Machine Learning in Healthcare

#L04-Linear models for classification

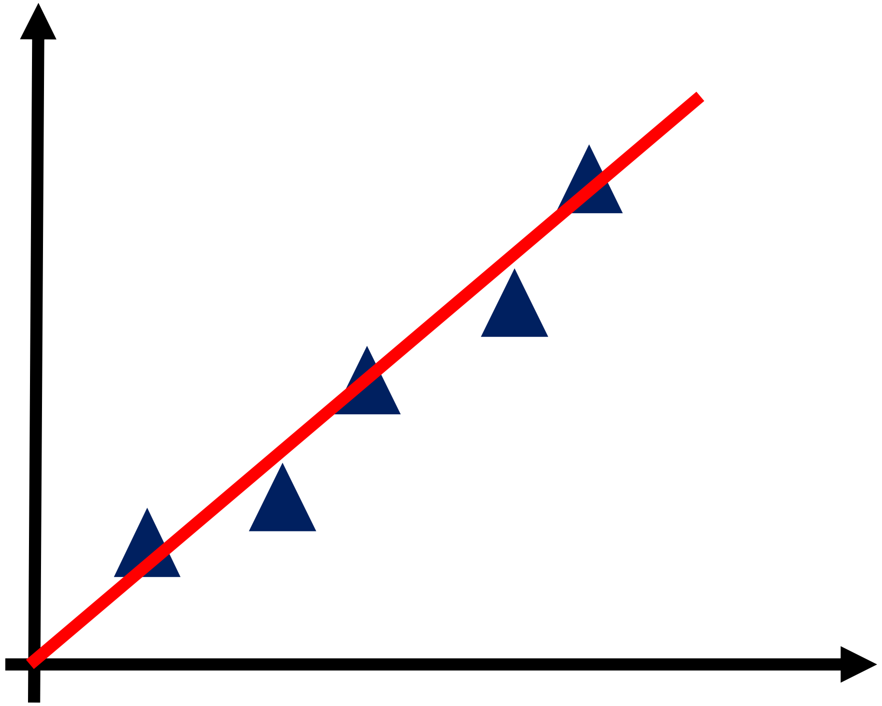
Technion-IIT, Haifa, Israel

Asst. Prof. Joachim Behar
Biomedical Engineering Faculty, Technion-IIT
Artificial intelligence in medicine laboratory (AIMLab.)
https://aim-lab.github.io/
Twitter: @lab_aim

Classification versus regression

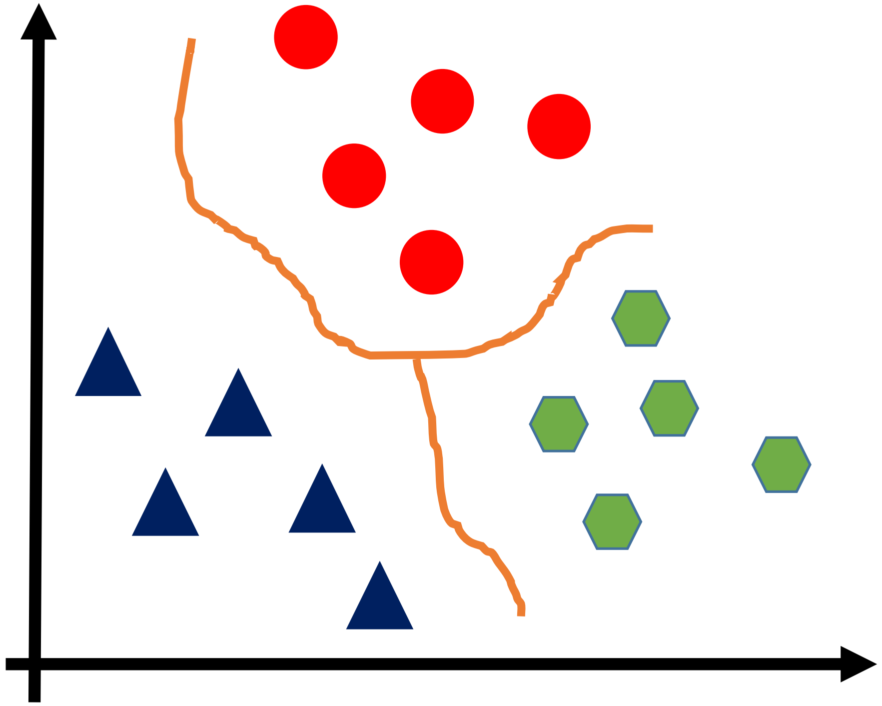
Regression versus classification

Regression



Estimate relationships among usually continuous variables.

Classification



Identify decision boundary between examples of different classes.

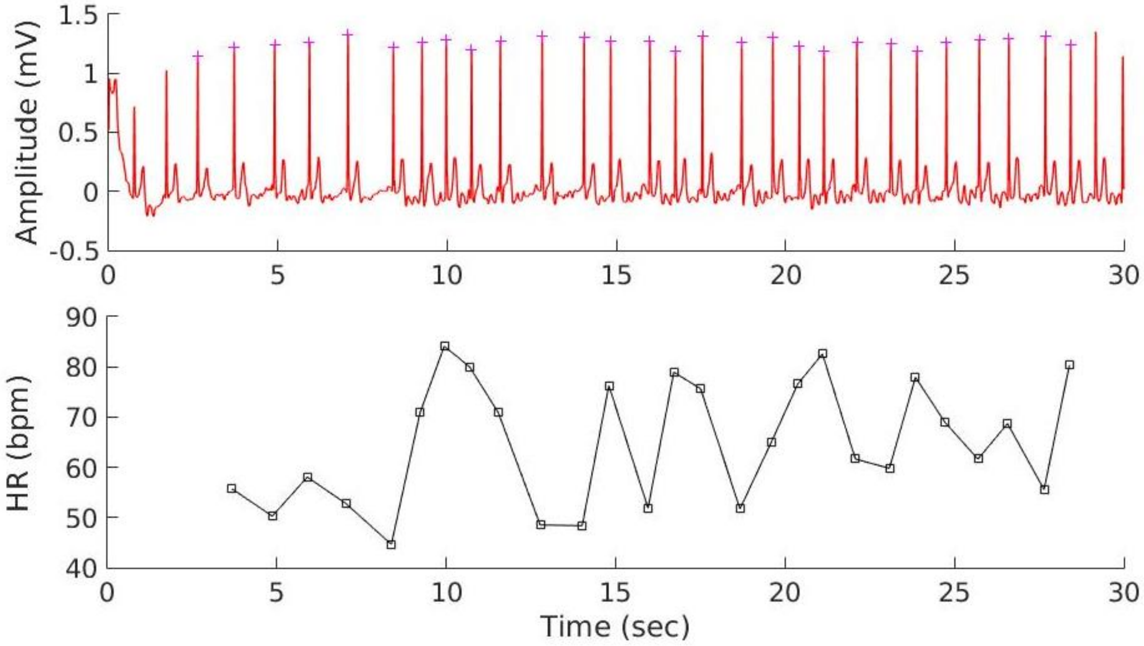
Classification

- Examples:
  - Tumor: Malignant/benign?
  - Rhythm: Atrial fibrillation/normal sinus?
- Let's consider a binary classification problem for now:

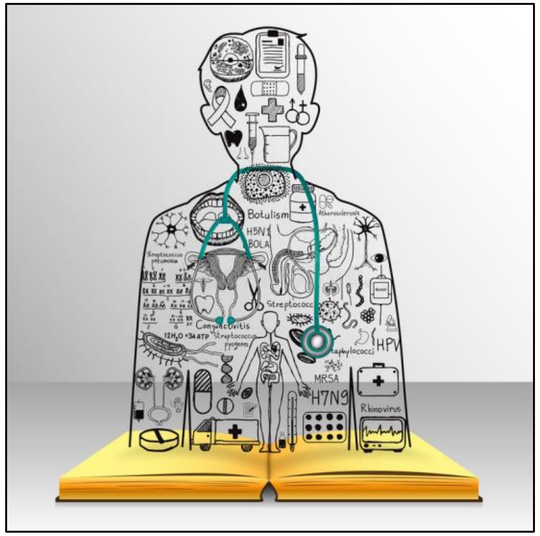
$$y \in \{0,1\}$$

0: negative class (non-AF),

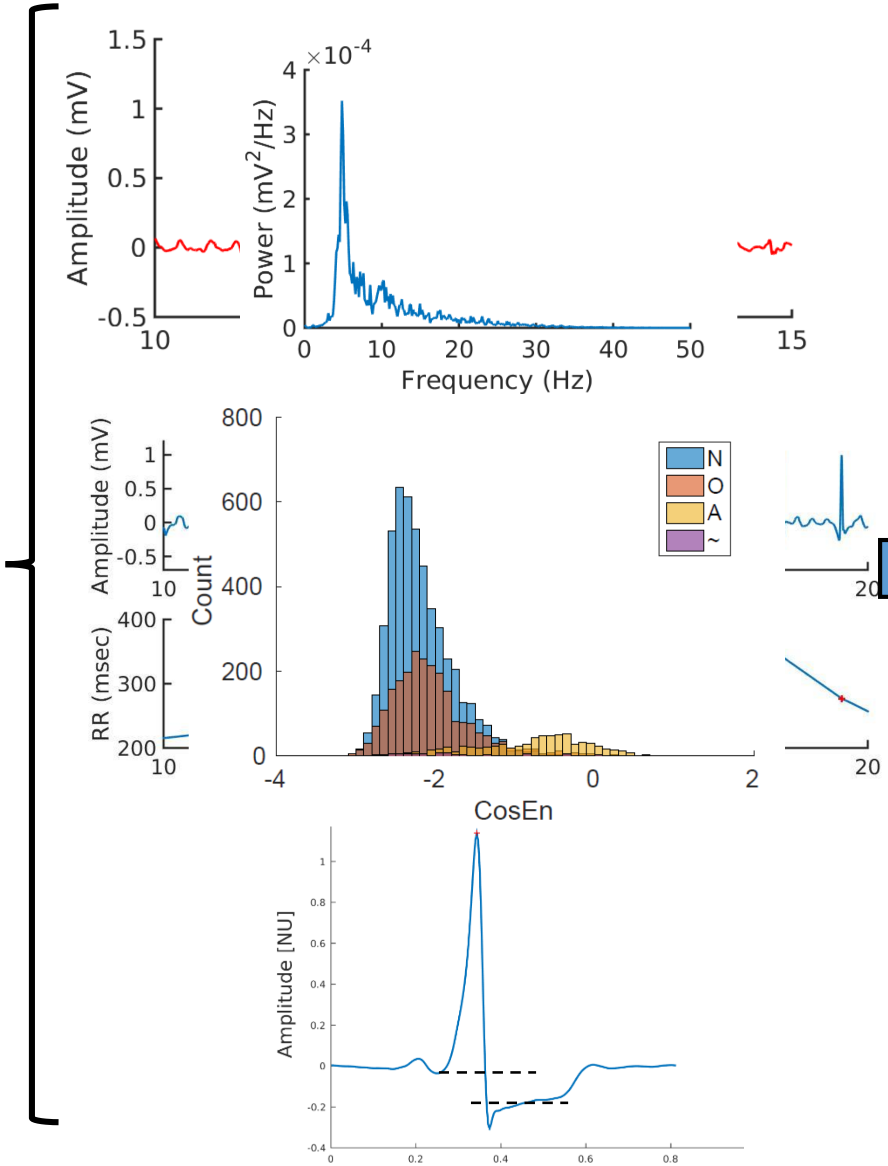
1: positive class (AF).



