V. Classification

1) Logistic regression

1.1) The cross-entropy loss function

1.2) SoftMax regression

2) Applications

2.1) Classifying data points in 2d

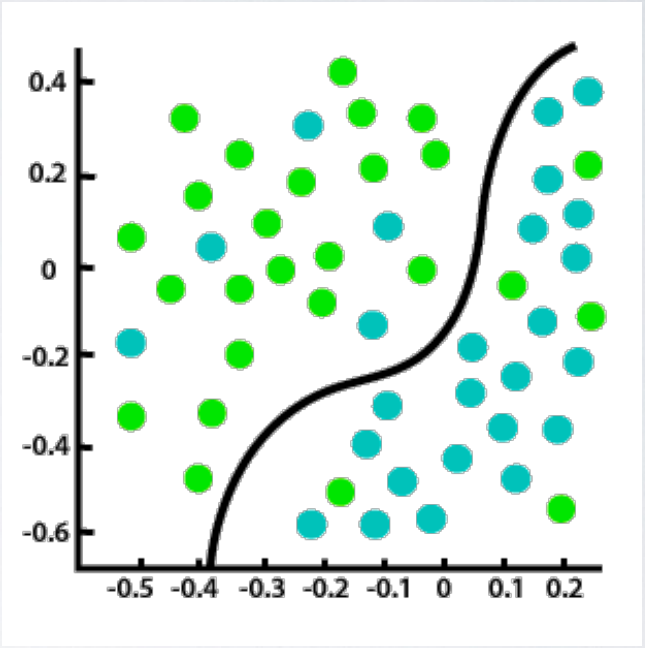
2.2) Classifying signal/background with collider events

1)

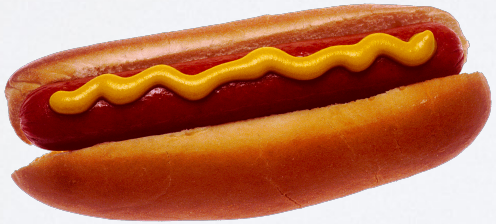
Logistic regression

- Discrete outcomes from the input data

kitty or croissant



hot dog

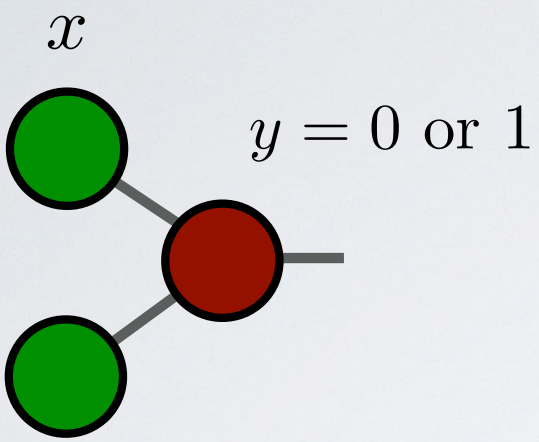


not hot dog

start with simple case:
output is 0 or 1

1.1)

Cross-entropy loss function



recall the Perceptron which acted as a linear classifier

$$\sigma(\vec{x}.\vec{w} + b) = \sigma(\vec{X}.\vec{W})$$

theta/sign function

$$\sigma(z) = \Theta(z)$$

"hard" classifier

decide the answer is 0 or 1

sigmoid/logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

"soft" classifier

probability that the answer is 0 or 1

1.1)

Cross-entropy loss function

Probability that a data point X=(1,x) belongs to a category y=0,1 for a model of weights W

$$P(y=1|x,W) = \frac{1}{1+e^{-X.W}}$$

$$P(y=0|x,W) = 1 - P(y=1|x,W)$$

Boltzman distribution of a two state system

$$p_i = Q^{-1} e^{-E_i/(kT)} \qquad p_1 = \frac{1}{1+e^{-(E_0-E_1)/(kT)}}$$

$$Q = e^{-E_0/(kT)} + e^{-E_1/(kT)} \qquad p_0 = 1 - p_1$$

1.1)

