# Logistic Regression

COMP 527
Danushka Bollegala

UNIVERSITY OF LIVERPOOL

# Binary Classification

- Given an instance **x** we must classify it to either positive (1) or negative (0) class
    - We can use {1,-1} instead of {1,0} but we will use the latter formulation as it simplifies the notation in subsequent derivations
- Binary classification can be seen as learning a function *f* such that *f*(x) returns either 1 or 0, indicating the predicted class

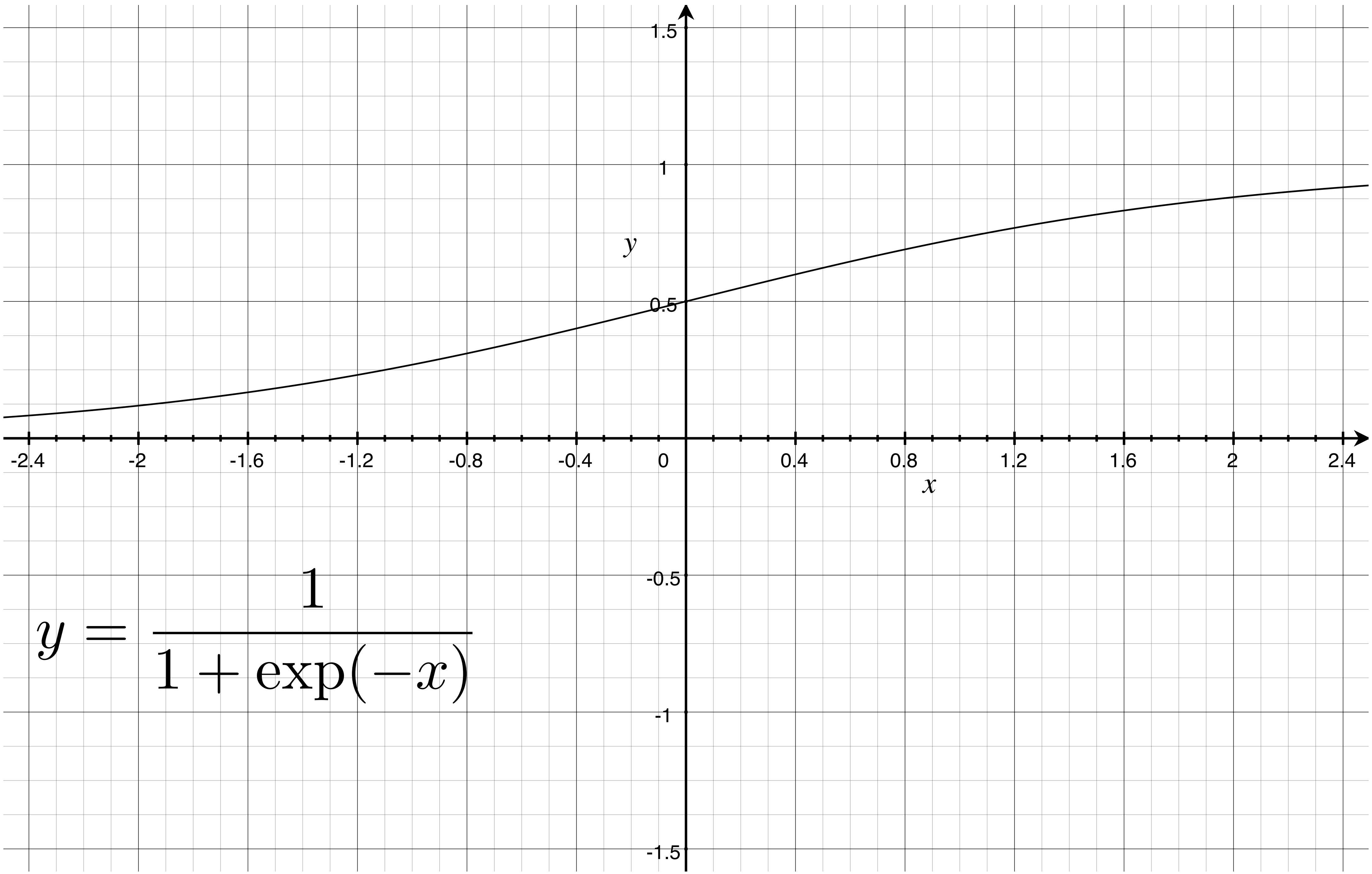
# Some terms in Machine Learning

- Training dataset with N instances
  - $\{(x_1,t_1), ..., (x_N,t_N)\}$
- Target label (class)
  - t: The class labels in the training dataset
  - Annotated by humans (supervised learning)
- Predicted label
  - Labels predicted by our model $f(x)$
- $P(A|B)$: conditional probability of observing an event A, given an event B
- $P(A)$: marginal probability of event A
  - We have *marginalised out* all the variables on which A depends upon (cf. margin of a probability table)
- Prior probability $P(B)$
- Posterior probability $P(B|A)$