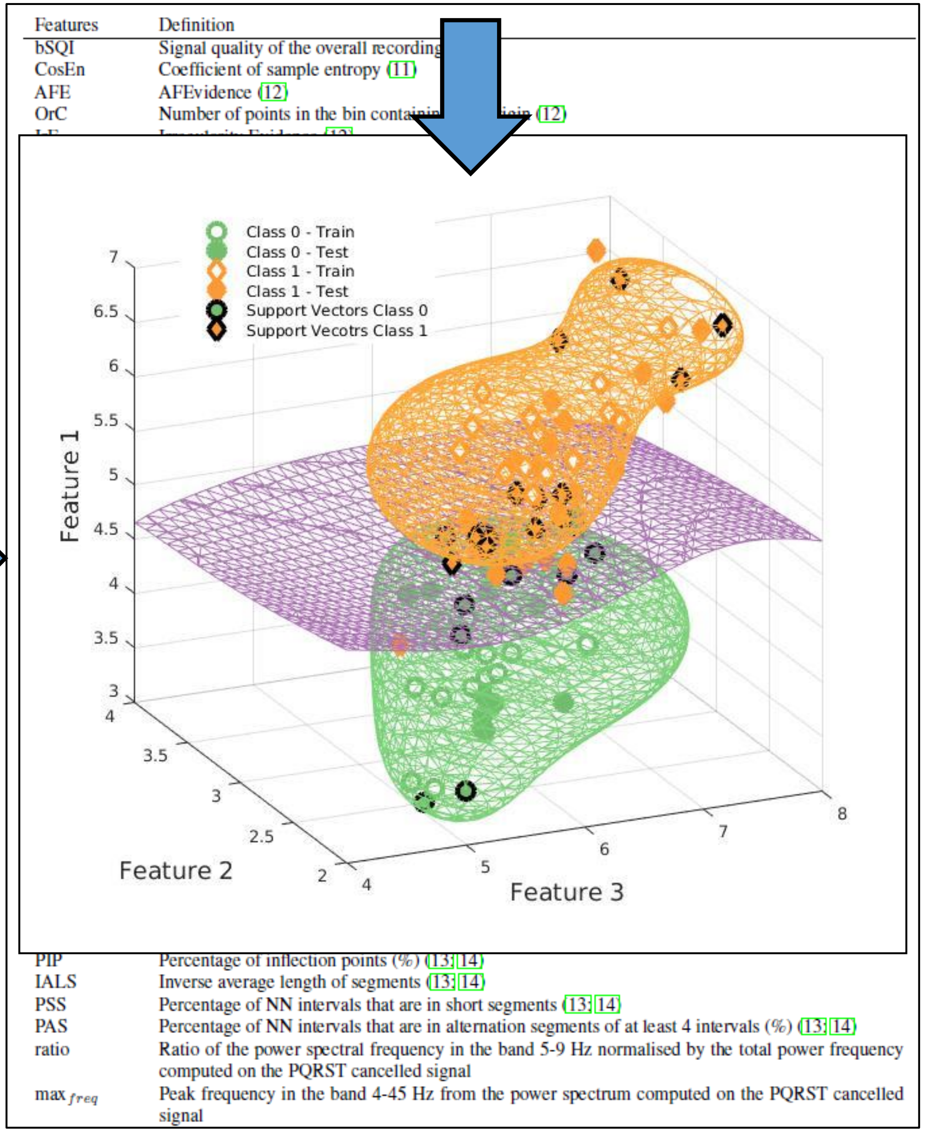
Behar et al., CinC 2017



Physiology



Physio-Features



Behar et al., CinC 2017

Classification

What do we choose?
For now we will stay linear!

Unknown
Target Function: $f: x \to y$

Training Examples
$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
$\{y^{(1)}, y^{(2)}, \dots, y^{(m)}\}$

Hypothesis Class
$H = \{h\}$

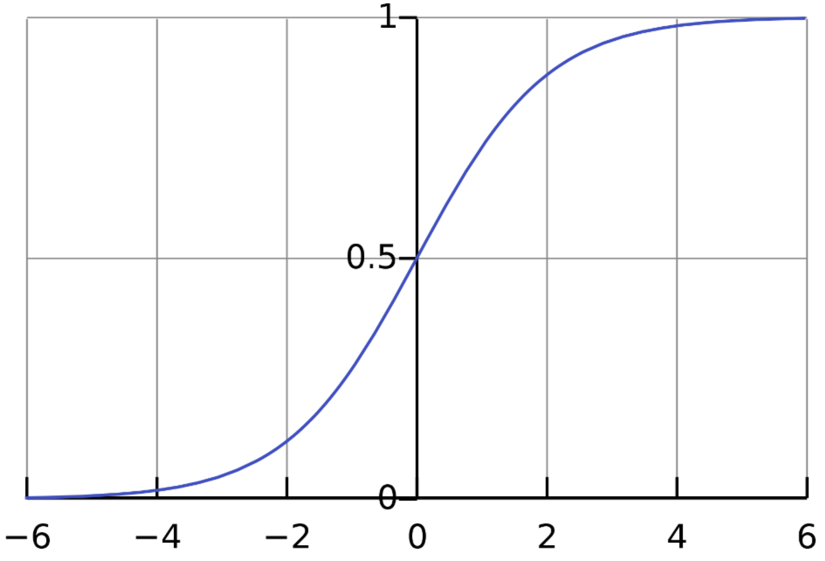
Classification Algorithm

Final Hypothesis
$\hat{h} \approx f$

LR hypothesis representation

Hypothesis representation

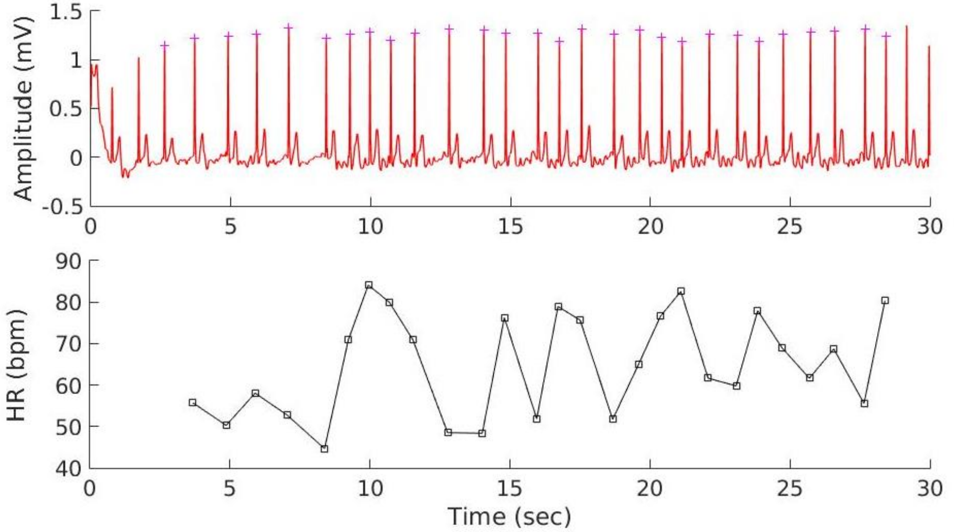
- Linear regression: $h_w(x) = w^T x$
- Logistic regression: $h_w(x) = g(w^T x)$
  - With $g(z) = \frac{1}{1+e^{-z}} = \sigma(z)$ the **sigmoid function**.
  - $\lim_{z \to +\infty} \sigma(z) = 1$,
  - $\lim_{z \to -\infty} \sigma(z) = 0$,
  - $\sigma(0) = 0.5$.

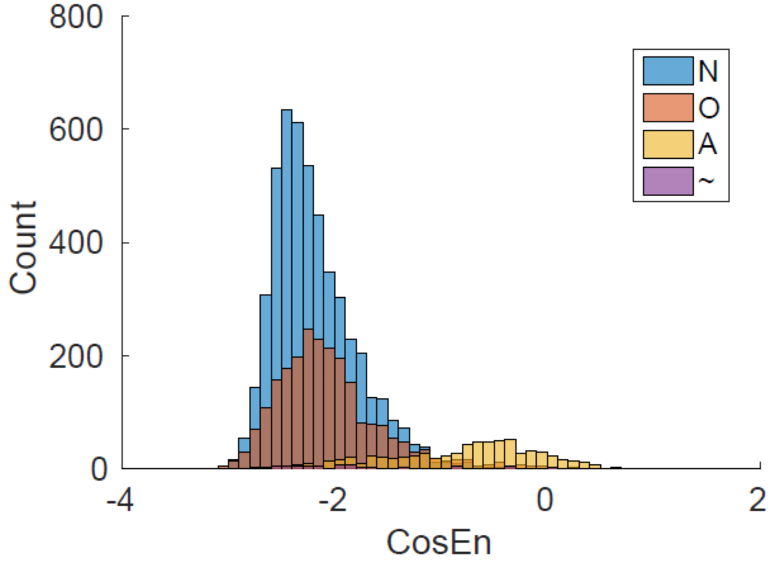


Remark: “Logistic regression” is not a “regression” algorithm but a classification one. The naming is just historical (and somewhat confusing!).