Cross-entropy loss function

Given a model, $\vec{W}$,the **likelihood** that it matches a data set $\mathcal{D} = \{\vec{x}_i, y_i\}$ is

$$P(\{\vec{x}_i, y_i\}|\vec{W}) = \prod_{i=1}^{n} [\sigma(\vec{X}_i.\vec{W})]^{y_i} [1 - \sigma(\vec{X}_i.\vec{W})]^{1-y_i}$$

where the loss function, $\mathcal{L}$, is -log (likelihood)

$$\mathcal{L}(\vec{W}) = -\sum_{i=1}^{n} y_i \log(\sigma(\vec{X}_i.\vec{W})) + (1 - y_i) \log(1 - \sigma(\vec{X}_i.\vec{W}))$$

This is called the cross-entropy

1.1)

Cross-entropy loss function

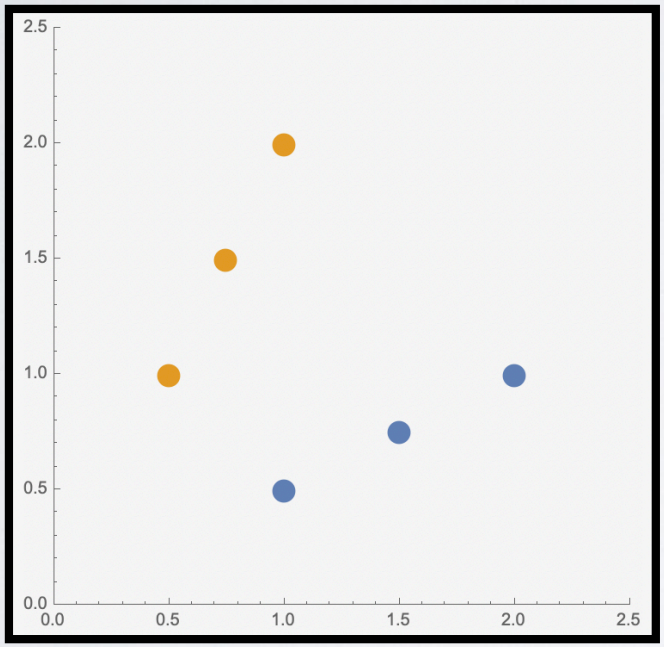
Minimising the cross entropy results in a set of optimal weights for our model

$$\frac{\partial}{\partial \vec{W}} \mathcal{L}(\vec{W}) = \sum_i \vec{X}_i \left(\sigma(\vec{X}_i . \vec{W}) - y_i \right) = \sum_i \vec{X}_i \left(\hat{y}_i - y_i \right)$$

Training the model can be achieved through gradient descent methods

1.1)

Explicit example



from the data:

```
Data = {
{{2,1},0},{{1,1/2},0},{{3/2,3/4},0},
{{1,2},1},{{1/2,1},1},{{3/4,3/2},1}
};
```

compute cross entropy loss function at values for the model parameters:

$$\mathcal{L}(\{-1.5, 1, -1\})$$
$$\mathcal{L}(\{0, 1, -1\})$$
$$\mathcal{L}(\{-1.5, 1, 1\})$$

implement the gradient of the loss function and find the minimum via a few iterations of gradient descent

$$\vec{\nabla}\mathcal{L} = 0$$

1.2)

SoftMax regression

The generalistion from a binary output (0,1) to a discrete list of values (0,1,2…) (referred to as ***labels***) requires generalising ***Logistic Regression*** to ***SoftMax regression***

The generalisation is quite simple. Recall the Boltzman distribution of a multi-state system

$$p_I = Q^{-1} e^{-E_I/(kT)} \qquad I = 1, ..., M$$

$$Q = \sum_{I=1}^{M} e^{-E_I/(kT)} \qquad \text{therefore} \qquad \sum_{I=1}^{M} p_I = 1$$

1.2)

SoftMax regression

probability of being in a state I

$$P(y = I|\vec{x}, \{\vec{W}_J\}) = \frac{1}{1 + \sum_{J \neq I} e^{-\vec{X}.(\vec{W}_J - \vec{W}_I)}}$$

which we may think of as a vector of probabilties

$$\vec{P}(\vec{x}, \{\vec{W}_J\}) = \begin{pmatrix} P(y = 1|\vec{x}, \{\vec{W}_J\}) \\ P(y = 2|\vec{x}, \{\vec{W}_J\}) \\ \vdots \\ P(y = M|\vec{x}, \{\vec{W}_J\}) \end{pmatrix}$$

