Linear Models for Classification
Discriminant Functions

Karan Nathwani

Topics

- Linear Discriminant Functions
  - Definition (2-class), Geometry
  - Generalization to $K > 2$ classes
  - Distributed representation
- Methods to learn parameters
  1. Least Squares Classification
  2. Fisher's Linear Discriminant
  3. The Perceptron Algorithm

Discriminant Functions

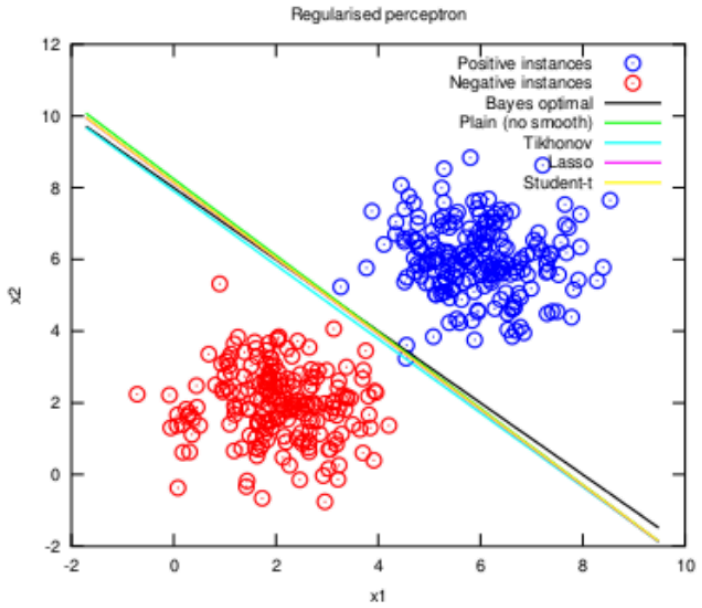
- A discriminant function assigns input vector $\boldsymbol{x}=[x_1,..,x_D]^{\mathrm{T}}$ to one of K classes denoted by C_k
- We restrict attention to linear discriminants
  - i.e., Decision surfaces are hyperplanes
  - Each surface represented by equation

$$y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\,\boldsymbol{x} + w_0$$

$$\boldsymbol{w}=[w_1,..,w_D]^{\mathrm{T}}$$

- First consider $K = 2$, and then extend to $K > 2$

Linear discriminant examples

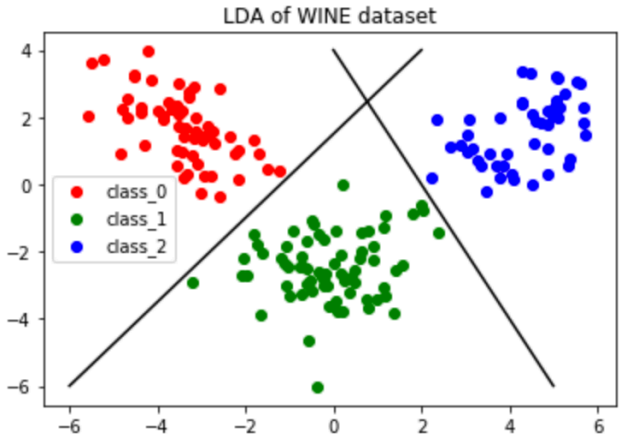


Regularization methods

$$Tikhonov: \quad \lambda \sum_{\forall i} w_i^2$$

$$Lasso: \quad \lambda \sum_{\forall i} |w_i|$$

$$Student-t: \quad \lambda \sum_{\forall i} \log(1+w_i^2)$$



What does value of $y(\boldsymbol{x})$ tell you?

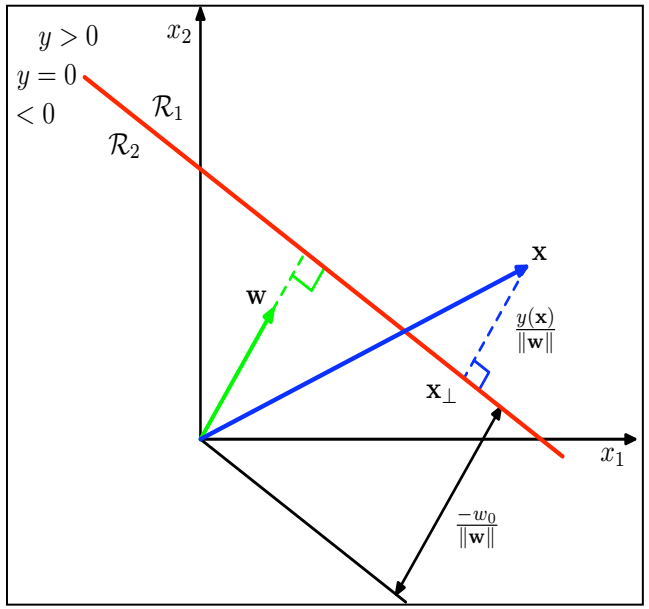
- 2-class case:

$$y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\,\boldsymbol{x} + w_0$$

A linear function of input vector

$\boldsymbol{w}$ is weight vector and w_0 is *bias*

Negative of bias sometimes called *threshold*



- Three cases: $y > 0, y = 0, y < 0$
  - Assign $\boldsymbol{x}$ to C_1 if $y(\boldsymbol{x}) \geq 0$ else C_2
- Defines decision boundary $y(\boldsymbol{x}) = 0$
  - It corresponds to a $(D\text{-}1)$- dimensional hyperplane in a D-dimensional input space

Distance of Origin to Surface is w_0

Let $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ be points on surface $y(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}+w_0=0$

- Because $y(\boldsymbol{x}_A)=y(\boldsymbol{x}_B)=0$, we have $\boldsymbol{w}^T(\boldsymbol{x}_A-\boldsymbol{x}_B)=0$,
- Thus $\boldsymbol{w}$ is orthogonal to every vector lying on decision surface
- So $\boldsymbol{w}$ determines orientation of the decision surface