Hypothesis representation

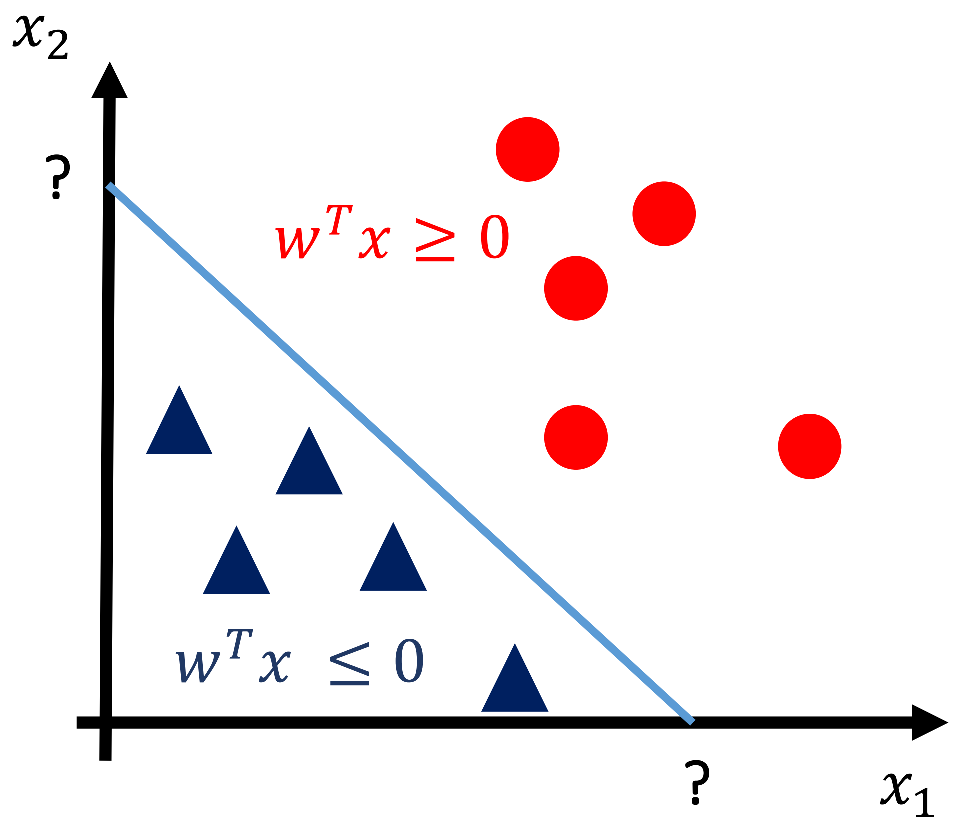
- Interpretation of the **probabilistic output**:
  - $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ CosEn \end{bmatrix}$,
  - $h_w(x) = 0.7 = P(y = 1|x, w)$,
  - This individual has 70% chance to have AF.





Hypothesis representation

- Interpretation of the **decision boundary**:
  - $h_w(x) = \sigma(w^T x) = \frac{1}{1+e^{-w^T x}}$
- Example:
  - $y = 1$ if $h_w(x) \geq 0.5 \iff w^T x \geq 0$
  - Conversely, $y = 0$ if $h_w(x) \leq 0.5 \iff w^T x \leq 0.$
  - This gives the decision boundary.
- $e.g.\ h_w(x) = \sigma(-3 + x_1 + x_2)$



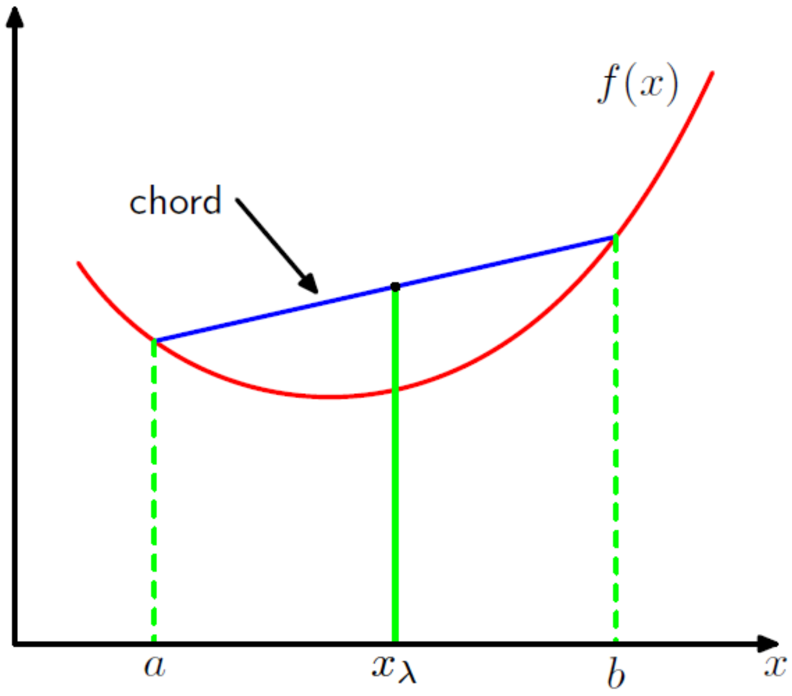
LR Cost Function

Cost function

- We have a training set of
  - m examples: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
  - With target labels: $\{y^{(1)}, y^{(2)}, \dots, y^{(m)}\}$
- How do we find the weights w of the LR model?
  - Reminder in linear regression:
    - $J(w) = \frac{1}{m}\sum_{i=1}^{m}(h_w\left(x^{(i)}\right) - y^{(i)})^2$
  - In LR we have $h_w(x) = \sigma(z)$ and thus the resulting cost function $J(w)$ is **non-convex**. Thus we need to find another cost function that is convex.

Reminder: convex function

Figure 1.31 A convex function $f(x)$ is one for which every chord (shown in blue) lies on or above the function (shown in red).

