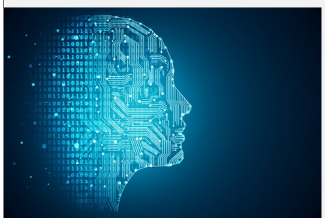


DS 310 Machine Learning
Linear Classifiers: Loss functions

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science and Engineering, Bioinformatics & Genomics, Public Health Sciences and Neuroscience
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

Perceptron objective function

- We did not so far explicitly specify an objective function or loss function for the perceptron
- Can we write down a loss function for the perceptron?

$$E_{0/1}(\mathbf{w}) = \sum_p \max\{-d_p h_{\mathbf{w}}(\mathbf{x}_p), 0\}$$

where

d_p is the label $(+1$ or $-1)$ for sample $\mathbf{x}_p$

$h_{\mathbf{w}}(\mathbf{x}_p) = +1$ if $\mathbf{w} \cdot \mathbf{x}_p > 0$ and $h_{\mathbf{w}}(\mathbf{x}_p) = -1$ if $\mathbf{w} \cdot \mathbf{x}_p < 0$

$-d_p h_{\mathbf{w}}(\mathbf{x}_p) = +1$ if and only if d_p and $h_{\mathbf{w}}(\mathbf{x}_p)$ agree

in which case $\max\{-d_p h_{\mathbf{w}}(\mathbf{x}_p), 0\} = \max\{1,0\} = 1$

$-d_p h_{\mathbf{w}}(\mathbf{x}_p) = -1$ if and only if d_p and $h_{\mathbf{w}}(\mathbf{x}_p)$ disagree

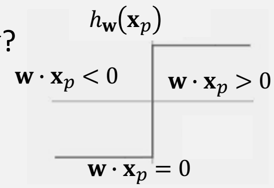
in which case $\max\{-d_p h_{\mathbf{w}}(\mathbf{x}_p), 0\} = \max\{-1,0\} = 0$

The max operation ensures that contribution of a sample $\mathbf{x}_p$ to $E_{0/1}(\mathbf{w})$ is 1 if $h_{\mathbf{w}}(\mathbf{x}_p)$ misclassifies $\mathbf{x}_p$ and it is 0 otherwise.

Perceptron objective function

$$E_{0/1}(\mathbf{w}) = \sum_p \max\{-d_p h_{\mathbf{w}}(\mathbf{x}_p), 0\}$$

The perceptron loss function simply counts the number of misclassified samples.



- Is $E_{0/1}(\mathbf{w})$ differentiable with respect to $\mathbf{w}$?
- No, because $h_{\mathbf{w}}(\mathbf{x}_p)$ is not differentiable with respect to $\mathbf{w}$
- We cannot use gradient descent!
- Nevertheless, there is a non gradient based algorithm, namely, Rosenblatt's perceptron algorithm which is guaranteed to converge to a separating hyperplane (as we proved)
  - but only when the classes are separable.

Perceptron Algorithm

- Perceptron algorithm is guaranteed to converge to a separating hyperplane whenever the training data are linearly separable.
- What if the training data are not linearly separable?
  - All bets are off.
  - The algorithm runs for ever, cycling indefinitely trying to correct errors that cannot be corrected (proof omitted)
- Can we come up with an algorithm that converges when the data are separable, and achieves a reasonable compromise solution when the data are not separable?
  - Yes, as we shall see next

Can we define an alternative differentiable loss function?

$g_{\mathbf{w}}(\mathbf{x}_p) = \mathbf{w} \cdot \mathbf{x}_p$

$h_{\mathbf{w}}(\mathbf{x}_p) = +1$ if $\mathbf{w} \cdot \mathbf{x}_p > 0$ and $h_{\mathbf{w}}(\mathbf{x}_p) = -1$ if $\mathbf{w} \cdot \mathbf{x}_p < 0$