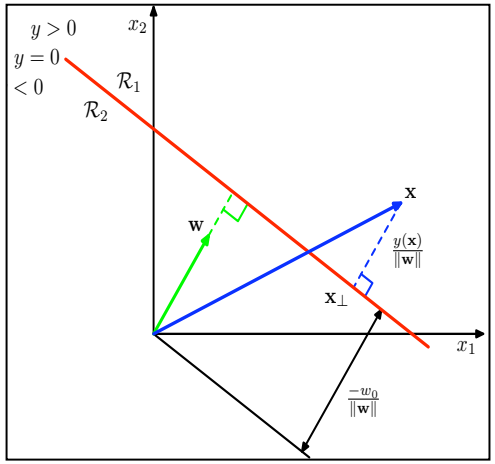
– If $\boldsymbol{x}$ is a point on surface then $y(\boldsymbol{x})=0$ or $\boldsymbol{w}^T\boldsymbol{x}=-w_0$

- Normalized distance from origin to surface:

$$\frac{\mathrm{w}^T\mathrm{x}}{\|\mathrm{w}\|}=-\frac{w_0}{\|\mathrm{w}\|}$$

where $\|\mathrm{w}\|$ is the norm defined as

$$\|\mathrm{w}\|^2=\mathrm{w}^T\mathrm{w}=w_1^2+..+w_{M-1}^2$$



– Where elements of $\boldsymbol{w}$ are normalized by dividing by its norm $\|\boldsymbol{x}\|$

» By definition of a normalized vector which has length 1

– w_0 sets distance of origin to surface

Distance of $\boldsymbol{x}$ to surface is $\boldsymbol{y(x)/||w||}$

Let $\boldsymbol{x}$ be an arbitrary point

– We can show that $y(\boldsymbol{x})$ gives signed measure of perpendicular distance r from $\boldsymbol{x}$ to surface as follows:

- If $\boldsymbol{x}_{\mathrm{p}}$ is orthogonal projection of $\boldsymbol{x}$ to surface then

$$x = x_p + r\frac{\mathrm{w}}{||\,\mathrm{w}\,||} \text{ by vector addition}$$

Second term is a vector normal to surface.

This vector is parallel to w

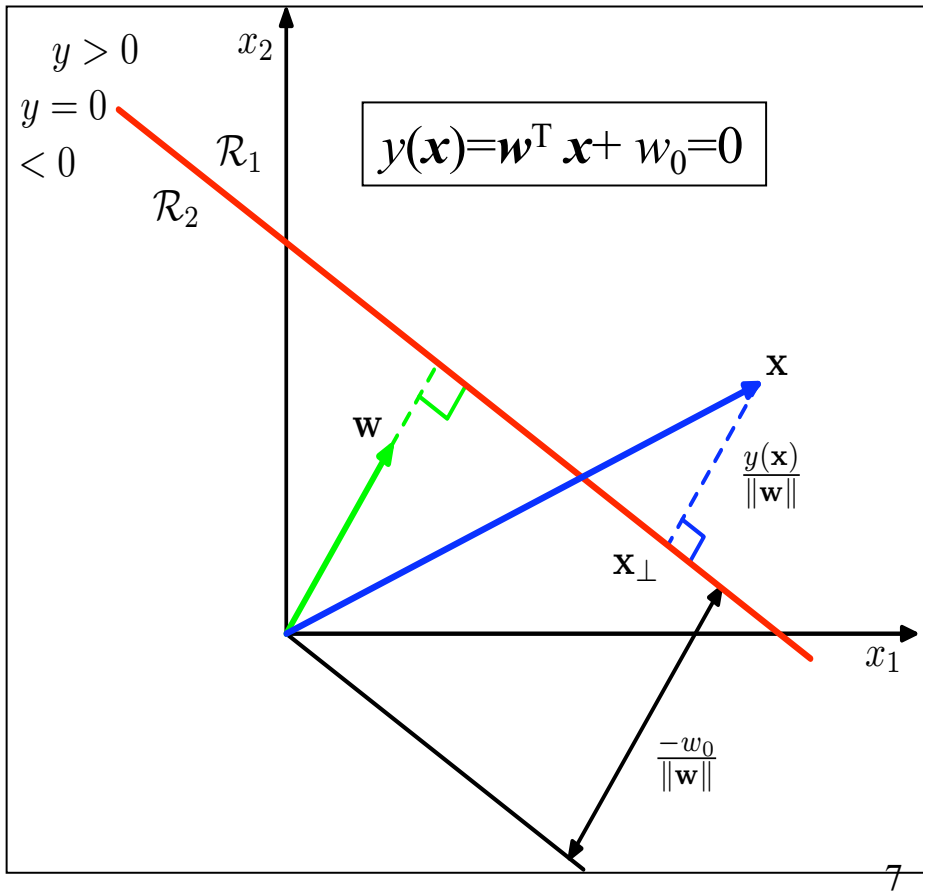
which is normalized by length ||w|| .

Since a normalized vector has length 1

we need to scale by r.

From which we can get

$$r = \frac{y(\mathrm{x})}{||\,\mathrm{w}\,||}$$



Augmented vector

- We have $y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\,\boldsymbol{x} + w_0$

  where $\boldsymbol{x}=[x_1,..,x_D]^{\mathrm{T}}$ $\boldsymbol{w}=[w_1,..,w_D]^{\mathrm{T}}$

- With dummy input $x_0=1$
  - We have $\tilde{\boldsymbol{w}} = (w_0, \boldsymbol{w})$ $\tilde{\boldsymbol{x}} = [1, \boldsymbol{x}]$ and $y(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}$
  - It passes through origin in *augmented* $D+1$ dimensional space
    - Since the bias term is zero in this space
  - Example:
    - Data point $\boldsymbol{x} = [0.2, 0.3, 0.4, 0.5]$ is augmented to [1.0, 0.2, 0.3, 0.4, 0.5]
- We can discard bias term with augmentation