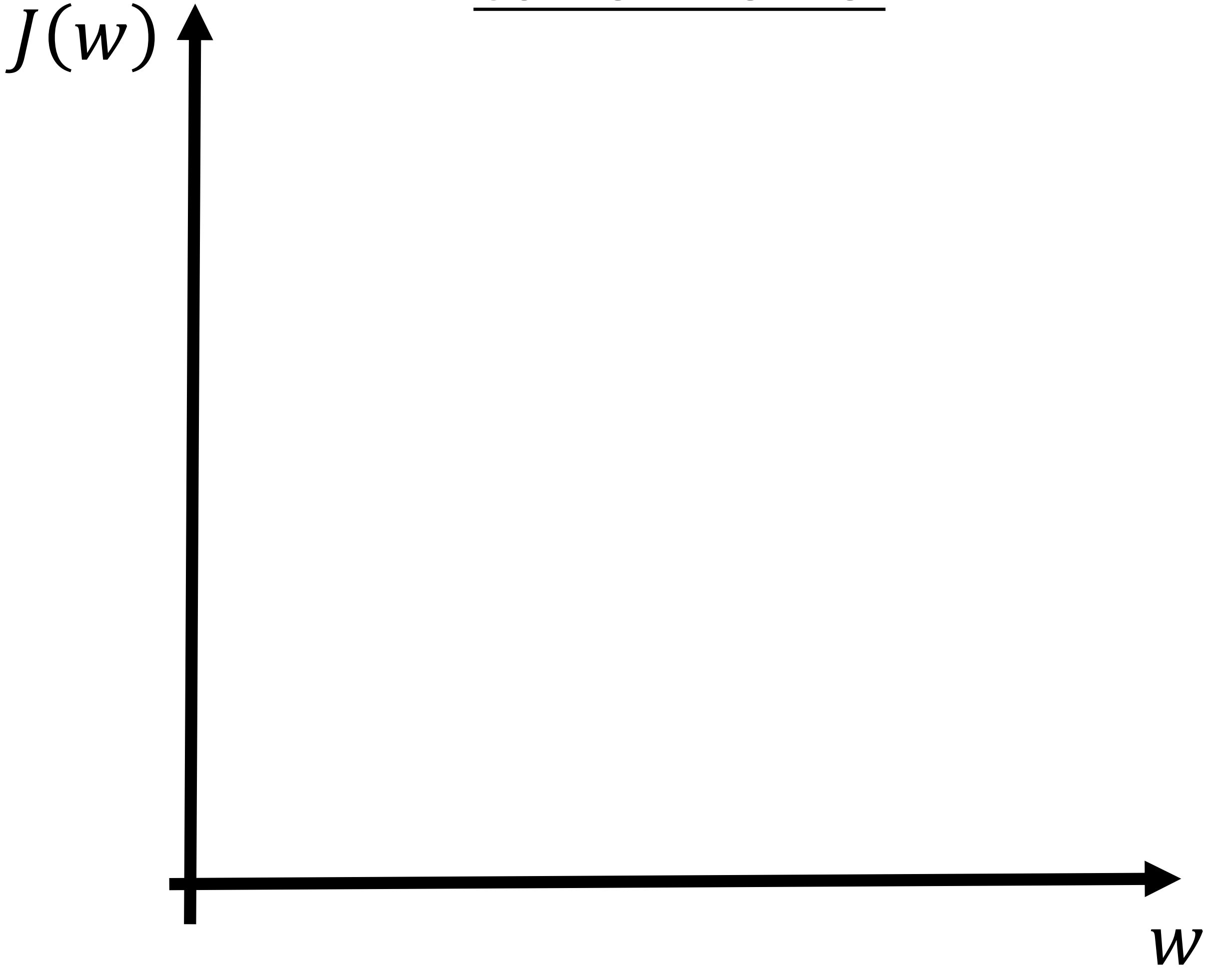


$$f(\lambda a + (1-\lambda)b) \leqslant \lambda f(a) + (1-\lambda)f(b).$$

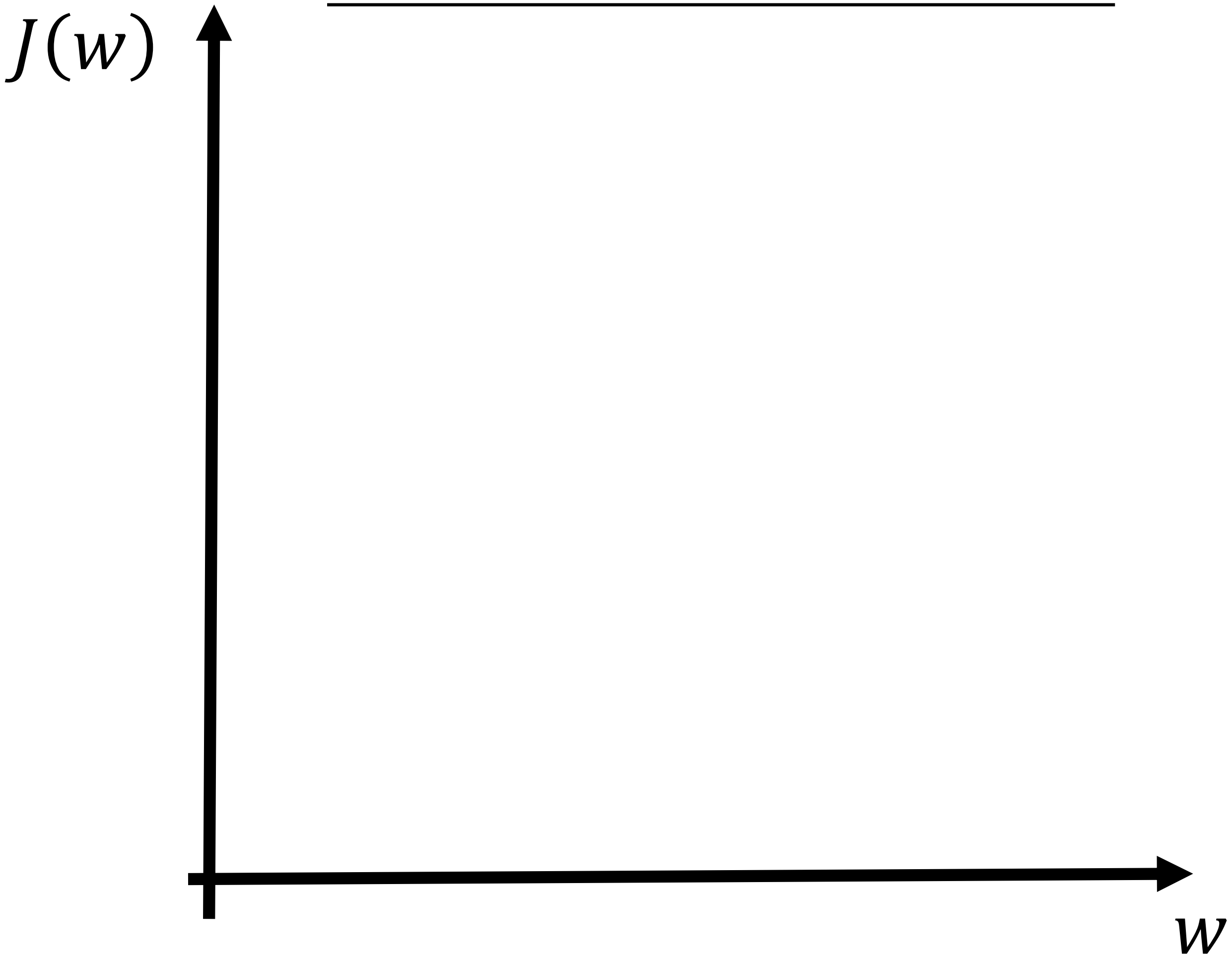
Image: From Bishop's Pattern recognition and machine learning, Figure 1.31

Reminder: convex function

Convex: we like!



Non-convex: we want to avoid!



Cost function in LR

- We define the following **error**:

$$E_w(x,y) = \begin{cases} -log\big(h_w(x)\big), if\ y = 1 \\ -log\big(1 - h_w(x)\big), if\ y = 0, \end{cases}$$

  - If $y = 1$ and $h_w(x) \to 0$ then $E_w(x,y) \to \infty$
  - If $y = 1$ and $h_w(x) \to 1$ then $E_w(x,y) \to 0.$
  - Ibid $y = 0.$
- We can re-write it:
  - $E_w(x,y\,) = -ylog\big(h_w(x)\big) - (1-y)\log(1-h_w(x))$

AIMLab.

Cost function in LR

- The **cost function**:
  - $J(w) = \frac{1}{m}\sum_{i=1}^{m} E_w\left(x^{(i)}, y^{(i)}\right),$
  - $J(w) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)} log\left(h_w\left(x^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log(1 - h_w\left(x^{(i)}\right))\right].$
- It is possible to show that the cost function $J(w)$ is convex.
- This is called the **Cross-Entropy** cost function or log loss.

Cost function in LR

- Why do we choose this particular error definition?
  - Maximum likelihood estimate.
    - $argmax_w \mathcal{L}(w|Y,X) = argmax_w (\prod_{i=1}^{m} \mathcal{L}(w|y^{(i)}, x^{(i)}))$
  - Convex cost function.

Gradient descent

Optimization algorithm

- We want to use some optimization algorithm to solve our optimization problem given the cost function we defined.
- Optimization is about maximizing/minimizing an objective function $g(x)$ parametrized by x.
- In machine learning we are interesting in minimizing $J(w)$ parametrized by w.
- We want to find $min_w\big(J(w)\big)$
- For that purpose we will use **gradient descent**.

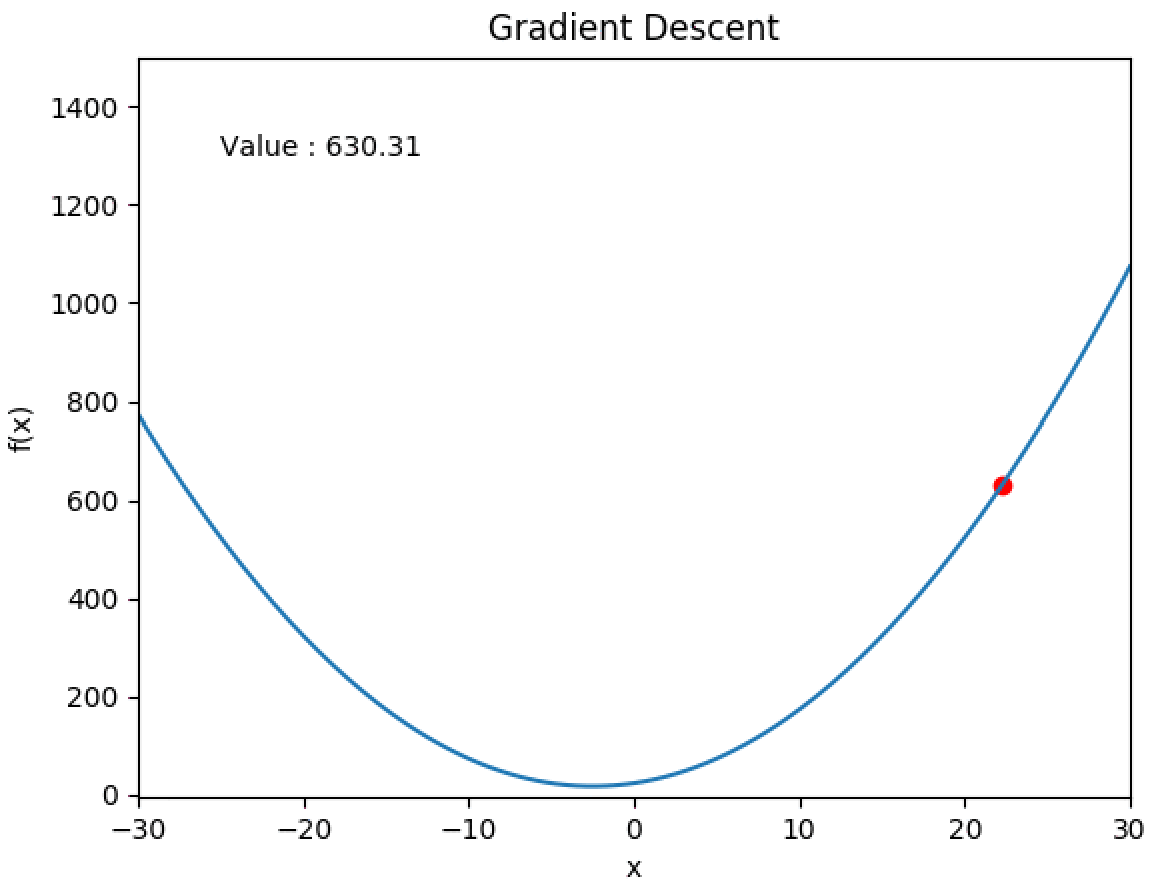
Optimization algorithm

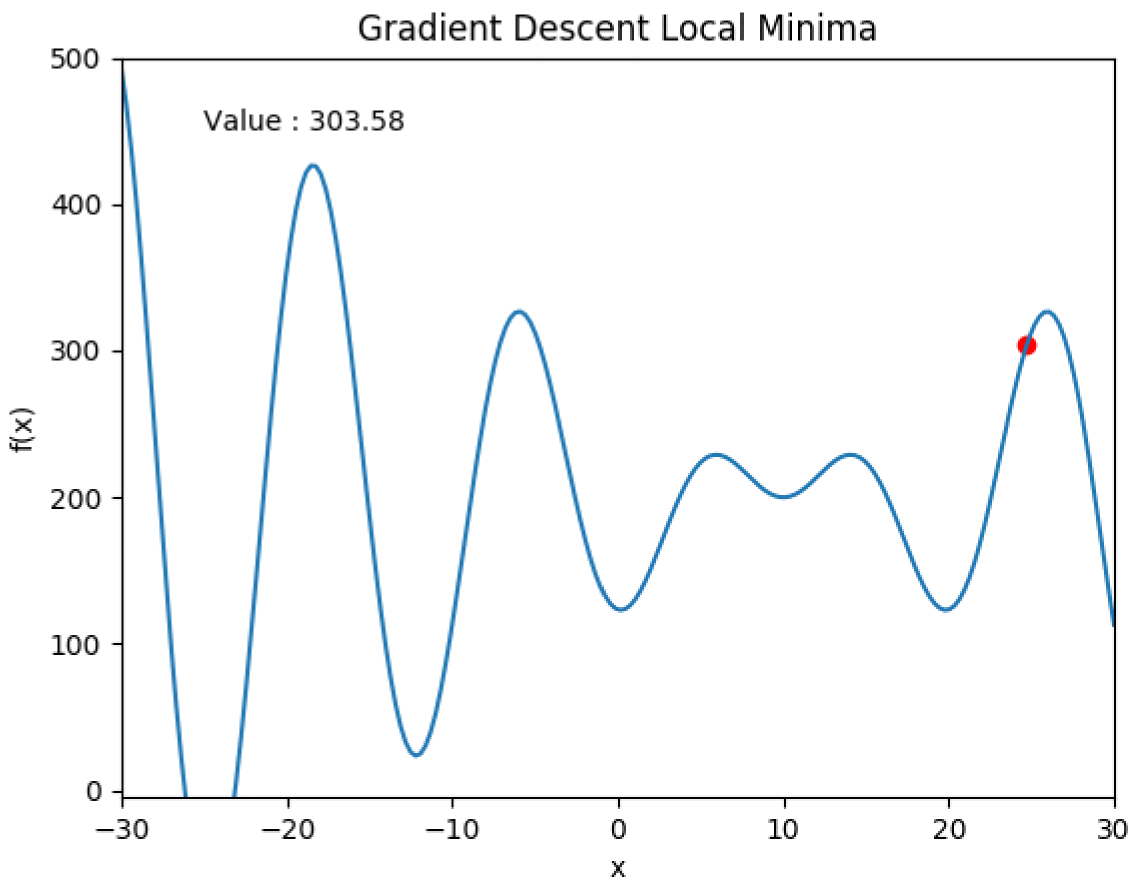
- Gradient descent, update w_j:

$$w_j \coloneqq w_j\text{-}\alpha\frac{\partial J(w)}{\partial w_j}$$

- More sophisticated alternative to gradient descent (but based on gradient descent) exist: conjugate gradient, BFGS, L-BFGS etc.