Let $E_p(\mathbf{w}) = \max\{-d_p g_{\mathbf{w}}(\mathbf{x}_p), 0\}$

$E_{soft}(\mathbf{w}) = \sum_p E_p(\mathbf{w})$

where d_p is the label ($+1$ or -1) for sample $\mathbf{x}_p$

$-d_p g_{\mathbf{w}}(\mathbf{x}_p) =$

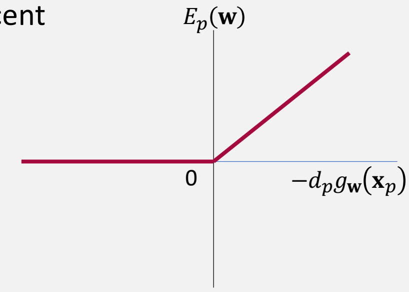
- $+g_{\mathbf{w}}(\mathbf{x}_p)$ if d_p and $g_{\mathbf{w}}(\mathbf{x}_p)$ are of different signs
- $-g_{\mathbf{w}}(\mathbf{x}_p)$ if d_p and $g_{\mathbf{w}}(\mathbf{x}_p)$ are of same sign

The max operation ensures that contribution of a sample $\mathbf{x}_p$ to $E_{soft}(\mathbf{w})$ is

- $g_{\mathbf{w}}(\mathbf{x}_p)$ whenever $h_{\mathbf{w}}(\mathbf{x}_p)$ misclassifies $\mathbf{x}_p$ and
- 0 otherwise.

Can we define an alternative loss function?

- We can show that $E_{soft}(\mathbf{w}) = \sum_p \max\{-d_p g_{\mathbf{w}}(\mathbf{x}_p), 0\}$ is convex, continuous, and has first order derivatives with respect to $\mathbf{w}$ except where $g_{\mathbf{w}}(\mathbf{x}_p)$=0.
- So we can minimize $E_{soft}(\mathbf{w})$ with respect to $\mathbf{w}$ using (sub) gradient descent



Vector and matrix calculus

Scalar analog

$f(x)$	$\frac{df}{dx}$
ax	a
x^2	$2x$
ax^2	$2ax$
e^{ax}	ae^{ax}

Vector or Matrix counterpart

$f(\mathbf{w})$	$\frac{df}{d\mathbf{w}}$
$\mathbf{W}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{W}^T a$	a
$\mathbf{W}^T\mathbf{W}$	$2\mathbf{W}$
$\mathbf{w}^T\mathbf{B}\mathbf{w}$	$2\mathbf{B}\mathbf{w}$
$\mathbf{a}\cdot\mathbf{w}$	$\mathbf{a}$
$e^{\mathbf{a}\cdot\mathbf{w}}$	$\mathbf{a}e^{\mathbf{a}\cdot\mathbf{w}}$

a scalar constant
x scalar variable
$\mathbf{w}$ vector variable
$\mathbf{A}$ constant matrix
$\mathbf{B}$ a constant square matrix
$\mathbf{W}$ a square matrix variable
$\mathbf{a}$ a constant vector

Reference: http://www.cs.cmu.edu/~mgormley/courses/10601/slides/10601-matrix-calculus.pdf

An alternative loss function $E_{soft}(\mathbf{w})$

- We can show that $E_{soft}(\mathbf{w}) = \sum_p \max\{-d_p g_{\mathbf{w}}(\mathbf{x}_p), 0\}$ is convex, and differentiable with respect to $\mathbf{w}$ except at loss = 0.
- So we can minimize $E_{soft}(\mathbf{w})$ with respect to $\mathbf{w}$ using (sub)gradient descent

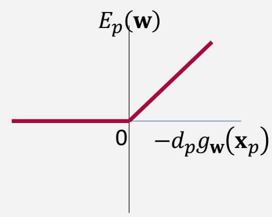
$$\nabla_{\mathbf{w}} E_{soft} = \nabla_{\mathbf{w}} \sum_{p:d_p = h_{\mathbf{w}}(\mathbf{x}_p)} \max\{-d_p g_{\mathbf{w}}(\mathbf{x}_p), 0\} + \nabla_{\mathbf{w}} \sum_{p:d_p \neq h_{\mathbf{w}}(\mathbf{x}_p)} \max\{-d_p g_{\mathbf{w}}(\mathbf{x}_p), 0\}$$

$$= 0 + \nabla_{\mathbf{w}} \sum_{p:d_p \neq h_{\mathbf{w}}(\mathbf{x}_p)} -d_p g_{\mathbf{w}}(\mathbf{x}_p)$$