Extension to Multiple Classes

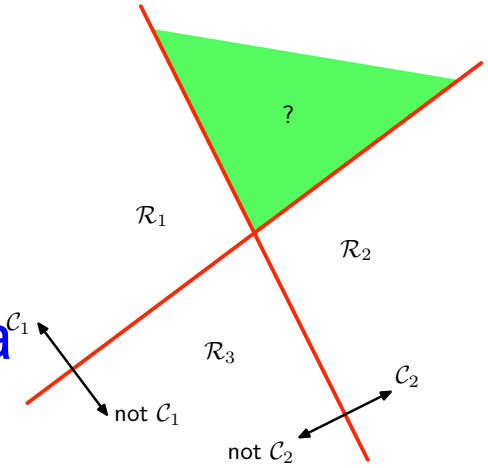
- Two approaches:
  - Using several two-class classifiers
    - But leads to serious difficulties
  - Use K linear functions

Multiple Classes with 2-class classifiers

- By using several 2-class classifiers

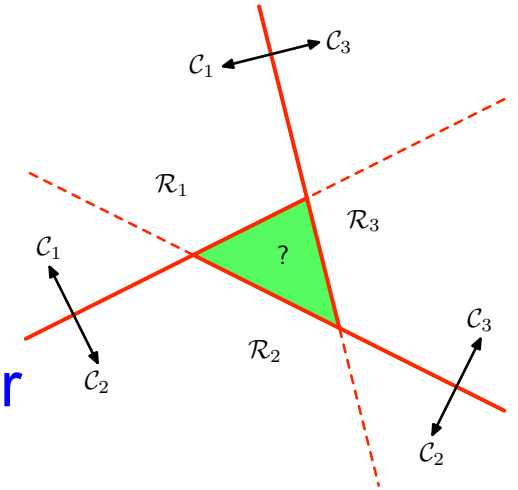
One-versus-the-rest

Build a K class discriminant
Use $K-1$ classifiers, each solve a two-class problem



One-versus-one

Alternative is $K(K-1)/2$ binary discriminant functions, one for every pair



Both result in ambiguous regions of input space

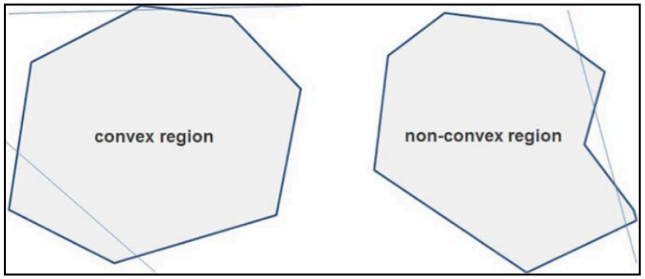
Multiple Classes with K discriminants

- Consider a single K class discriminant of the form

$$y_k(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}{}_k\, \boldsymbol{x} + w_{k0}$$

- Assign a point $\boldsymbol{x}$ to class C_k if $y_k(\boldsymbol{x}) > y_j(\boldsymbol{x})$ for all $j \neq k$

- Decision boundary between C_k and C_j is given by $y_k(\boldsymbol{x}) = y_j(\boldsymbol{x})$
  - This corresponds to $D-1$ dimensional hyperplane defined by
  - $(\boldsymbol{w}_{\mathrm{k}} - \boldsymbol{w}_{\mathrm{j}})^{\mathrm{T}}\, \boldsymbol{x} + (w_{k0} - w_{j0}) = 0$
  - Same form as the decision boundary for 2-class case $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 = 0$
- Decision regions of such discriminants are always singly connected and convex
  - Proof follows



Convexity of Decision Regions (Proof)

Consider two points $\boldsymbol{x}_\mathrm{A}$ and $\boldsymbol{x}_\mathrm{B}$ both in decision region R_k

Any point $\hat{\boldsymbol{x}}$ on line connecting $\boldsymbol{x}_\mathrm{A}$ and $\boldsymbol{x}_\mathrm{B}$ can be expressed as

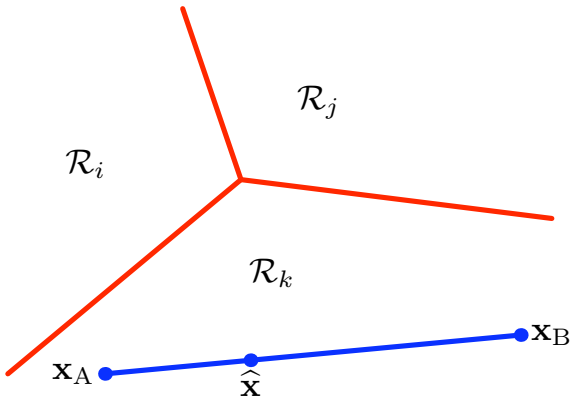
$$\hat{\boldsymbol{x}} = \lambda \boldsymbol{x}_\mathrm{A} + (1-\lambda)\boldsymbol{x}_B \quad \text{where } 0 \le \lambda \le 1$$

From linearity of discriminant functions

$$y_k(\boldsymbol{x}) = \boldsymbol{w}^\mathrm{T}{}_k\, \boldsymbol{x} + w_{k0}$$

Combining the two, we have

$$\hat{\mathrm{x}} = \lambda \mathrm{x}_\mathrm{A} + (1-\lambda)\mathrm{x}_B$$



Because $\boldsymbol{x}_\mathrm{A}$ and $\boldsymbol{x}_\mathrm{B}$ lie inside R_k it follows that