Optimization algorithm



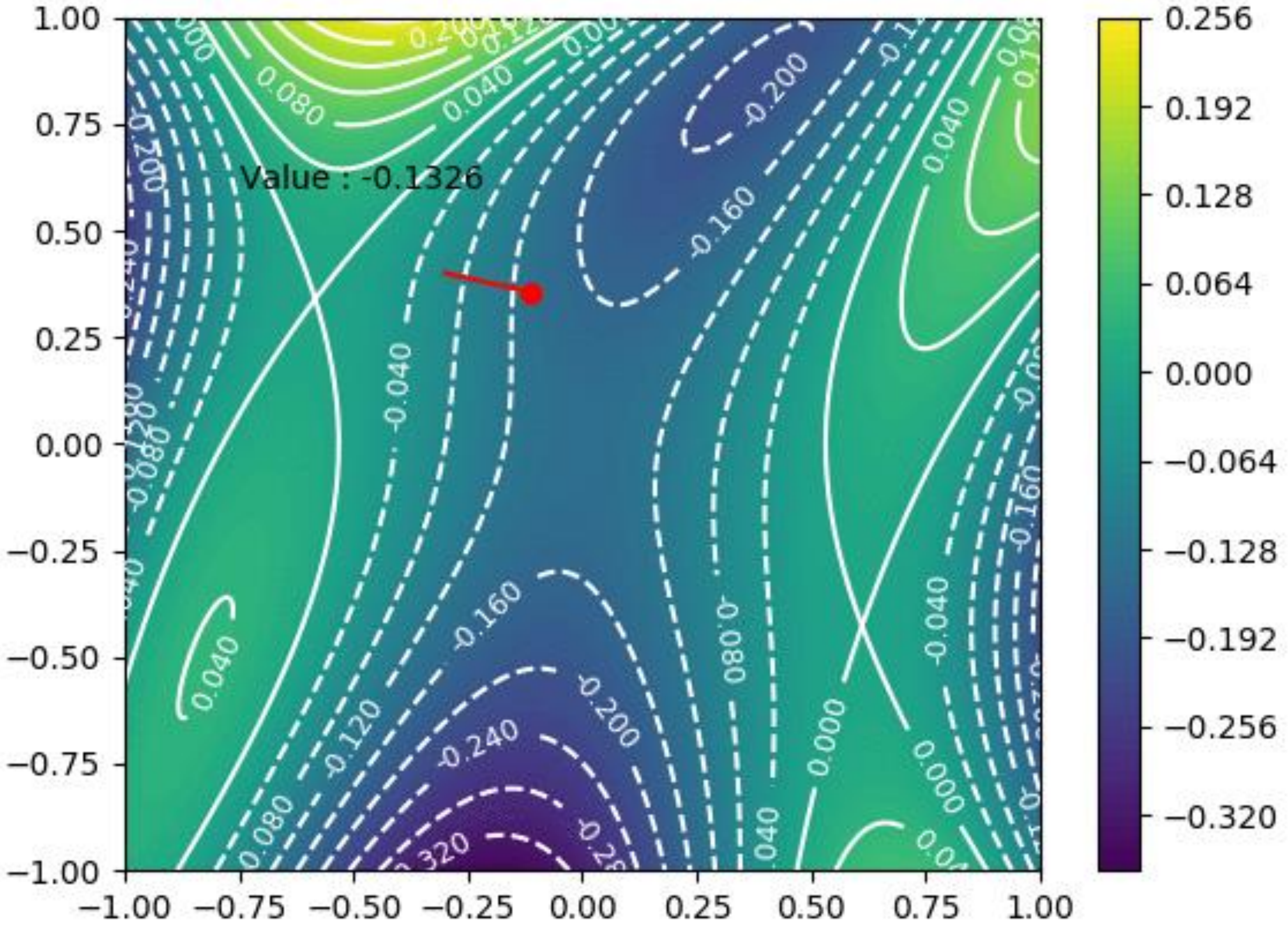


https://github.com/Shathra/gradient-descent-demonstration

Optimization algorithm

- How do we use gradient descent with LR? Snapshot here.
- The overall cost function:
  - $J(w) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}log\left(h_w\left(x^{(i)}\right)\right) - (1 - y^{(i)})\log(1 - h_w\left(x^{(i)}\right))\right]$
- We need to compute $\frac{\partial J(w)}{\partial w_j}, \forall\ j \in [1, \dots, n_x]$:
  - $\frac{\partial J(w)}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$
- Gradient descent, update w_j:
  - $w_j := w_j\text{-}\alpha\frac{\partial J(w)}{\partial w_j} = w_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$
- What about feature scaling? Yes, we need it!

Optimization algorithm



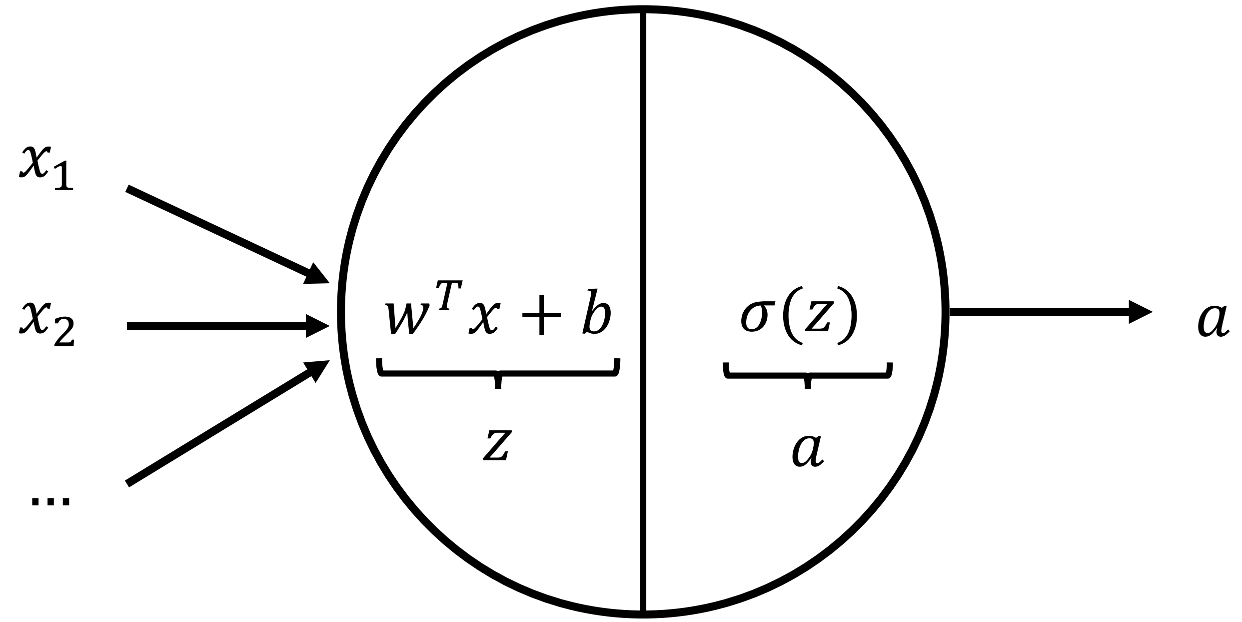
https://github.com/Shathra/gradient-descent-demonstration

LR gradient descent

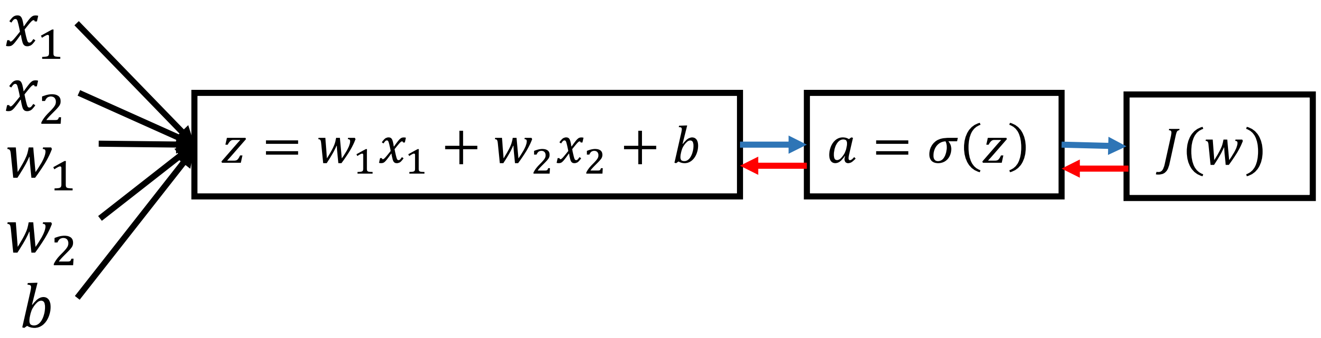
AIMLab.