1.2)

SoftMax regression

the data is then given as $\{\vec{x}, \vec{y}\}$

and so the loss function is

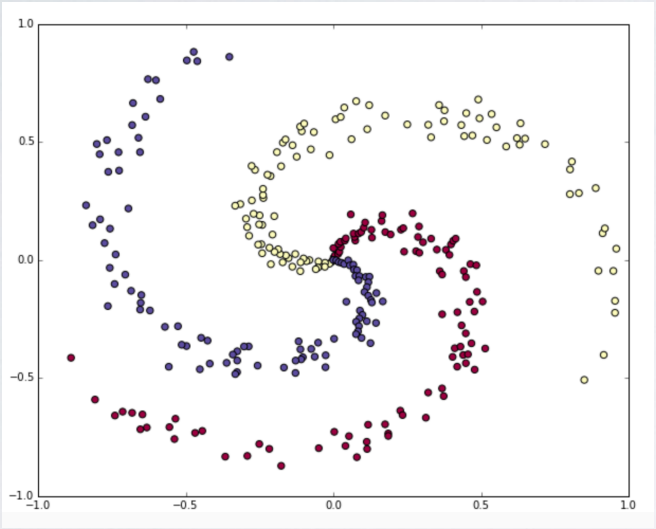
$$\mathcal{L}(\{\vec{W}_J\}) = -\sum_{i=1}^{n} \vec{y}_i . \log(\vec{P}(\vec{x}_i, \{\vec{W}_J\})) + (\vec{1} - \vec{y}_i) . \log(\vec{1} - \vec{P}(\vec{x}_i, \{\vec{W}_J\}))$$

vector of 1s

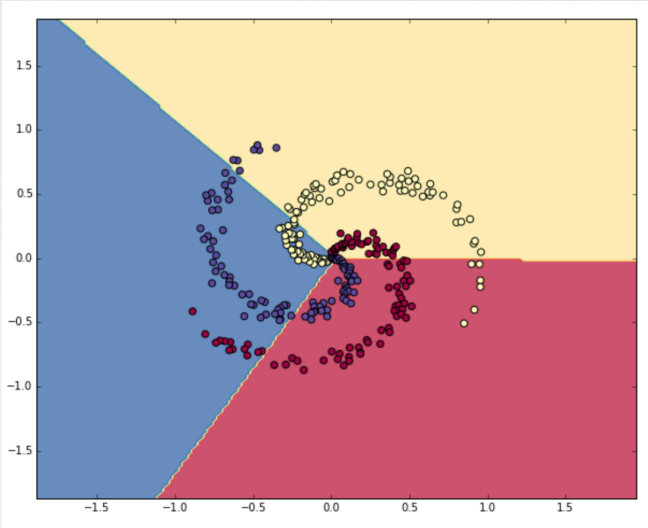
2.1)

Classifying points in 2d

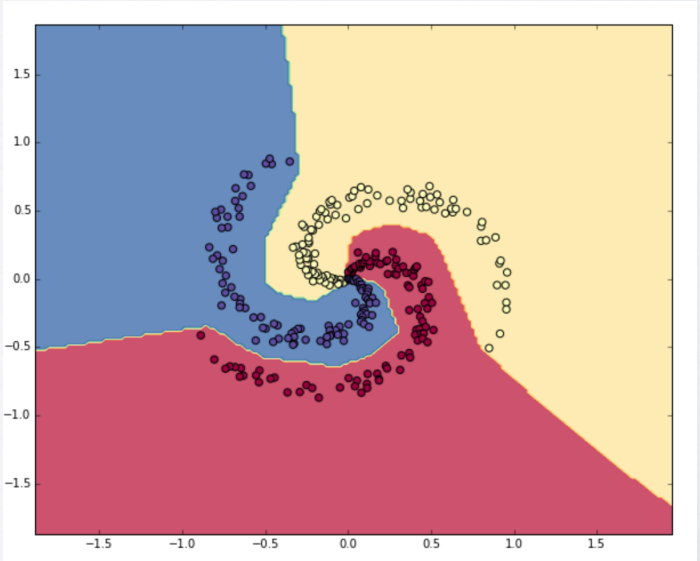
https://cs231n.github.io/neural-networks-case-study/



linear classifier



neural network classifier



follow the tutorial to see how to implement it

2.2)