$$= \sum_{p:d_p \neq h_{\mathbf{w}}(\mathbf{x}_p)} -d_p \nabla_{\mathbf{w}} g_{\mathbf{w}}(\mathbf{x}_p)$$

$$= \sum_{p:d_p \neq h_{\mathbf{w}}(\mathbf{x}_p)} -d_p \nabla_{\mathbf{w}} (\mathbf{w} \cdot \mathbf{x}_p)$$

$$= \sum_{p:d_p \neq h_{\mathbf{w}}(\mathbf{x}_p)} -d_p \mathbf{x}_p$$



Minimizing $E_{soft}(\mathbf{w})$ using (sub) gradient descent

$$\nabla_{\mathbf{w}} E_{soft} = \sum_{p:d_p \neq h_{\mathbf{w}}(\mathbf{x}_p)} -d_p\, \mathbf{x}_p$$

$y_p = h_{\mathbf{w}}(\mathbf{x}_p)$

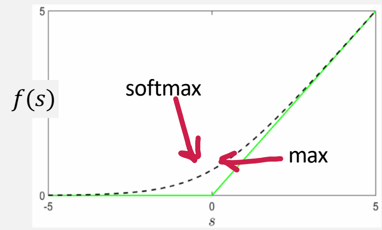
$$\mathbf{w} \leftarrow \mathbf{w} - \eta\, \nabla_{\mathbf{w}} E_{soft}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_{p:d_p \neq h_{\mathbf{w}}(\mathbf{x}_p)} d_p\, \mathbf{x}_p$$

- We add a fraction of $\mathbf{x}_p$ if the desired label is $+1$ and the predicted label is -1
- We subtract a fraction of $\mathbf{x}_p$ if the desired label is -1 and the predicted label is $+1$
- The key difference from the perceptron algorithm is that because we perform gradient descent, we minimize the loss (error) over the training data even if the classes are not linearly separable!

Remarks on the E_{soft} loss function

- $E_{soft}(\mathbf{w}) = \sum_p \max\{-d_p g_{\mathbf{w}}(\mathbf{x}_p), 0\}$ has a trivial minimum at $\mathbf{w}$ =0 that we must take steps in our code to avoid
- We can minimize E_{soft} using only first order (sub)gradient descent (higher order derivatives do not exist)
- Can we approximate E_{soft} by a smooth loss function, say E_{smooth} so we can use a broader range of optimization methods, including higher order methods?
  - Yes
  - By replacing max by softmax



Approximating max by softmax

Suppose $\max\{a, b\} = a$

Recall that $\log e^x = x$

$\max\{a, b\} = b + (a - b) = \log e^b + \log e^{a-b}$

Let $\text{softmax}\,\{a, b\} = \log\left(e^a + e^b\right)$

Note that $\log e^b + \log\left(1 + e^{a-b}\right) = \log\left(e^b\left(1 + e^{a-b}\right)\right)$

$= \log\left(e^a + e^b\right) = \text{softmax}\,\{a, b\}$

$\text{softmax}\,\{a, b\} - \max\{a, b\}$

$= \log e^b + \log\left(1 + e^{a-b}\right) - \log e^b - \log e^{a-b}$

$= \log\left(1 + e^{a-b}\right) - \log e^{a-b}$

$= \log \frac{\left(1+e^{a-b}\right)}{e^{a-b}} = 1 + \frac{1}{e^{a-b}} \approx 1$ especially when $e^{a-b} \gg 1$

E_{soft} to E_{smooth} via softmax

- $E_{soft}(\mathbf{w}) = \sum_p \max\{-d_p g_{\mathbf{w}}(\mathbf{x}_p), 0\}$

Approximating max by softmax, we have:

- $E_{smooth}(\mathbf{w}) = \sum_p \log \left(e^0 + e^{-d_p \mathbf{w}\cdot\mathbf{x}_p}\right)$
  $= \sum_p \log \left(1 + e^{-d_p \mathbf{w}\cdot\mathbf{x}_p}\right)$
- E_{smooth}
  - Is convex and infinitely differentiable, hence we can use higher order optimization methods
  - Does not have a trivial minimum at $\mathbf{w} = 0$
- Empirically, we find that only the first few iterations improve E_{smooth} before the magnitude of the weights starts to increase and become very large