# Logistic Regression

- is not a *regression* model
- is a *classification* model
- is the basis of many advanced machine learning methods
  - neural networks, deep learning, conditional random fields, ...
- Try to fit a logistic sigmoid function to predict the class labels

# Logistic Sigmoid Function




$$y = \frac{1}{1 + \exp(-x)}$$

# Why do we use logistic sigmoid?

- Reason 1:
  - We must squash the prediction score $\mathbf{w}^T\mathbf{x}$, which is in the range $(-\infty,+\infty)$ to the range [0,1] when performing binary classification
- Reason 2: (Bayes' Rule)
  - Posterior ∝ Conditional x Prior

$$
\begin{aligned}
P(t=1|x) &= \frac{P(x|t=1)P(t=1)}{P(x)} \\
&= \frac{P(x|t=1)P(t=1)}{P(t=1)P(x|t=1)+P(t=0)P(x|t=0)} \\
&= \frac{1}{1+\frac{1}{\frac{P(x|t=1)P(t=1)}{P(t=0)P(x|t=0)}}}
\end{aligned}
$$

$$
\exp(a) = \frac{P(x|t=1)P(t=1)}{P(t=0)P(x|t=0)}
$$

$$
P(t=1|x) = \frac{1}{1+\exp(-a)} = \sigma(a)
$$

# Likelihood

- We have a probabilistic model (logistic sigmoid function $\sigma(\mathbf{w}^T\mathbf{x})$) that tells us the probability of a particular training instance **x** being positive (t=1) or negative (t=0)
- We can use this model to predict the probability of the entire training dataset
  - *likelihood* of the training dataset
- However, this dataset is already *observed* (we have it with us)
- If we want to *explain* this training dataset, then our model must maximise the likelihood for this training dataset (more than any other labelling of the dataset)
- Maximum Likelihood Estimate/Principle (MLE)

# Maximum Likelihood Estimate

$$\begin{aligned} y_n &= \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) = \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x_n})} \\ \boldsymbol{t} &= (t_1, \ldots, t_n)^\top \\ p(\boldsymbol{t}|\boldsymbol{w}) &= \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{(1-t_n)} \end{aligned}$$

By taking the negative of the logarithm of the above product we define the cross-entropy error function

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

Home Work 1

By differentiating E(w) w.r.t. w we get ∇E(w) as follows:

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - t_n)x_n$$

Home Work 2

# HW1: Derivation of Cross Entropy Error Function

# HW2: Derivation of the gradient

# Updating the weight vector

- Generic update rule

$$\boldsymbol{w}^{(r+1)} = \boldsymbol{w}^{(r)} - \eta \nabla E(\boldsymbol{w})$$

- Update rule with cross-entropy error function

$$\boldsymbol{w}^{(r+1)} = \boldsymbol{w}^{(r)} - \eta (y_n - t_n) \boldsymbol{x}_n$$

# Logistic Regression Algorithm

- Given a set of training instances $\{(x_1,t_1), \ldots, (x_N,t_N)\}$, learning rate, $\eta$, and iterations T
- Initialise weight vector $\mathbf{w} = \mathbf{0}$
- For j in 1,...,T
  - For n in 1,...,N
    - if $\text{pred}(\mathbf{x}_i) \neq t_i$ #misclassification
      - $\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \eta(y_n - t_n)\mathbf{x}_n$
- Return the final weight vector **w**

# Prediction Function *pred*

- Given the weight vector **w**, returns the class label for an instance **x**
  - if $\mathbf{w}^T\mathbf{x} > 0$:
    - predicted label = +1 # positive class
  - else:
    - predicted label = 0 # negative class

# Online vs. Batch

- Online vs. Batch Logistic Regression
  - The algorithm we discussed in the previous slides is an *online algorithm* because it considers only one instance at a time and updates the weight vector
    - Referred to as the Stochastic Gradient Descent (SGD) update
  - In the batch version, we will compute the cross-entropy error over the *entire* training dataset and then update the weight vector
    - Popular optimisation algorithm for the batch learning of logistic regression is the Limited Memory BFGS (L-BFGS) algorithm
- Batch version is slow compared to the SGD version. But shows slightly improved accuracies in many cases
- SGD version can require multiple iterations over the dataset before it converges (if ever)
- SGD is a technique that is frequently used with large scale machine learning tasks (even when the objective function is non-convex)

# References

- Bishop (Pattern Recognition and Machine Learning) Section 4.3.2
- Software
  - scikit-learn (Python)
    - http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
  - Classias (C)
    - http://www.chokkan.org/software/classias/