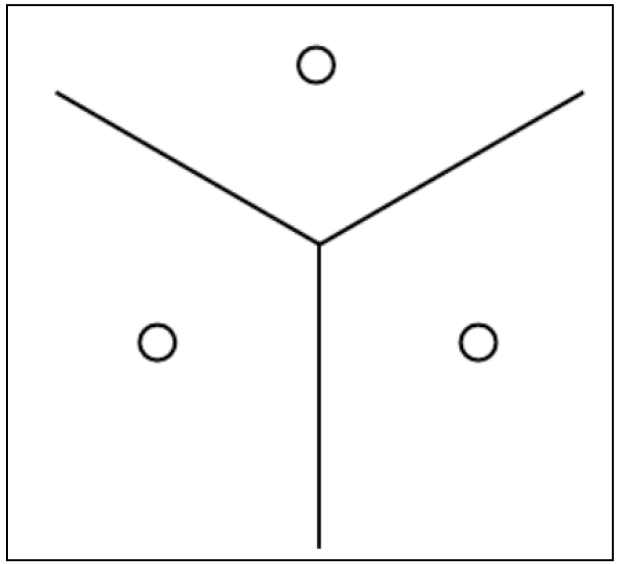
$y_k(\boldsymbol{x}_\mathrm{A}) > y_j(\boldsymbol{x}_\mathrm{A})$ and $y_k(\boldsymbol{x}_\mathrm{B}) > y_j(\boldsymbol{x}_\mathrm{B})$ for all $j \neq k$

Hence $\hat{\mathbf{x}}$ also lies inside R_k

Thus R_k is singly-connected and convex
(single straight line connects any two points in region)

No. of examples and no. of regions

- K discriminants need $O(K)$ examples:
- Nearest-neighbor : each training sample (circle) defines at most one region
  - y value associated with each example defines the output for all points within region

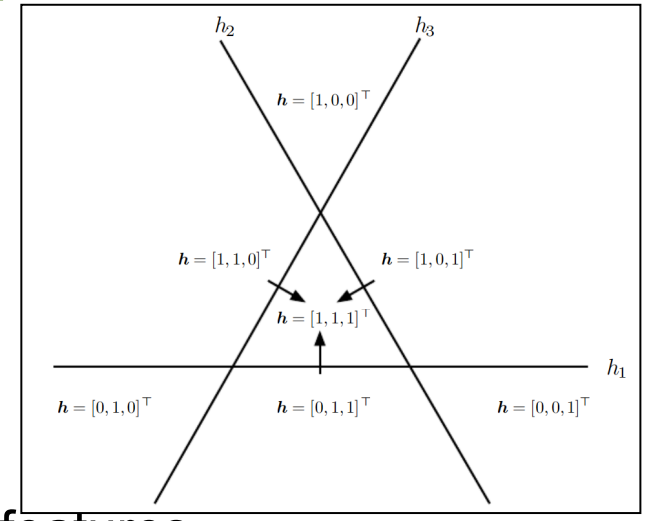


More regions than examples

- Suppose we need more regions than examples
- Two questions of interest
  1. Is it possible represent a complicated function efficiently?
  2. Is it possible for the estimated function to generalize well for new inputs?
- Answer to both is yes
  - $O(2^K)$ regions can be defined with $O(K)$ examples
    - By introducing dependencies between regions through assumptions on data generating distribution

Core idea of deep learning

- Assume data was generated by composition of factors, at multiple levels in a hierarchy
  - Many other similarly generic assumptions
- These mild assumptions allow exponential gain in no of samples and no of regions
  - An example of a distributed representation is a vector of n binary features
    - It can take 2^n configurations
      - Whereas in a symbolic representation, each input is associated with a single symbol (or category)
      - Here h_1, h_2 and h_3 are three binary features



Learning the Parameters of Linear Discriminant Functions

- Three Methods
  - Least Squares
  - Fisher's Linear Discriminant
  - Perceptrons
- Each is simple but several disadvantages

Least Squares for Classification

- Analogous to regression: simple closed- form solution exists for parameters
- Each $C_k, k = 1,..K$ is described by its own linear model

$$y_k(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}{}_k \boldsymbol{x} + w_{k0}$$

Note: $\boldsymbol{x}$ and $\boldsymbol{w}$ have D dimensions each

- Create augmented vector
  - replace $\boldsymbol{x}$ by $(1, \boldsymbol{x}^{\mathrm{T}})$ and $\boldsymbol{w}_k$ by $(w_{k0}, \boldsymbol{w}_k^{\mathrm{T}})$

Now $\boldsymbol{x}$ and $\boldsymbol{w}$ are (D+1) × 1

- Grouping y_ks into a $K \times 1$ vector: $\boldsymbol{y}(\boldsymbol{x}) = \mathbf{W}^{\mathrm{T}} \boldsymbol{x}$

  $\mathbf{W}^{\mathbf{T}}$ is the *parameter matrix* whose k^{th} column is a $D+1$ dimensional vector $\boldsymbol{w}_k$ (including bias). It is $K \times (D+1)$.
- New input vector $\boldsymbol{x}$ is assigned to class for which output $y_k = \boldsymbol{w}^{\mathrm{T}}{}_k \boldsymbol{x}$ is largest
- Determine $\mathbf{W}$ by minimizing squared error

Parameters using Least Squares

- Training data $\{\boldsymbol{x}_n, \mathbf{t}_n\}, n = 1, \ldots, N$

  $\mathbf{t}_n$ is a column vector of K dimensions using 1-of–K form

- Define matrices

  $\mathrm{T} \equiv n^{\text{th}}$ row is the vector $\mathbf{t}^{\mathrm{T}}{}_n$ This is $N \times K$

  $\mathbf{X} \equiv n^{\text{th}}$ row of which is $\boldsymbol{x}_n{}^{\mathrm{T}}$

  This is the $N \times (D+1)$ design matrix

- Sum of squares error function

$$E_{\mathrm{D}}(\mathbf{W}) = \tfrac{1}{2}\ \mathrm{Tr}\ \{(\mathbf{XW} - \mathrm{T})^{\mathrm{T}}\ (\mathbf{XW} - \mathrm{T})\}$$

Notes:

(XW-T) is error vector, whose square is a diagonal matrix

Trace is the sum of diagonal elements

Minimizing Sum of Squares

- Sum of squares error function

$$E_D(\mathbf{W}) = \tfrac{1}{2}\,\mathrm{Tr}\,\{(\mathbf{XW}-\mathrm{T})^{\mathrm{T}}(\mathbf{XW}-\mathrm{T})\}$$

