goes down.
- However, if we make $c(x|\theta)$ too flexible we will have little confidence that $c(x|\theta)$ will perform well on a novel input x^*. Since we only constrain the decision function on the known training points, a flexible $c(x|\theta)$ may change rapidly as we move away from the train data, leading to poor generalisation.
- To constrain the complexity of $c(x|\theta)$ we may minimise the penalised empirical risk

  $$R'(\theta|\mathcal{D}) = R(\theta|\mathcal{D}) + \lambda P(\theta)$$

  where $P(\theta)$ is a function that penalises complex functions $c(x|\theta)$. The *regularisation constant,* λ, determines the strength of this penalty and is typically set by validation.

Empirical Risk



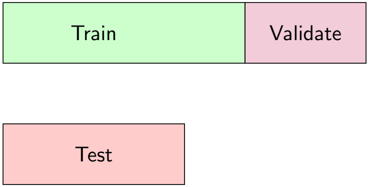
$\mathcal{X}, \mathcal{C}$
x^*
$p(x, c)$
θ
$c^* = c(x^*|\theta)$
$\langle L(c, c(x|\theta)) \rangle_{p(x,c)} + \lambda P(\theta)$

Empirical risk approach. Given the dataset $\mathcal{X}, \mathcal{C}$, a model of the data $p(x, c)$ is made, usually using the empirical distribution. For a classifier $c(x|\theta)$, the parameter θ is learned by minimising the penalised empirical risk with respect to θ. The penalty parameter λ is set by validation. A novel input x^* is then assigned to class $c(x^*|\theta)$, given this optimal θ.

Heuristic justification for squared Penalty

- For the linear decision function above, it is reasonable to penalise wildly changing classifications in the sense that if we change the input $\mathbf{x}$ by only a small amount we expect (on average) minimal change in the class label.
- The squared difference in $\boldsymbol{\theta}^\mathsf{T}\mathbf{x} + \theta_0$ for two inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ is $\left(\boldsymbol{\theta}^\mathsf{T}\Delta\mathbf{x}\right)^2$ where $\Delta\mathbf{x} \equiv \mathbf{x}_2 - \mathbf{x}_1$.
- By constraining the length of $\boldsymbol{\theta}$ to be small we limit the ability of the classifier to change class for only a small change in the input space.
- Assuming the distance between two datapoints is distributed according to an isotropic multivariate Gaussian with zero mean and covariance $\sigma^2\mathbf{I}$, the average squared change is $\left\langle \left(\boldsymbol{\theta}^\mathsf{T}\Delta\mathbf{x}\right)^2 \right\rangle = \sigma^2\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\theta}$, motivating the choice of the Euclidean squared length of the parameter $\boldsymbol{\theta}$ as the penalty term, $P(\theta) = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\theta}$.

Validation

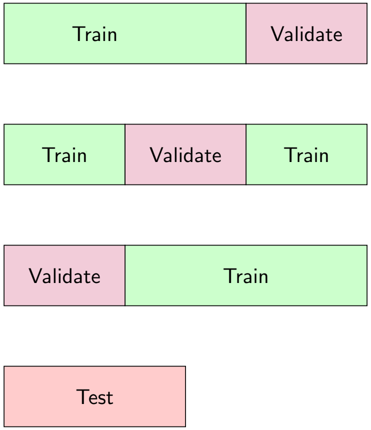


Models can be trained using the train data based on different regularisation parameters. The optimal regularisation parameter is determined by the empirical performance on the validation data. An independent measure of the generalisation performance is obtained by using a separate test set.

Validation

- In penalised empirical risk minimisation we need to set the regularisation constant λ. This can be achieved by evaluating the performance of the learned classifier $c(x|\theta)$ on validation data $\mathcal{D}_{validate}$ for several different λ values, and choosing the λ which gave rise to the classifier with the best performance.
- It's important that the validation data is not the data on which the model was trained since we know that the optimal setting for λ in that case is zero, and again we will have little confidence in the generalisation ability.
- Given a dataset $\mathcal{D}$ we split this into disjoint parts, $\mathcal{D}_{train}, \mathcal{D}_{validate}$, where the size of the validation set is usually chosen to be smaller than the train set. For each parameter λ_a one then finds the minimal empirical risk parameter θ_a. The optimal λ is chosen as that which gives rise to the model with the minimal validation risk. Using the optimal regularisation parameter λ, many practitioners retrain θ on the basis of the whole dataset $\mathcal{D}$.