E_{soft} to E_{smooth} via softmax

- $E_{smooth}(\mathbf{w}) = \sum_p \log \left(1 + e^{-d_p \mathbf{w} \cdot \mathbf{x}_p}\right)$
- Empirically, we find that only the first several iterations improve E_{smooth} before the magnitude of the weights starts to become very large
- Solution: regularization – limit the magnitude of weights from increasing without bounds
- $E^R_{smooth}(w_0, \boldsymbol{\omega}) = \sum_p \log \left(1 + e^{-d_p \boldsymbol{\omega} \cdot \mathbf{x}_p - d_p w_0}\right) + \lambda \|\boldsymbol{\omega}\|^2$ where $\mathbf{w} = [w_0\ w_1 \cdots w_N]^T,\ \boldsymbol{\omega} = [w_1 \cdots w_N]^T$

λ is set to a small value, e.g., 0.0001 and prevents the weights from increasing without bounds

Alternatively, λ can be optimized using cross-validation

We will study regularization in greater detail later

E_{soft} to E_{smooth} via softmax

$E^R_{smooth}(w_0, \boldsymbol{\omega}) = \sum_p \log \left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right) + \lambda\|\boldsymbol{\omega}\|^2$ where if

$$\mathbf{w} = [w_0\, w_1 \cdots w_N]^T, \qquad \boldsymbol{\omega} = [w_1 \cdots w_N]^T$$

Gradient based update:

$$\nabla_{\boldsymbol{\omega}} E^R_{smooth}(w_0, \boldsymbol{\omega}) = \nabla_{\boldsymbol{\omega}}\left(\sum_p \log \left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right) + \lambda\|\boldsymbol{\omega}\|^2\right)$$

$$= \sum_p \frac{1}{\left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right)} \nabla_{\boldsymbol{\omega}}\left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right) + \nabla_{\boldsymbol{\omega}}\left(\lambda\|\boldsymbol{\omega}\|^2\right)$$

$$= -\sum_p \frac{e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}}{\left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right)} d_p \mathbf{x}_p + 2\lambda\boldsymbol{\omega}$$

Weight update

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta\, \nabla_{\boldsymbol{\omega}} E^R_{smooth}(w_0, \boldsymbol{\omega})$$

E_{soft} to E_{smooth} via softmax

$E^R_{smooth}(w_0, \boldsymbol{\omega}) = \sum_p \log \left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right) + \lambda\|\boldsymbol{\omega}\|^2$ where $\mathbf{w} = [w_0\, w_1 \cdots w_N]^T, \; \boldsymbol{\omega} = [w_1 \cdots w_N]^T$

Gradient based update:

$\nabla_{w_0} E^R_{smooth}(w_0, \boldsymbol{\omega}) = \nabla_{w_0} \left(\sum_p \log \left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right) + \lambda\|\boldsymbol{\omega}\|^2\right)$

$= \sum_p \frac{1}{\left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right)} \nabla_{w_0}\left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right) + 0$

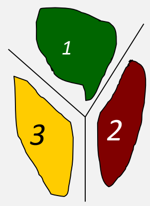
$= -\sum_p \frac{e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}}{\left(1 + e^{-d_p \boldsymbol{\omega}\cdot\mathbf{x}_p - d_p w_0}\right)} d_p$

Weight update

$w_0 \leftarrow w_0 - \eta\, \nabla_{\boldsymbol{\omega}} E^R_{smooth}(w_0, \boldsymbol{\omega})$

Multi-class extension

Predicted class label for $\mathbf{x}_p$ is given by $y_p = argmax_c\ \mathbf{w}_c \cdot \mathbf{x}_p$



Decision surface between class k and j is given by $(\mathbf{w}_k - \mathbf{w}_j) \cdot \mathbf{x} = 0$

Predicted class label for $\mathbf{x}_p$ is d_p

Suppose we define E_p, the error on sample $\mathbf{x}_p$

$$E_p(\mathbf{w}_1, \cdots \mathbf{w}_C) = \max_{c=1,\cdots,C;\&\, c \neq d_p} \left\{0,\ \mathbf{x}_p \cdot \left(\mathbf{w}_c - \mathbf{w}_{d_p}\right)\right\}$$