- Set derivative w.r.t. $\mathbf{W}$ to zero, gives solution

$$\mathbf{W} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathrm{T} = \mathbf{X}^{\dagger}\mathrm{T}$$

where $\mathbf{X}^{\dagger}$ is pseudo-inverse of matrix $\mathbf{X}$

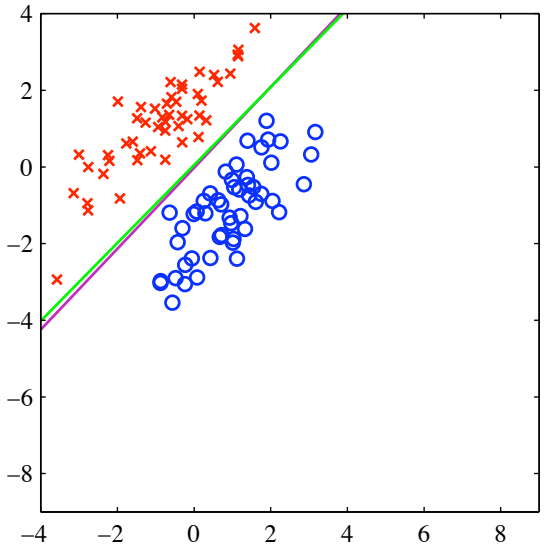
- Discriminant function, after rearranging, is

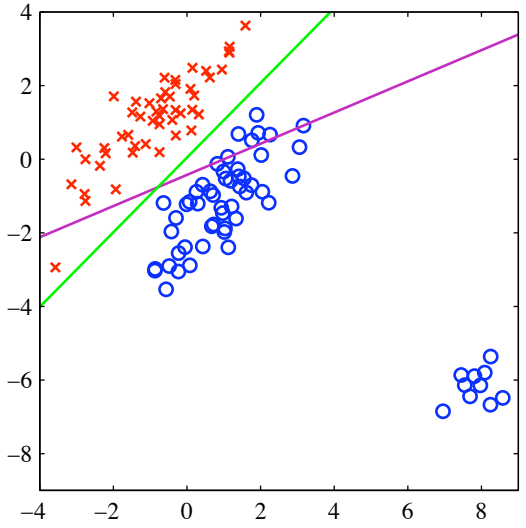
$$y(\boldsymbol{x}) = \mathbf{W}^{\mathrm{T}}\boldsymbol{x} = \mathrm{T}^{\mathrm{T}}(\mathbf{X}^{\dagger})^{\mathrm{T}}\boldsymbol{x}$$

- An exact closed form solution for $\mathbf{W}$ using which we can classify $\boldsymbol{x}$ to class k for which y_k is maximum

  but has severe limitations

Least Squares is Sensitive to Outliers





Magenta: Least Squares

Green: Logistic Regression (more robust)

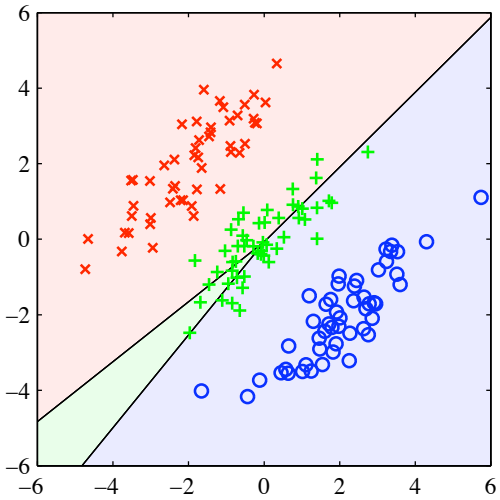
Sum of squared errors penalizes predictions that are “too correct”

Or long way from decision boundary

SVMs have an alternate error function (hinge function)
that does not have this limitation

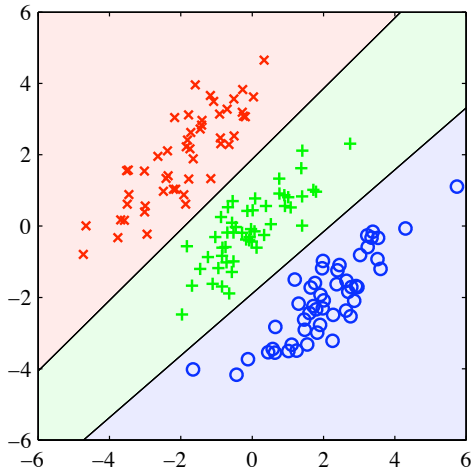
Disadvantages of Least Squares

Least Squares



Three classes
2-D space

Logistic Regression



Region assigned to green class is too small, mostly misclassified
Yet linear decision boundaries of logistic regression can give perfect results

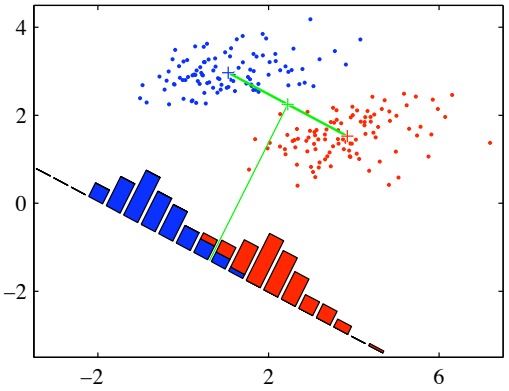
- Lack robustness to outliers
- Certain datasets unsuitable for least squares classification
- Decision boundary corresponds to ML solution
  under Gaussian conditional distribution
- But binary target values have a distribution far from Gaussian

4. Fisher Linear Discriminant

- View classification in terms of dimensionality reduction
  - Project D-dimensional input vector $\boldsymbol{x}$ into one dimension using $y = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$
  - Place threshold on y to classify