Cross Validation

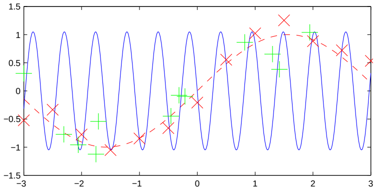


In cross-validation the dataset is split into several train-validation sets. Depicted is 3-fold cross-validation. For a range of regularisation parameters, the optimal regularisation parameter is found based on the empirical validation performance averaged across the different splits.

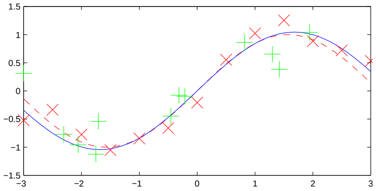
Cross validation

- In cross-validation the dataset is partitioned into training and validation sets multiple times with validation results obtained for each partition.
- Each partition produces a different training $\mathcal{D}^i_{train}$ and validation $\mathcal{D}^i_{validate}$ set, along with an optimal penalised empirical risk parameter θ^i_a and associated (unregularised) validation performance $R(\theta^i_a|\mathcal{D}^i_{validate})$.
- The performance of regularisation parameter λ_a is taken as the average of the validation performances over i. The best regularisation parameter is then given as that with the minimal average validation error.
- In K-fold cross-validation the data $\mathcal{D}$ is split into K equal sized disjoint parts $\mathcal{D}_1, \dots, \mathcal{D}_K$. Then $\mathcal{D}^i_{validate} = \mathcal{D}_i$ and $\mathcal{D}^i_{train} = \mathcal{D} \backslash \mathcal{D}^i_{validate}$. This gives a total of K different training-validation sets over which performance is averaged. In practice 10-fold cross-validation is popular, as is leave-one-out cross-validation in which the validation sets consist of only a single example.

Validation example



(c)



(d)

The true function which generated the noisy data is the dashed line; the function learned from the data is given by the solid line. **(a)**: The unregularised fit ($\lambda = 0$) to training given by $\times$. Whilst the training data is well fitted, the error on the validation examples, denoted by $+$, is high. **(b)**: The regularised fit ($\lambda = 0.5$). Whilst the train error is high, the validation error is low.

Empirical risk approach

Benefits

- In the limit of a large amount of training data the empirical distribution tends to the correct distribution.
- The discriminant function is chosen on the basis of minimal risk, which is the quantity we are ultimately interested in.
- The procedure is conceptually straightforward.

Drawbacks

- It seems extreme to assume that the data follows the empirical distribution, particularly for small amounts of train data. More reasonable assumptions for $p(x)$ would take into account likely x that could arise, not just those in the train data.
- If the loss function changes, the discriminant function needs to be retrained.
- Some problems require an estimate of the confidence of the prediction. Whilst there may be heuristic ways to evaluating confidence in the prediction, this is not inherent in the framework.

Bayesian Decision Approach

Bayesian decision approach

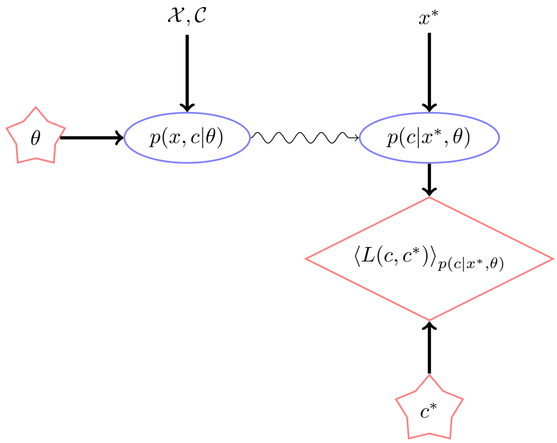
- An alternative to using the empirical distribution is to first fit a model $p(c, x|\theta)$ to the train data $\mathcal{D}$.
- How the model $p(c, x|\theta)$ is trained is a separate issue. This can be done for example using maximum likelihood. For example

$$\theta = \arg\max_{\theta} \sum_{n=1}^{N} \log p(c^n, x^n|\theta)$$

- Note that issues of overfitting also arise here and can be addressed using regularisation and validation. For example, adjust regularisation parameters to ensure that a validation likelihood is high.
- Given this model, the decision function $c(x)$ is automatically determined from the maximal expected utility (or minimal risk) with respect to this model, in which the unknown $p(c^{true}|x)$ is replaced with $p(c|x, \theta)$:

$$U(c(x^*)) = \sum_{c} U(c, c(x^*))p(c|x^*, \theta)$$

and $c(x^*) = \arg\max_{c(x^*)} U(c(x^*))$.