Measuring performance: the ROC curve

ROC = Receiver Operating Characteristic

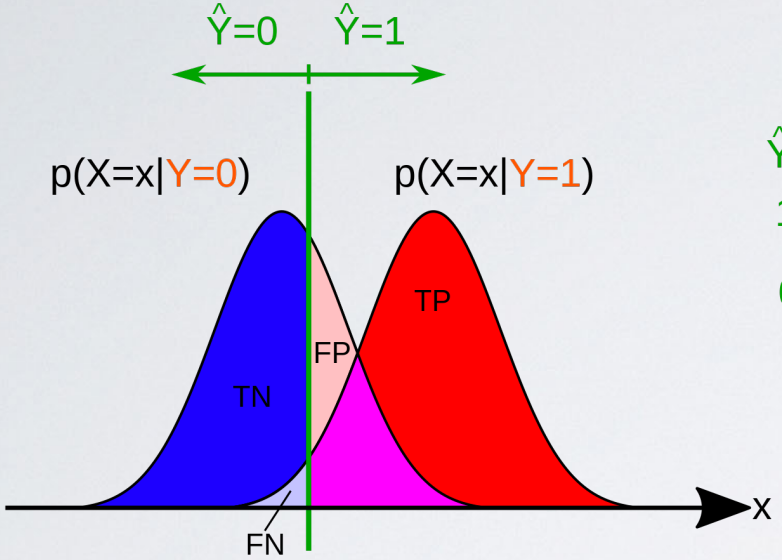
Once we've trained a classifier how do we intepret the inferred values?

- take a test: pass sample values to the trained model
- outcome is a probability between 0, 1 of the likelihood the test is positive. Let's say >1/2 means True and <1/2 is False

		actual result: True	actual result: False
test result	positive	True Positive	False Positive
test result	negative	False Negative	True Negative

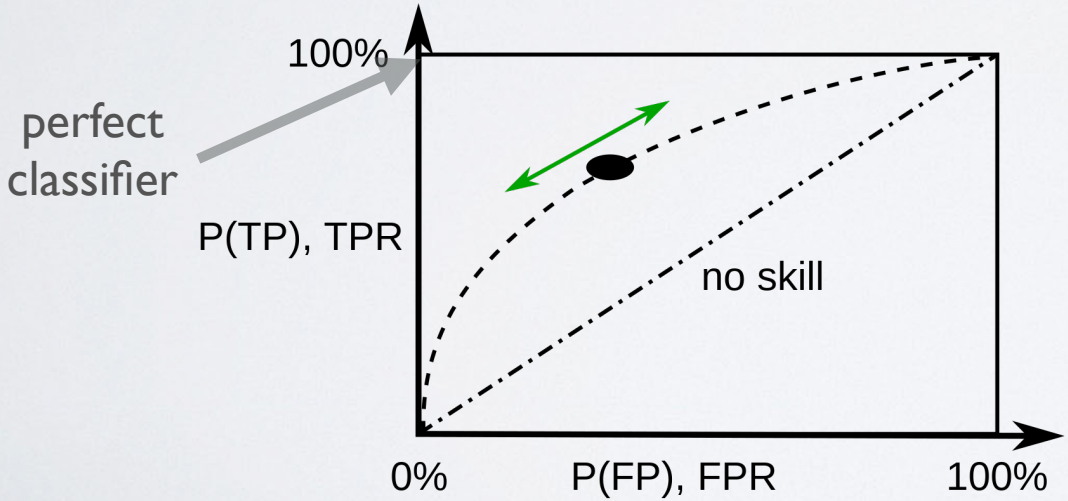
2.2)

Measuring performance: the ROC curve



Ŷ \ Y	1	0
1	TP	FP
0	FN	TN

Y is the actual result



P(X=x|Y=1) is the result from the model

P(X=x|Y=0) = 1-P(X=x|Y=0)

Ŷ = 1 if P(X=x|Y=1) > threshold,
= 0 if P(X=x|Y=1) < threshold

(…figure from Wlkipedia! sorry…)