    $y \geq -w_0$ as C_1 and otherwise C_2

    we get standard linear classifier



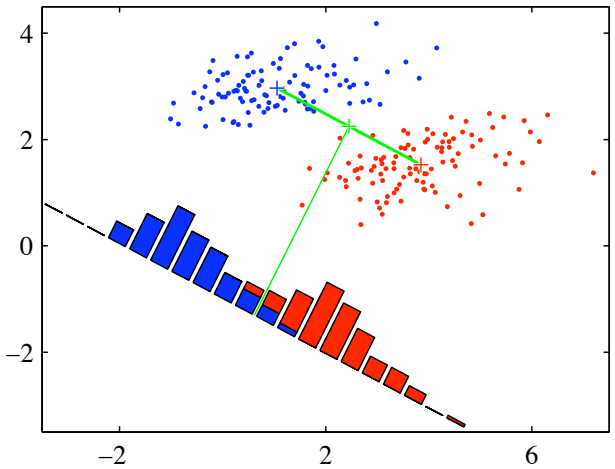
- Classes well-separated in D-space may strongly overlap in 1-dimension
  - Adjust component of the weight vector $\boldsymbol{w}$
  - Select projection to maximize class-separation

Fisher: Maximizing Mean Separation

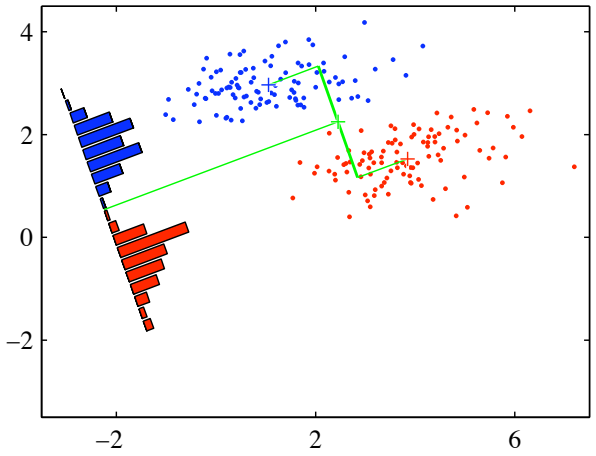
- Two class problem:
  - N_1 points of class C_1 and N_2 points of class C_2
- Mean Vectors $\boxed{\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{n\varepsilon C_1}\boldsymbol{x}_n}$ $\boxed{\boldsymbol{m}_2 = \frac{1}{N_1}\sum_{n\varepsilon C_2}\boldsymbol{x}_n}$
- Choose $\boldsymbol{w}$ to best separate class means
- Maximize $m_2 - m_1 = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{m}_2 - \boldsymbol{m}_1)$,

  where $m_k = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_k$ is the mean of projected data of class C_k
- Can be made arbitrarily large by increasing $\boldsymbol{w}$
  - Introduce Lagrange multiplier to enforce ($\boldsymbol{w}$ to have unit length) $\Sigma_i w_i^2 = 1$
  - There is still a problem with this approach, and Fisher proposed asolution

Fisher: Minimizing Variance



Means are well-separated but classes overlap



Projection based on Fisher showing greatly improved class separation

- Maximizing mean separation is insufficient for classes with non-diagonal covariance
- Fisher formulation
  1. Maximize function to separate projected class means
  2. Also give small variance within each class, thereby minimizing the class overlap

Fisher Criterion and its Optimization

- In 1-dimensional space, within class variance is

$$s_k^2 = \Sigma_{n \varepsilon Ck}(y_n - m_k)^2, \text{ where } y_n = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n$$

  – Total within-class variance is given by $s_1^2 + s_2^2$

- Fisher criterion = $J(\boldsymbol{w}) = (m_2 - m_1)^2 / s_1^2 + s_2^2$

Rewriting $J(\boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}} S_{\mathrm{B}} \boldsymbol{w} \,/\, \boldsymbol{w}^{\mathrm{T}} S_{\mathrm{W}} \boldsymbol{w}$ where S_{B} is the *between class* covariance $S_{\mathrm{B}} = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^{\mathrm{T}}$ and S_W is the *within-class* covariance matrix

$$S_W = \Sigma_{n \in C1}(\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \Sigma_{n \in C2}(\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T$$

- Differentiating wrt $\boldsymbol{w}$, $J(\boldsymbol{w})$ is maximized when

$$(\boldsymbol{w}^{\mathrm{T}} S_{\mathrm{B}} \boldsymbol{w}) S_{\mathrm{W}} \boldsymbol{w} = (\boldsymbol{w}^{\mathrm{T}} S_{\mathrm{W}} \boldsymbol{w}) S_{\mathrm{B}} \boldsymbol{w}$$

Dropping scalar factors (in parentheses) & noting S_{B} is in same direction as $(\boldsymbol{m}_2 - \boldsymbol{m}_1)$ & multiplying by S_{W}^{-1}

$$\boldsymbol{w} \;\alpha\; S_W^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

Relation to Least Squares

- Least Squares: goal of making Model predictions as close as possible to target values
- Fisher: require maximum class separation
- For two-class problem Fisher is special case of least squares
  - Proof starts with sum-of-square errors and shows that weight vector found coincides with Fisher criterion

Fisher's Discriminant for Multiple Classes

- Can be generalized for multiple classes
- Derivation is fairly involved [Fukunaga 1990]

The Perceptron Algorithm

- Another example of a linear discriminant model
  - Others: discriminant functions, Fisher linear discrim
- Occupies an important place in the history of machine learning
- It corresponds to a two-class model
  - In which the input vector $\boldsymbol{x}$ is first transformed to give feature vector $\boldsymbol{\phi}(\boldsymbol{x})$
  - Then used to construct a generalized linear model

$$y(\boldsymbol{x}) = f\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right)$$