$\mathcal{X}, \mathcal{C}$
x^*
θ
$p(x, c|\theta)$
$p(c|x^*, \theta)$
$\langle L(c, c^*) \rangle_{p(c|x^*, \theta)}$
c^*

Bayesian decision approach. A model $p(x, c|\theta)$ is fitted to the data. After learning the optimal model parameters θ, we compute $p(c|x, \theta)$. For a novel x^*, the distribution of the assumed 'truth' is $p(c|x^*, \theta)$. The prediction (decision) is then given by that c^* which minimises the expected risk $\langle L(c, c^*) \rangle_{p(c|x^*, \theta)}$.

Discriminative versus Generative modelling

There are two main approaches to fitting $p(c, x|\theta)$ to data $\mathcal{D}$. These boil down to the two ways to parameterise a joint distribution.

Discriminative approach

In this case we write:

$$p(c, x|\theta) = p(c|x, \theta_{c|x})p(x|\theta_x)$$

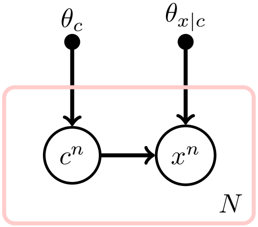
so that we we explicitly parameterise how to form the class distribution.

Generative approach

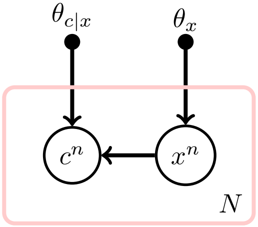
In this case we make a model of x given the class:

$$p(c, x|\theta) = p(x|c, \theta_{x|c})p(c|\theta_c)$$

Generative versus Discriminative modelling



(e)



(f)

Two generic strategies for probabilistic classification. **(a)**: Class dependent generative model of x. After learning parameters, classification is obtained by making x evidential and inferring $p(c|x)$. **(b)**: A discriminative classification method $p(c|x)$.

Example of the Generative Modelling Approach

We first fit a model to data, for example:

$$p(x|c) = \mathcal{N}(x|\mu_c, \Sigma_c)$$

using say maximum likelihood. This is easy – just use the empirical mean and covariance. We set $p(c = 1)$ to the fraction of training datapoints in class 1, and similarly for $p(c = 2)$. For any subsequent decision for class of input x^* we calculate:

$$p(c = 1|x^*) = \frac{p(x^*|c = 1)p(c = 1)}{p(x^*|c = 1)p(c = 1) + p(x^*|c = 2)p(c = 2)}$$

$$p(c = 2|x^*) = \frac{p(x^*|c = 2)p(c = 2)}{p(x^*|c = 1)p(c = 1) + p(x^*|c = 2)p(c = 2)}$$

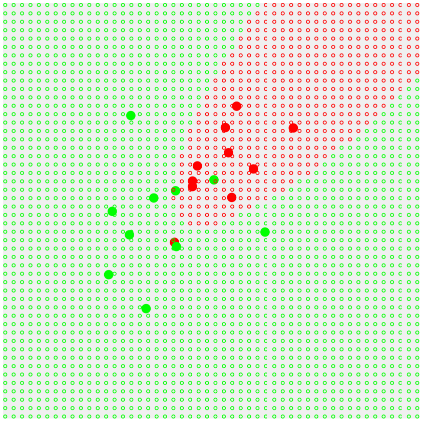
and

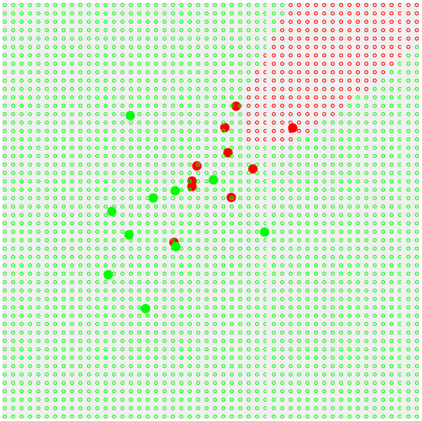
$$U(c^* = 1) = U(1, 1)p(c = 1|x^*) + U(2, 1)p(c = 2|x^*)$$

$$U(c^* = 2) = U(1, 2)p(c = 1|x^*) + U(2, 2)p(c = 2|x^*)$$

Then we set $c^* = 1$ if $U(c^* = 1) > U(c^* = 2)$. In this case the resulting decision boundary is non-linear.

Generative Modelling Examples


(g) Using an identity utility matrix $U = I$


(h) Using $U(1,1) = U(2,2) = 1, U(2,1) = -20, U(1,2) = 0$

The model is fitted independently of U. Note how changing the utility changes the decision boundary. The larger circle are the training points, and the smaller circles the test points.