Logistic regression

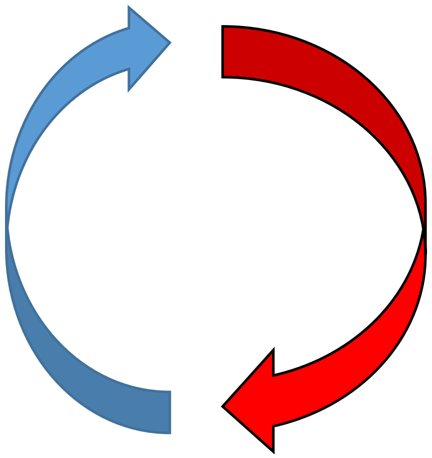
- Equations:
  - $z = w^T x + b$
  - $a = h_w(z) = \sigma(z)$
- Cost function:
  - $J(w) = -ylog\left(h_w(x)\right) - (1-y)\log(1-h_w(x))$
  - $J(w) = -ylog(a) + (1-y)\log(1-a)$



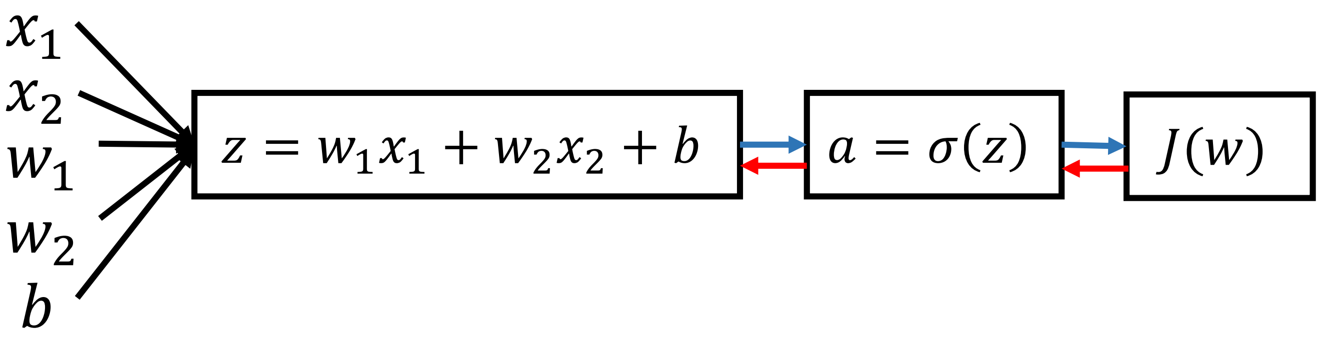
Logistic regression equations



- Forward propagation:
  - $z = w^T x + b$
  - $a = \sigma(z)$
- Backward propagation:
  - $w_1 := w_1 - \alpha \frac{\partial J(w)}{\partial w_1} = w_1 - \alpha(a-y)x_1$
  - $w_2 := w_2 - \alpha \frac{\partial J(w)}{\partial w_2} = w_2 - \alpha(a-y)x_2$
  - $b := b - \alpha \frac{\partial J(w)}{\partial b} = b - \alpha(a-y)$
- Iterate between forward and backward steps.

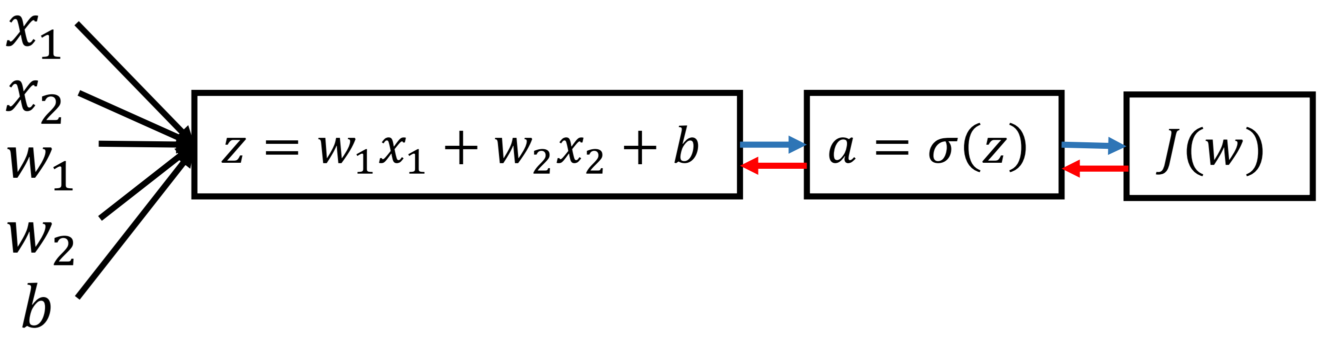


Logistic regression equations



- Now consider m examples
  - Forward propagation,
    - $z^{(i)} = w^T x^{(i)} + b, \forall i \in [1, m]$
    - $a^{(i)} = \sigma(z^{(i)}), \forall i \in [1, m]$
  - Backward propagation:
    - $w_1 := w_1 - \frac{1}{m} \sum_{i=1}^{m} \alpha \left(a^{(i)} - y^{(i)}\right) x_1^{(i)}$
    - $w_2 := w_2 - \frac{1}{m} \sum_{i=1}^{m} \alpha \left(a^{(i)} - y^{(i)}\right) x_2^{(i)}$
    - $b := b - \frac{1}{m} \sum_{i=1}^{m} \alpha \left(a^{(i)} - y^{(i)}\right).$

Logistic regression equations



- Now consider n_x input features:
  - Forward propagation:
    - $z^{(i)} = w^T x^{(i)} + b, \forall i \in [1, m]$
    - $a^{(i)} = \sigma(z^{(i)}), \forall i \in [1, m]$
  - Backward propagation:
    - $w_j \coloneqq w_j - \frac{1}{m}\sum_{i=1}^{m} \alpha\left(a^{(i)} - y^{(i)}\right)x_j^{(i)}, \forall j \in [1, n_x]$
    - $b \coloneqq b - \frac{1}{m}\sum_{i=1}^{m} \alpha\left(a^{(i)} - y^{(i)}\right).$
- So if we perform k iterations of gradient descent we need to go through two loops of m (forward) and then n (backward) steps. Can we vectorize?

How can we vectorise?

- Forward propagation
  - $z^{(1)} = w^T x^{(1)} + b$
  - $a^{(1)} = \sigma(z^{(1)})$
  - $z^{(2)} = w^T x^{(2)} + b$
  - $a^{(2)} = \sigma(z^{(2)})$
  - …

- Vectorized form of forward propagation:
  - $z = w^T X + b$
    - $z \in \mathbb{R}^m, z = \left[z^{(1)}, \dots, z^{(m)}\right]$
    - $w \in \mathbb{R}^{n_x}, w = [w_1, \dots, w_{n_x}]$
    - $X \in \mathbb{R}^{n_x \cdot m}, X = \left[x^{(1)}, \dots, x^{(m)}\right]$
    - $b \in \mathbb{R}^m, b = [b, b \dots, b]$

How can we vectorise?

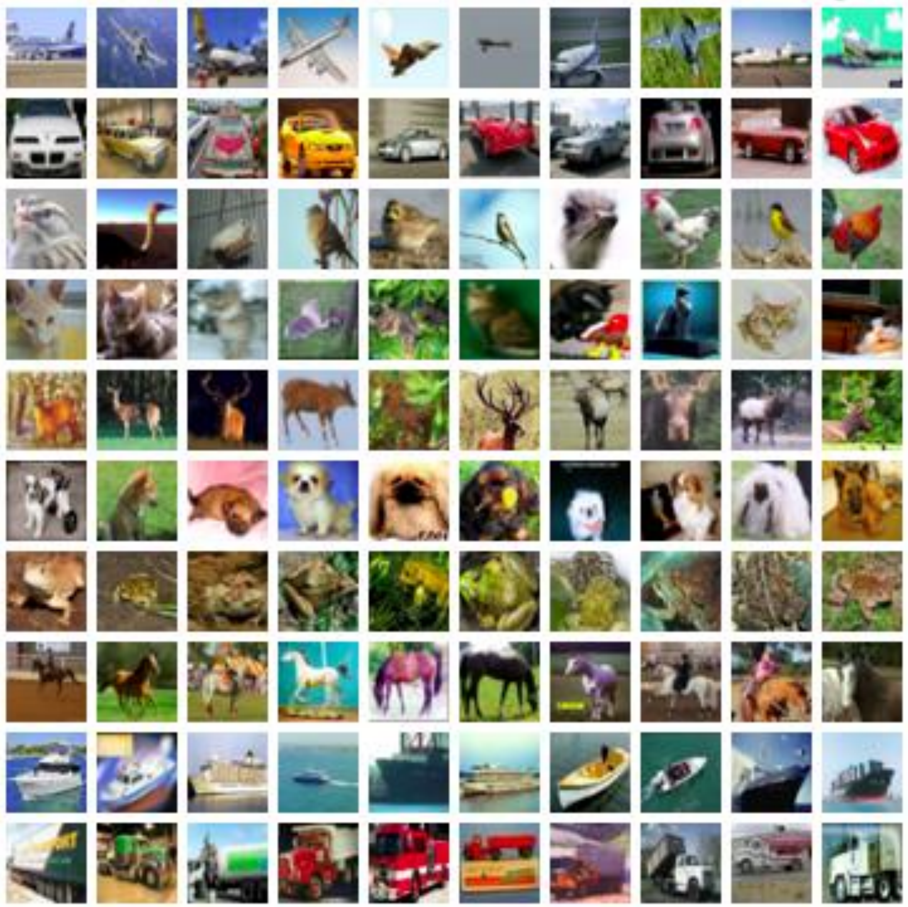
- Backward propagation:
  - $w_j \coloneqq w_j - \frac{1}{m}\sum_{i=1}^{m} \alpha\left(a^{(i)} - y^{(i)}\right)x_j^{(i)}, \forall j \in [1, n]$
- Vectorized backward propagation:
  - $w \coloneqq w - \alpha \frac{1}{m} X \left(\underline{a} - \underline{y}\right)$
  - $\underline{a} \in \mathbb{R}^m, \underline{y} \in \mathbb{R}^m, X \in \mathbb{R}^{n_x \cdot m}$

How can we vectorise?

- In conclusion, **the vectorized form of LR gradient descent**:
  - $z = w^T X + b$ $\quad (forward\ step)$
  - $w := w - \alpha \frac{1}{m} X \left(\underline{a} - \underline{y} \right)$ $\quad (backward\ step)$
  - $b := b - \frac{1}{m} \sum_{i=1}^{m} \alpha \left(a^{(i)} - y^{(i)} \right)$ $\quad (backward\ step)$