Perceptron Model

- Two-class model
  - Input vector $\boldsymbol{x}$ transformed by a fixed nonlinear transformation to give feature vector $\boldsymbol{\phi}(\boldsymbol{x})$

$$y(\boldsymbol{x}) = f\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right)$$

where non-linear activation $f(\bullet)$ is a step function

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

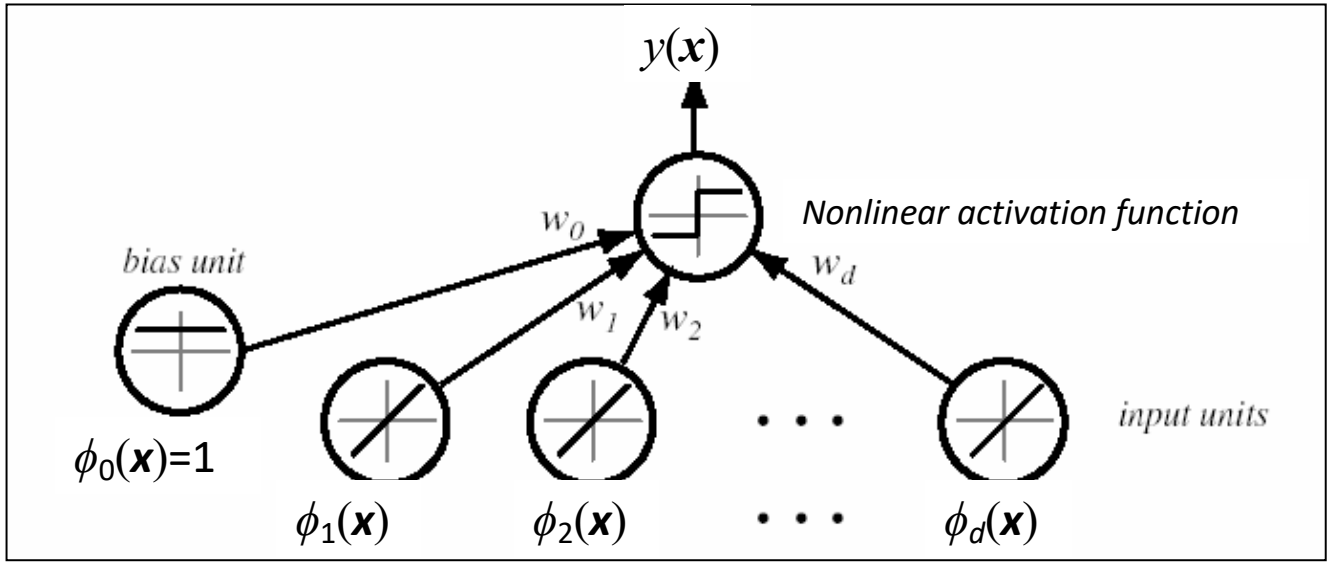
The vector $\boldsymbol{\phi}(\boldsymbol{x})$ includes a bias component $\phi_0(\boldsymbol{x})=1$

Perceptron functional diagram

- $y(\boldsymbol{x}) = f(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}))$ $\quad \boldsymbol{\phi}(\boldsymbol{x}) = [\phi_0(\boldsymbol{x}),.., \phi_d(\boldsymbol{x})]$ $\quad \boldsymbol{w}=[w_0,..,w_d]$

Non-linear activation function definition:

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$



Perceptron Target Coding

- Use a target coding scheme
  - In earlier discussion we focused on $t \in \{0,1\}$
    - Which is appropriate for probabilistic models
  - For perceptrons it is more convenient to use target values
    - $t = +1$, for class C_1 and $t = -1$ for C_2 matching the activation function

Determining parameters $\boldsymbol{w}$

- Perceptron criterion is motivated by error function minimization
- A natural choice is total number of misclassifications, $E(\boldsymbol{w})$
  - This error function is a piecewise constant function of $\boldsymbol{w}$ with discontinuities (unlike regression)
    - A change in $\boldsymbol{w}$ causes the decision boundary to move across one of the data points
  - Methods based on changing $\boldsymbol{w}$ using gradient of error function cannot be applied
    - Gradient is zero almost everywhere
  - Alternative error criterion is the *perceptron criterion*

Perceptron Criterion $E_P(\boldsymbol{w})$

- Seek $\boldsymbol{w}$ such that $\boldsymbol{x}_n \in C_1$ will have $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) \geq 0$ whereas patterns $\boldsymbol{x}_n \in C_2$ will have $\boldsymbol{x} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) < 0$
- Using $t \in \{+1, -1\}$, it follows that all patterns need to satisfy $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)t_n > 0$
  - Perceptron criterion associates zero error with any input correctly classified
  - For each misclassified sample, it tries to minimize $-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)t_n$ or

$$E_P(\boldsymbol{w}) = -\sum_{n \in M} \boldsymbol{w}^T \boldsymbol{\phi}_n t_n$$

M denotes set of all misclassified patterns and $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\boldsymbol{x}_n)$