Generative approach

Advantages Prior information about the structure of the data is often most naturally specified through a generative model $p(x|c)$. For example, for male faces, we would expect to see heavier eyebrows, a squarer jaw, *etc.*

Disadvantages The generative approach does not directly target the classification model $p(c|x)$ since the goal of generative training is rather to model $p(x|c)$. If the data x is complex, finding a suitable generative data model $p(x|c)$ is a difficult task. Furthermore, since each generative model is separately trained for each class, there is no competition amongst the models to explain the x data. On the other hand it might be that making a model of $p(c|x)$ is simpler, particularly if the decision boundary between the classes has a simple form, even if the data distribution of each class is complex.

Example of the Discriminative Modelling Approach

Let's first fit a model to data:

$$p(c = 1|x) = \sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right)$$

using say maximum likelihood. This is easy since the log-likelihood is convex.
For any subsequent decision for class of input x^* we calculate:

$$U(c^* = 1) = U(1,1)\sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right) + U(2,1)\left(1 - \sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right)\right)$$

$$U(c^* = 2) = U(1,2)\sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right) + U(2,2)\left(1 - \sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right)\right)$$

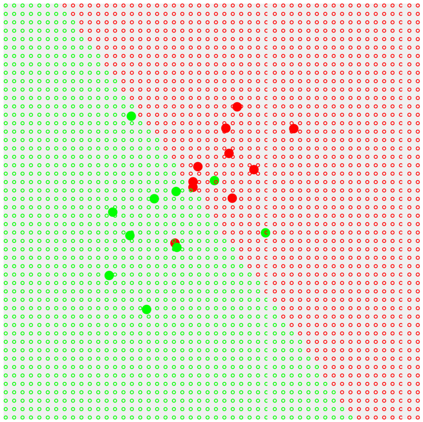
Then we set $c^* = 1$ if $U(c^* = 1) > U(c^* = 2)$.
For our example the specific utilities give:

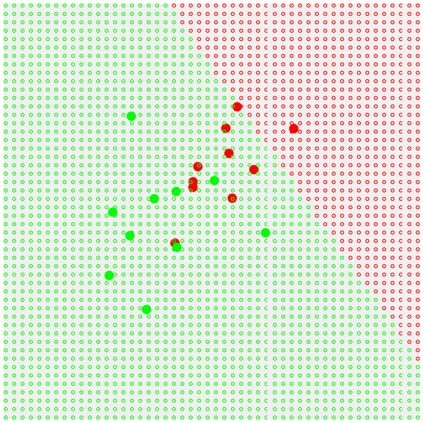
$$U(c^* = 1) = \sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right) - \left(1 - \sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right)\right) = 2\sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right) - 1$$

$$U(c^* = 2) = 1 - \sigma\left(\theta^\mathsf{T}\mathbf{x} + \theta_0\right)$$

For this model the resulting decision boundary is linear.

Discriminative Modelling Examples


(i) Using an identity utility matrix $U = I$


(j) Using $U(1,1) = U(2,2) = 1, U(2,1) = -20, U(1,2) = 0$

The model is fitted independently of U. Note how changing the utility changes the decision boundary. The larger circle are the training points, and the smaller circles the test points.

Discriminative approach

Advantages	The discriminative approach directly addresses finding an accurate classifier $p(c|x)$ based on modelling the decision boundary, as opposed to the class conditional data distribution in the generative approach. Whilst the data from each class may be distributed in a complex way, it could be that the decision boundary between them is relatively easy to model.
Disadvantages	Discriminative approaches are usually trained as ‘black-box’ classifiers, with little prior knowledge built used to describe how data for a given class is distributed. Domain knowledge is often more easily expressed using the generative framework.

Bayesian decision approach

Benefits

- This is a conceptually ‘clean’ approach, in which one tries ones best to model the environment (using either a generative or discriminative approach), independent of the subsequent decision process.
- In this case learning the environment is separated from the effect this will have on the expected utility.
- The decision c^* for a novel input x^* can be a highly complex function of x^* due to the maximisation operation.
- If $p(x, c|\theta)$ is the ‘true’ model of the data, this approach is optimal.

Drawbacks

- If the environment model $p(c, x|\theta)$ is poor, the prediction c^* could be highly inaccurate since modelling the environment is divorced from prediction.
- To avoid divorcing the learning of the model $p(c, x|\theta)$ from its effect on decisions, in practice one often includes regularisation terms in the environment model $p(c, x|\theta)$ which are set by validation based on an empirical loss.