2.2)

Discriminating signal from background

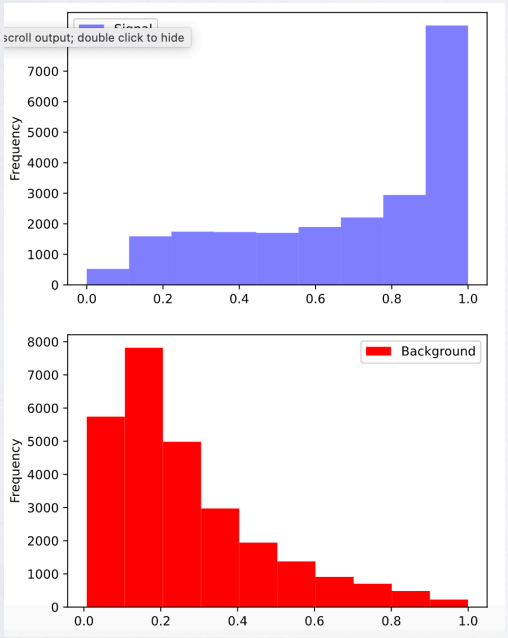
tutorial from: https://github.com/drckf/mlreview_notebooks/tree/master/jupyter_notebooks/notebooks

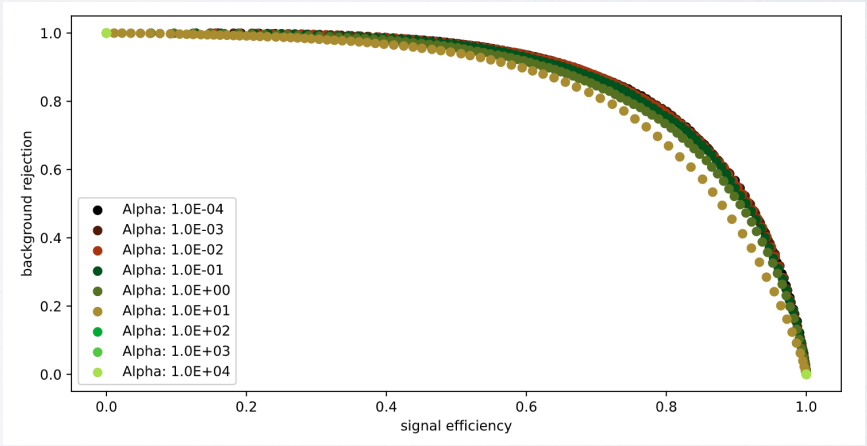
Super-symmetry has been a popular model for physics Beyond the Standard Model. LHC data has almost completely ruled out an realistic SUSY scenario

also in our repository with dataset:
`LogisticRegression_SUSY.ipynb`

- use generic pre-generated 'SUSY' dataset

MC generated samples based on theory models of Standard Model (SM = background) and supersymmetry (SUSY = signal)



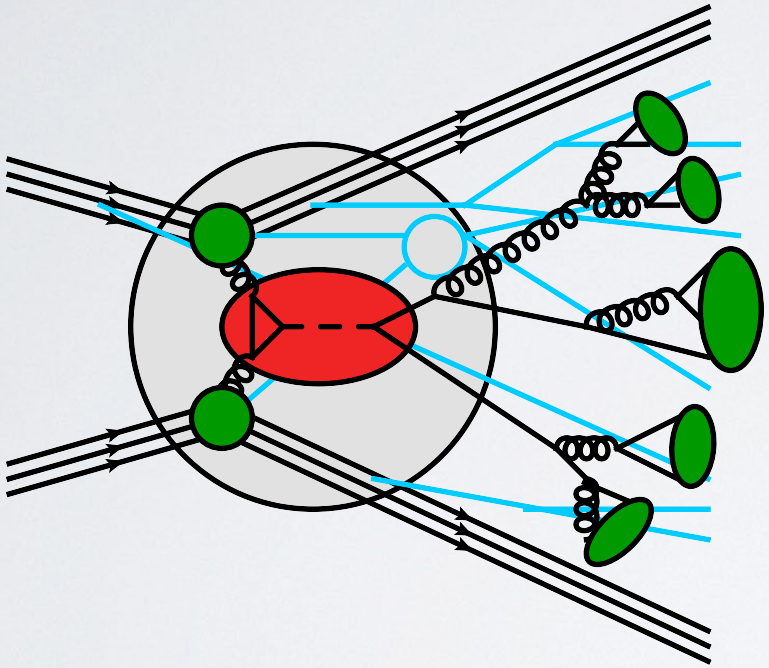


NB: ROC as before but with x → 1-x

2.2)