Multiclass classification

Multiclass classification



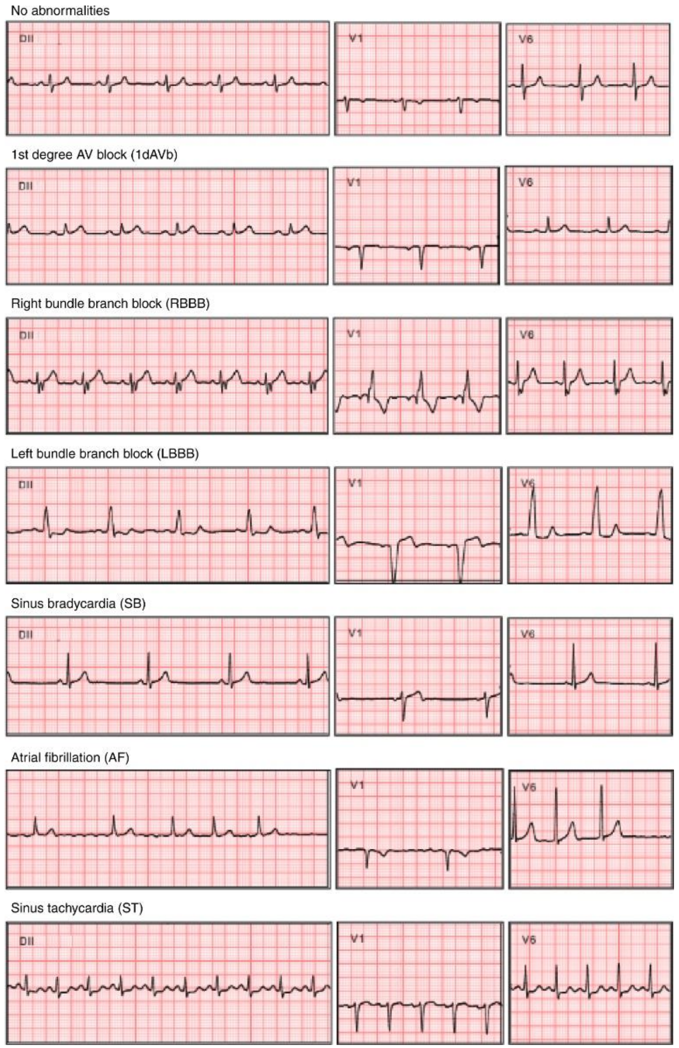
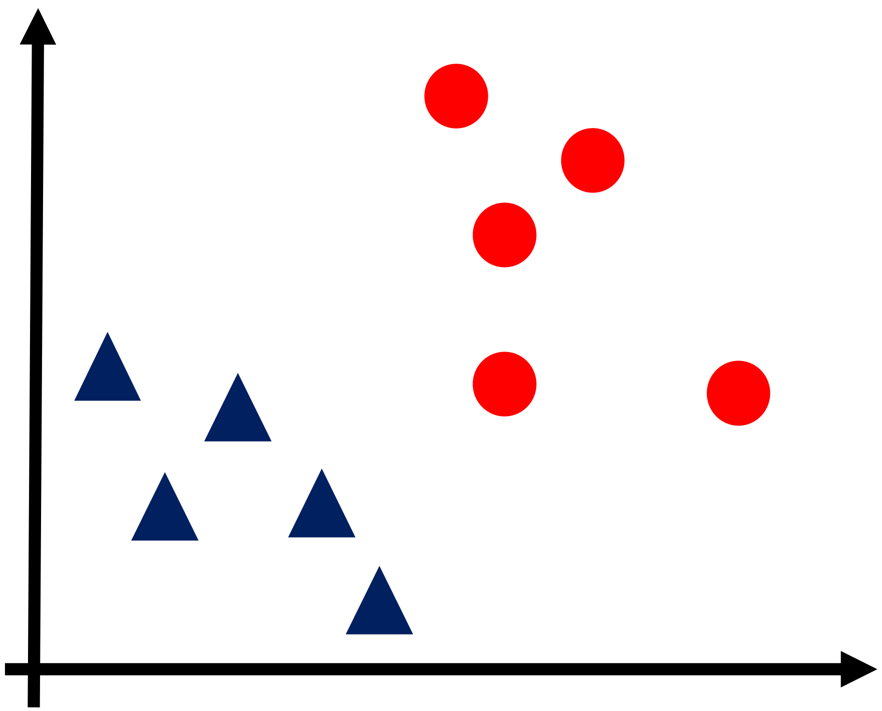


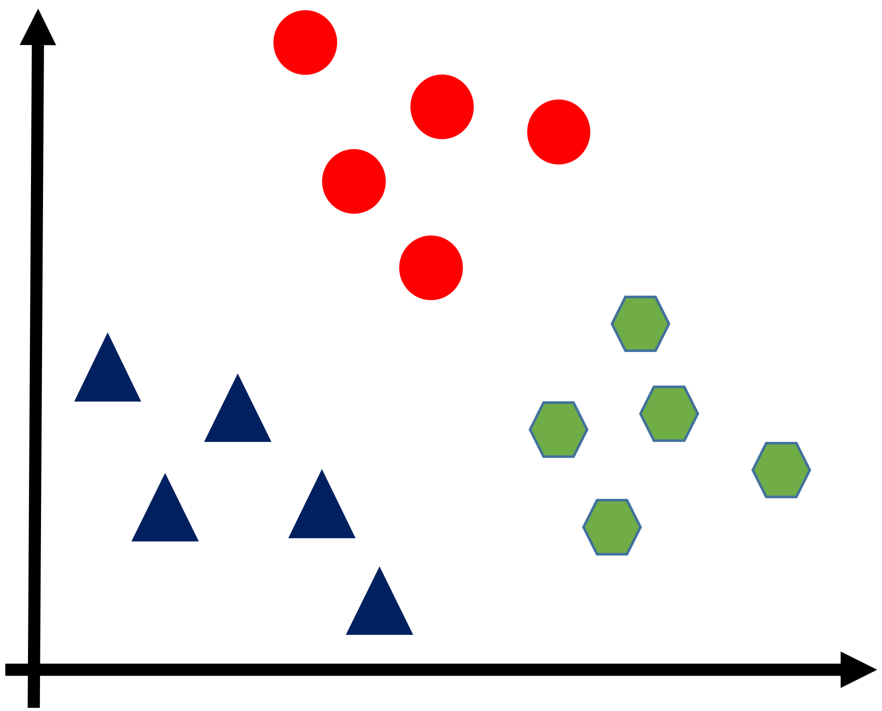
Image (left): https://analyticsindiamag.com/transfer-learning-for-multi-class-image-classification-using-deep-convolutional-neural-network/

Image (right): Ribeiro, Antônio H., et al. "Automatic diagnosis of the 12-lead ECG using a deep neural network." Nature communications 11.1 (2020): 1-9.

Multiclass classification

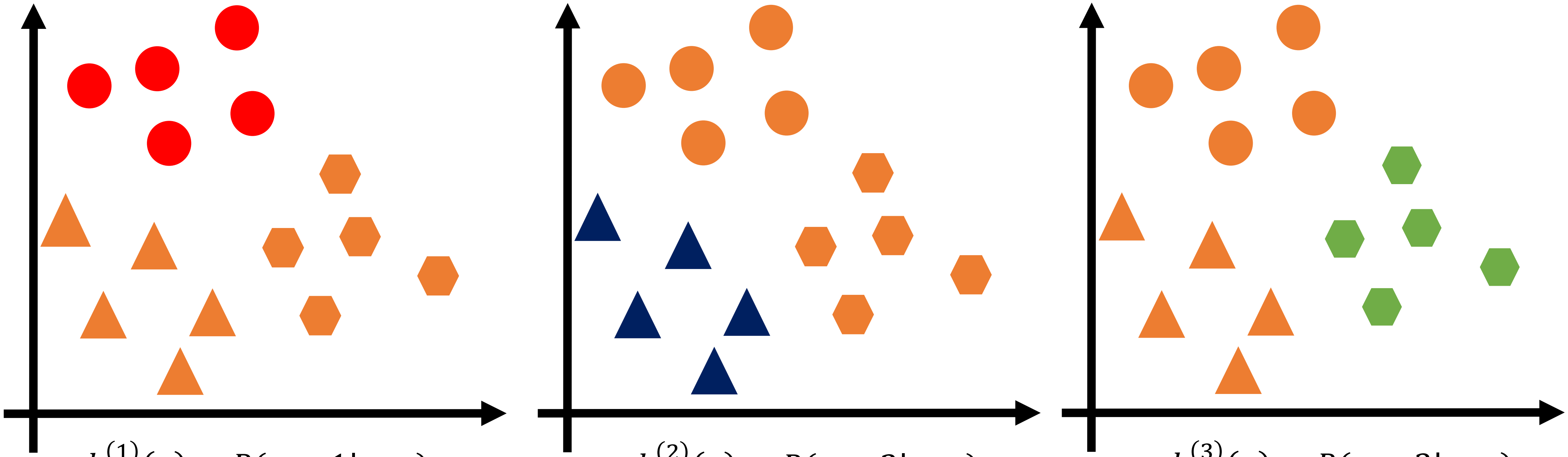
- We are now interested in a problem where the output is not binary.
- Consider the arrhythmia example. Say we now want to distinguish between categories: Atrial fibrillation (AF), other arrythmias (ARR) and normal sinus rhythm (NSR).





Multiclass classification

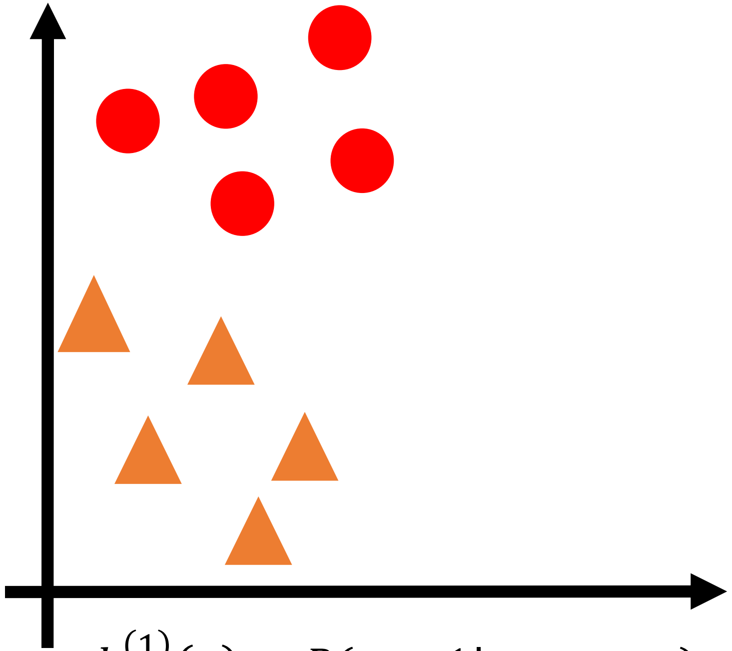
- How do we do that?
- “**One vs. all**” also called “**one vs. the rest**” approach.
- Transform the problem into a set of 2-class classification problems:



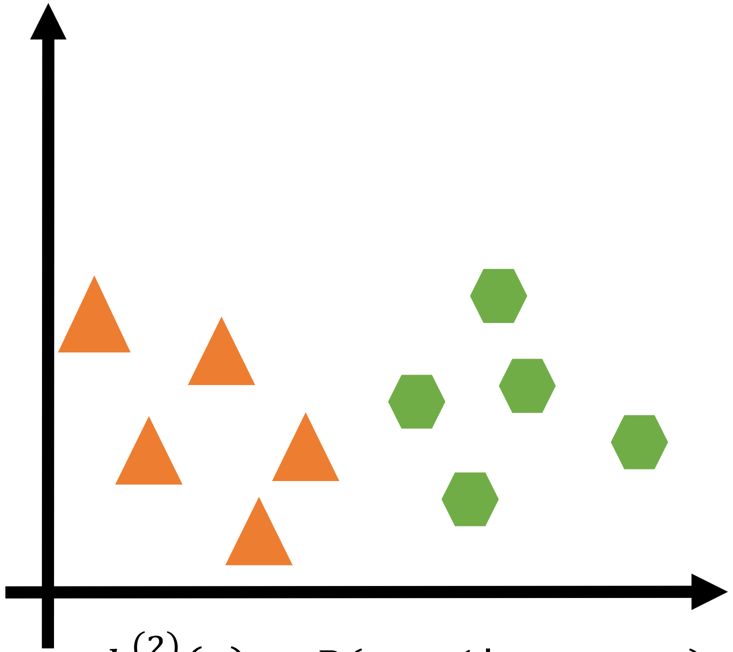
$h_w^{(1)}(x) = P(y = 1|x, w)$ $h_w^{(2)}(x) = P(y = 2|x, w)$ $h_w^{(3)}(x) = P(y = 3|x, w)$

Multiclass classification

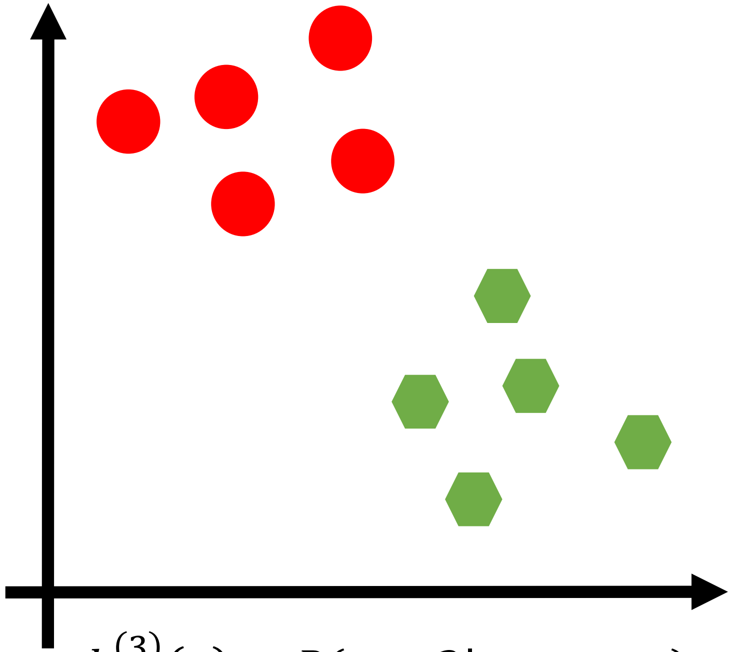
- "One vs. one".
- Transform the problem into a set of 2-class classification problems:



$h_w^{(1)}(x) = P(y = 1 | x_{\{C_1, C_2\}}, w)$



$h_w^{(2)}(x) = P(y = 1 | x_{\{C_1, C_3\}}, w)$



$h_w^{(3)}(x) = P(y = 2 | x_{\{C_2, C_3\}}, w)$

...

Multiclass classification

- Can be computationally expensive.
- Problem of ambiguous region.
  - How can we resolve?

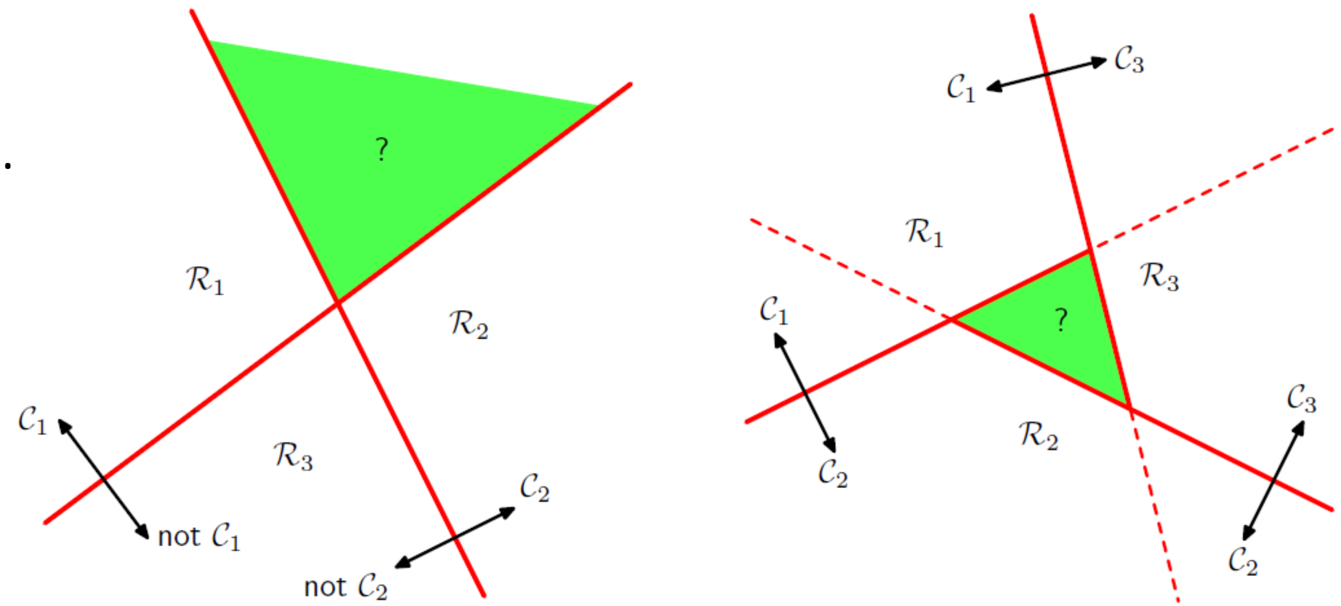
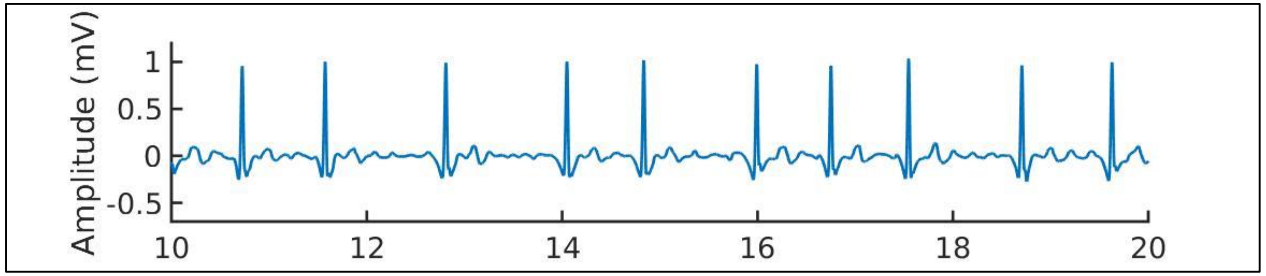


Figure 4.2 Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class $\mathcal{C}_k$ from points not in class $\mathcal{C}_k$. On the right is an example involving three discriminant functions each of which is used to separate a pair of classes $\mathcal{C}_k$ and $\mathcal{C}_j$.

Image: From Bishop's Pattern recognition and machine learning, Figure 4.2, p183

Multiclass classification

- On the example of the one vs. the rest
- So we come up with 3-classifiers, each classifier is trained to recognize one of the three classes.
- How do we classify a new observation x?
  - $max_k \left(h_w^{(k)}(x) \right)$
- Example:
  - You record:



Multiclass classification

- We saw the **one vs. the rest** and **one vs. one** approaches.
- These approaches can be used to generalize any binary classifier to a multiclass setting.
- Can we actually change the classifier (LR here) hypothesis class and cost function to be multiclass?
  - **Multinomial LR**. (Also called multiclass LR or softmax regression).

Multiclass classification

- Two class classification: $z = w^T x + b$
- Multinomial: $z = Wx + b$
  - $z, b \in \mathbb{R}^{n_y}$,
  - $W \in \mathbb{R}^{n_y \cdot n_x}$,
- From the z **vector** how do we classify? **Softmax** activation function:
  - $\mathrm{a} = softmax(\mathrm{z}) = e^{\mathrm{z}} / \sum_{\mathrm{k}=1}^{n_y} e^{z_k}$
- The softmax is also called normalized exponential function. It is a function that takes an input vector of size $n_y$and normalizes it into $n_y$probability distribution proportional to the exponentials of the input numbers.

https://scikit-learn.org/stable/auto_examples/linear_model/plot_logistic_multinomial.html

Multiclass classification

- Cost function we add for the two-class classification:
  - $J(w) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}log\left(h_w\left(x^{(i)}\right)\right) - (1 - y^{(i)})\log(1 - h_w\left(x^{(i)}\right))\right],$
- For the multinomial LR:
  - $J(w) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{n_y}[1\{y^{(i)} = k\}\log(h_{w_k}\left(x^{(i)}\right))],$
  - $J(w) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{n_y}\left[1\{y^{(i)} = k\}\log(\frac{\exp(w^{(k)T}x^{(i)})}{\sum_{j=1}^{K}\exp(w^{(k)T}x^{(i)})})\right].$
  - Thus two differences with what we had previously:
    - The cost function sums over the number of classes.
    - We use the $softmax$ activation function and not σ.

Multiclass classification

- We saw the **one vs. the rest** and **one vs. one** approaches which are general approaches for any binary classifier.
- **Multinomial LR** which is a generalization to multiclass specific to the LR model.
- What about if the classes are not exclusives?

AIMLab.

Take home

- Classification versus regression.
- **Hypothesis representation** and **Cross-entropy cost function.**
- **Convexity** of the cost function.
- Optimization using **gradient descent**.
- Multiclass classification:
  - General framework: **one vs. all, one vs. one,**
  - Model (LR) specific: **multinomial.**
- LR is one of the most popular classification algorithm. Use it as a baseline.
  - Advantages: efficient, interpretable, outputs probabilities.
  - Drawback: cannot solve non-linear problems since the LR decision surface is linear.

References

[1] Andrew Ng, Coursera, Machine Learning. Coursera.

[2] Andrew Ng, Coursera, Neural Networks and Deep Learning. Coursera.

[3] CSCE 666 Pattern Analysis | Ricardo Gutierrez-Osuna | CSE@TAMU

URL: http://research.cs.tamu.edu/prism/lectures/pr/pr_l10.pdf

[4] Pattern recognition and Machine Learning. Christopher M. Bishop. 2006 Springer Science.