Error on the training set

$$E = \sum_p E_p(\mathbf{w}_1, \cdots \mathbf{w}_C)$$

Multi-class softmax

Suppose we define E_p, the error on sample $\mathbf{x}_p$

$$E_p(\mathbf{w}_1, \cdots \mathbf{w}_C) = \max_{c \neq d_p} \mathbf{x}_p \cdot \mathbf{w}_c - \mathbf{w}_{d_p} \cdot \mathbf{x}_p$$

$$\approx \log\left(\sum_{c \neq d_p}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_c - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}\right)$$

$$E = \sum_p E_p(\mathbf{w}_1, \cdots \mathbf{w}_C)$$

$$\nabla_{\mathbf{w}_c} E_p = \nabla_{\mathbf{w}_c}\left(\log\left(\sum_{c \neq d_p}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}\right)\right)$$

$$= \left(\frac{1}{\sum_{c \neq d_p}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}}\right) \nabla_{\mathbf{w}_c}\left(\log\left(\sum_{j \neq c \neq d_p} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p} + \right)\right)$$

$$= \left(\frac{1}{\sum_{c \neq d_p}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}}\right)\left(0 + \left(e^{\mathbf{x}_p \cdot \mathbf{w}_c - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}\right) \quad \nabla_{\mathbf{w}_c}\left(\mathbf{x}_p \cdot \mathbf{w}_c\right)\right)$$

$$= \left(\frac{\left(e^{\mathbf{x}_p \cdot \mathbf{w}_c - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}\right)}{\sum_{c \neq d_p}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}}\right) \mathbf{x}_p \text{ (assuming } c \neq d_p\text{)}$$

$$\mathbf{w}_c \leftarrow \mathbf{w}_c - \eta \sum_p \left(\frac{\left(e^{\mathbf{x}_p \cdot \mathbf{w}_c - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}\right)}{\sum_{j=1}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}}\right) \mathbf{x}_p$$

Multi-class softmax

Suppose we define E_p, the error on sample $\mathbf{x}_p$

$E_p(\mathbf{w}_1, \cdots \mathbf{w}_C) \approx$

$E = \sum_p E_p(\mathbf{w}_1, \cdots \mathbf{w}_C)$

$$\nabla_{\mathbf{w}_{d_p}} E_p = \nabla_{\mathbf{w}_{d_p}} \left(\log \left(\sum_{j=1}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p} \right) \right)$$

$= \left(\frac{1}{\sum_{j=1}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}} \right) \nabla_{\mathbf{w}_{d_p}} \left(\sum_{j=1}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p} \right)$

$$= \left(\frac{1}{\sum_{j=1}^{C} e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p}} \right) \left(\sum_{j=1}^{C} \nabla_{\mathbf{w}_{d_p}} \left(e^{\mathbf{x}_p \cdot \mathbf{w}_j - \mathbf{w}_{d_p} \cdot \mathbf{x}_p} \right) \right)$$

$= -\sum_{j=1}^{C} \mathbf{x}_p$ (assuming $\mathbf{w}_j \neq \mathbf{w}_{d_p}$)

Multi-class extension

- The softmax based loss function for multi-class perceptron needs to be regularized for the same reason its 2-class counterpart needs to be regularized

The Perceptron Algorithms Revisited

The perceptron learns by adding misclassified positive or subtracting misclassified negative examples to an arbitrary weight vector, which (without loss of generality) we assumed to be the zero vector. So the final weight vector is a linear combination of the training samples

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i,$$

where, since the sign of the coefficient of $\mathbf{x}_i$ is given by label y_i, the α_i are positive values, proportional to the number of times, misclassification of $\mathbf{x}_i$ has caused the weight to be updated. It is called the embedding strength of the sample $\mathbf{x}_i$.