Discriminating signal from background

overview of data and event generation



Monte Carlo integration for a given final state (including phase-space cuts) results in a set of **momenta** and **particle types** together with a weight (e.g. squared amplitude x PDFs)

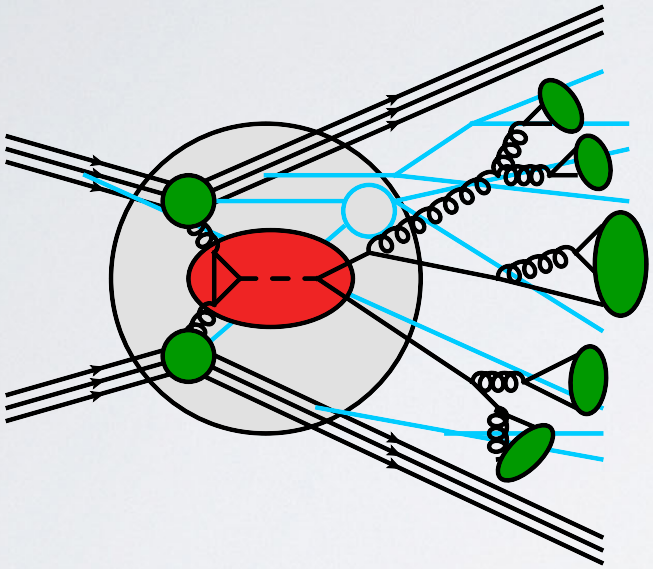
from this data we can compute:

- observable kinematic quantities (transverse momentum, rapidity, azimuthal angle,…)
- differential distributions - histogram weights according to value of a given observable

2.2)

Discriminating signal from background

overview of data and event generation



y - rapidity

p_T - transverse momentum

m_T - transverse mass

Φ - azimuthal angle

$$(E, p_x, p_y, p_z) = (m_T \cosh(y), p_T \cos(\phi), p_T \sin(\phi), m_T \sinh(y))$$
$$\overset{m=0}{=} p_T(\cosh(y), \cos(\phi), \sin(\phi), \sinh(y))$$

$$E\frac{d^3\sigma}{d\vec{p}} = \frac{d^3\sigma}{d\phi\, dy\, dp_T}$$

2.2)

Discriminating signal from background

overview of data and event generation

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right)$$

rapidity

angle from z-axis

$$y = \frac{1}{2} \ln \frac{\cos^2(\theta/2) + m^2/4p^2 + \ldots}{\sin^2(\theta/2) + m^2/4p^2 + \ldots}$$

$$\overset{m^2 \ll p^2}{\approx} -\ln \tan(\theta/2) \equiv \eta$$

pseudo rapidity

	signal	lepton 1 pT	lepton 1 eta	lepton 1 phi	lepton 2 pT	lepton 2 eta	lepton 2 phi	missing energy magnitude	missing energy phi	MET_rel	axial MET	M_R
0	0.0	0.972861	0.653855	1.176225	1.157156	-1.739873	-0.874309	0.567765	-0.175000	0.810061	-0.252552	1.921887
1	1.0	1.667973	0.064191	-1.225171	0.506102	-0.338939	1.672543	3.475464	-1.219136	0.012955	3.775174	1.045977
2	1.0	0.444840	-0.134298	-0.709972	0.451719	-1.613871	-0.768661	1.219918	0.504026	1.831248	-0.431385	0.526283
3	1.0	0.381256	-0.976145	0.693152	0.448959	0.891753	-0.677328	2.033060	1.533041	3.046260	-1.005285	0.569386
4	1.0	1.309996	-0.690089	-0.676259	1.589283	-0.693326	0.622907	1.087562	-0.381742	0.589204	1.365479	1.179295

SM or 'other model' as defined by simulation

MET - missing transverse energy