Dual Representation

The decision function can be rewritten as:

$$h(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle)$$
$$= \operatorname{sgn}(\langle \textstyle\sum_{j} \alpha_j y_j \mathbf{x}_j , \mathbf{x} \rangle)$$
$$= \operatorname{sgn}(\textstyle\sum_{j} \alpha_j y_j \langle \mathbf{x}_j , \mathbf{x} \rangle)$$

The update rule is
if:

$$y_i(\textstyle\sum_{j} \alpha_j y_j \langle \mathbf{x}_j , \mathbf{x}_i \rangle) \leq 0$$

Then

$$\alpha_j \leftarrow \alpha_j + \eta$$

WLOG, we can take $\eta = 1$

Capabilities and limitations of a perceptron

Capabilities

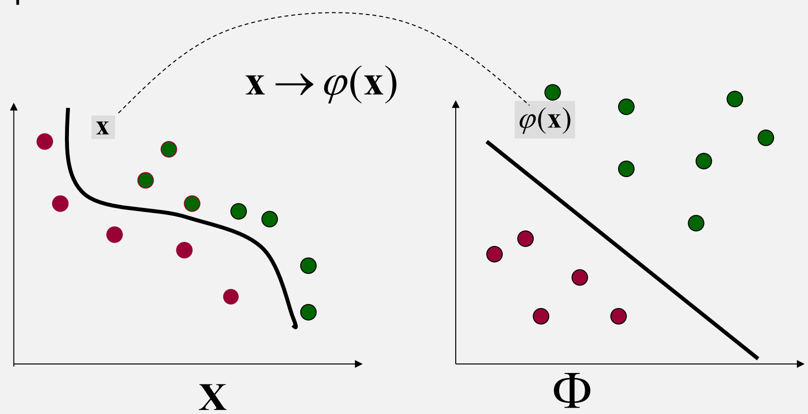
- Perceptron can represent threshold functions
- Perceptron can learn linear decision boundaries

Limitations

- What if the data are not linearly separable?
  - More complex networks?
  - Non-linear transformations into a feature space where the data become separable?

Extending Linear Classifiers

Map data into a feature space where they are linearly separable

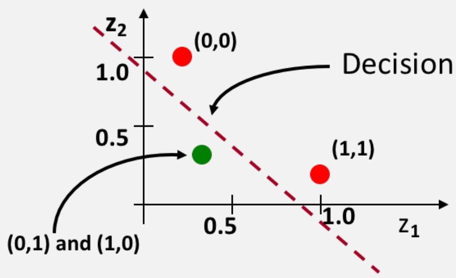


Exclusive OR revisited

In the feature (hidden) space:

$$\varphi_1(x_1, x_2) = e^{-\|\mathbf{X}-\mathbf{W}_1\|^2} = z_1$$
$$\varphi_2(x_1, x_2) = e^{-\|\mathbf{X}-\mathbf{W}_2\|^2} = z_2$$

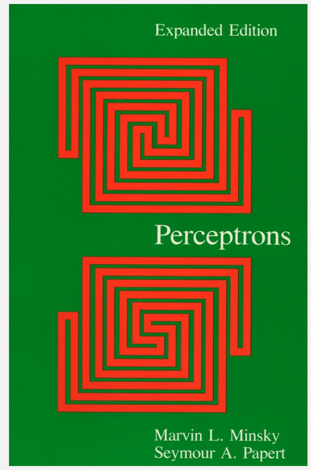
$$\mathbf{W}_1 = [1,1]^T$$
$$\mathbf{W}_2 = [0,0]^T$$



When mapped into the feature space < z_1 , z_2 >, C1 and C2 become *linearly separable.* So a linear classifier with φ_1(x) and φ_2(x) as inputs can be used to solve the XOR problem.

"Perceptrons" (1969)

"The perceptron […] has many features that attract attention: its linearity, its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. *There is no reason to suppose that any of these virtues carry over to the many-layered version*. Nevertheless, *we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension is sterile*."
[pp. 231 – 232]



Postscript

- Minsky and Papert's book had a chilling effect on machine learning research in the US for the next 25 years
  - A few die-hards continued to work on machine learning
  - Artificial Intelligence research shifted to knowledge-based systems
  - Some success with human-engineered knowledge bases
  - Knowledge engineering bottleneck encountered (1980's)
  - Renewed interest in machine learning (mid-late 1980's)
  - Practical approaches to training multi-layer neural networks (late 1980s)
  - Data and computing revolution (1990s – 2000s)
  - Machine learning takes over Artificial Intelligence (2010 – present)