Property of Perceptron Criterion

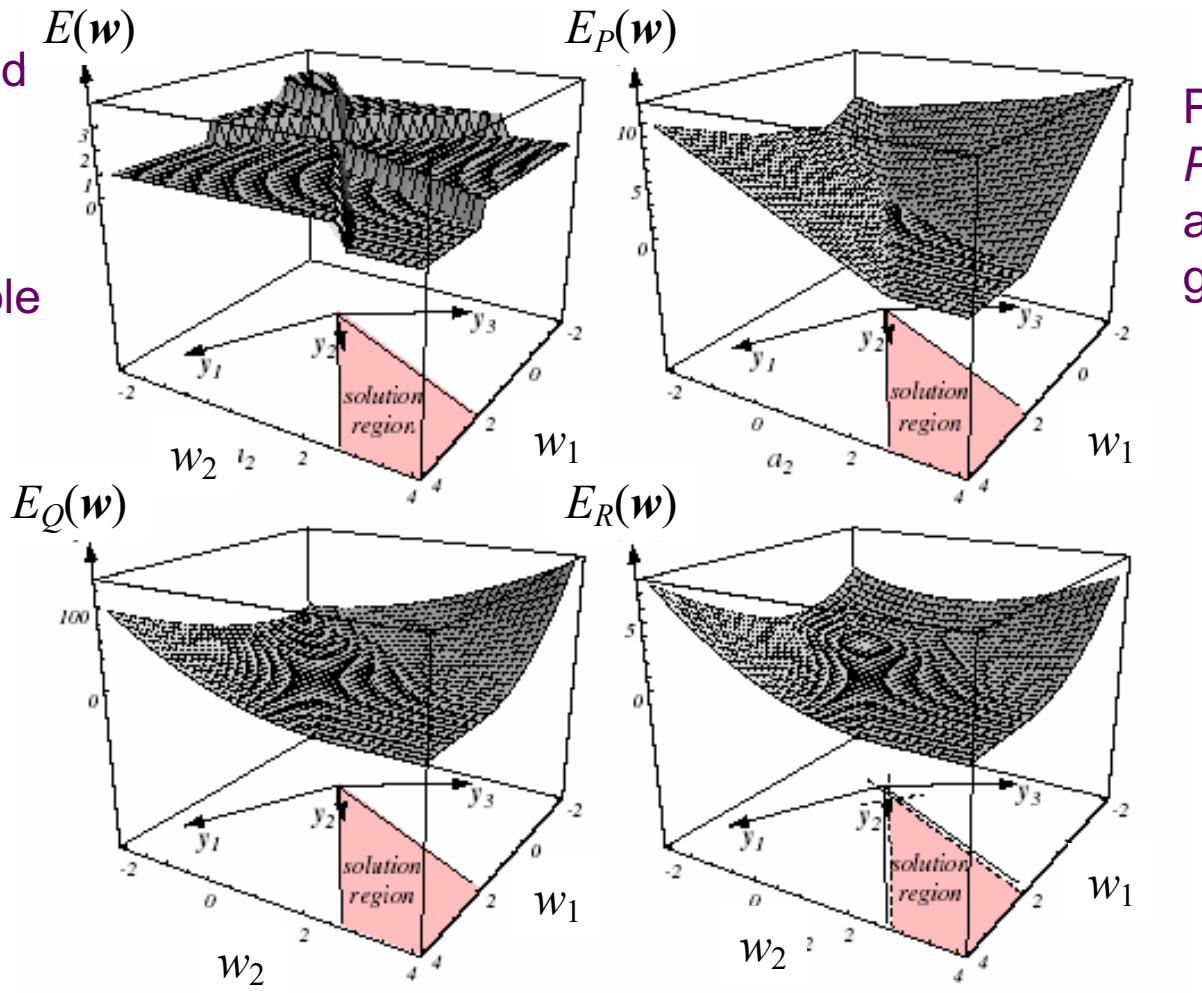
- Contribution of a misclassified sample

  $-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)t_n$ is a linear function of $\boldsymbol{w}$ in regions of $\boldsymbol{w}$ space where it is misclassified and zero in regions where it is correctly classified

- The total error function is therefore piecewise linear

Comparison of Four Criterion functions

No of misclassified samples: *Piecewise constant*, unacceptable



Perceptron criterion: *Piecewise linear*, acceptable for gradient descent

Squared Error with margin

$$E_R(\boldsymbol{w}) = \frac{1}{s}\sum_{n\in M}\frac{(\boldsymbol{w}^T\boldsymbol{\phi}_n - b)^2}{\|\boldsymbol{\phi}_n\|}$$

Squared error: Useful when patterns are not linearly separable

$$E_Q(\boldsymbol{w}) = \sum_{n\in M}\left(\boldsymbol{w}^T\boldsymbol{\phi}_n\right)^2$$

Perceptron Algorithm

- Error function $$E_P(\boldsymbol{w}) = -\sum_{n \in M} \boldsymbol{w}^T \boldsymbol{\phi}_n t_n$$
- Apply Stochastic Gradient Descent to E_P
- Change in weight is given by

$$\boldsymbol{w}^{\tau+1} = \boldsymbol{w}^{\tau} - \eta \nabla E_P(\boldsymbol{w}^{\tau}) = \boldsymbol{w}^{\tau} + \eta \boldsymbol{\phi}_n t_n$$

  η is learning rate, τ indexes the steps

- The algorithm

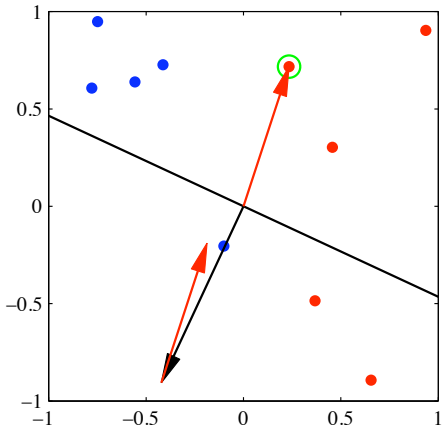
  Cycle through the training patterns in turn

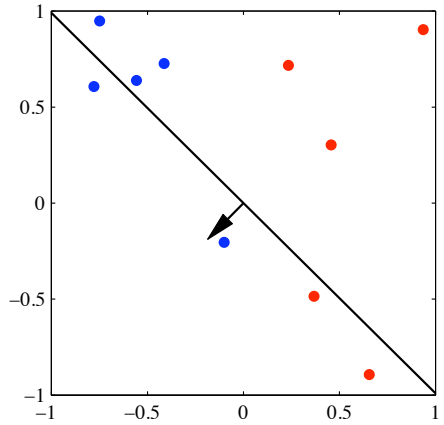
  If incorrectly classified for class C_1 add to weight vector

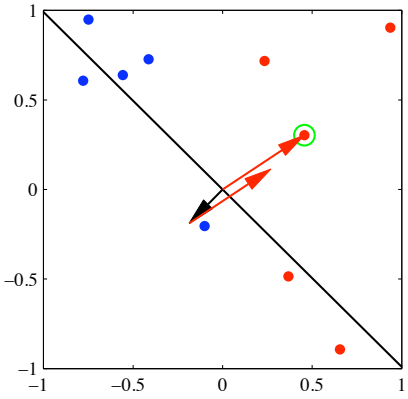
  If incorrectly classified for class C_2 subtract from weight vector

Convergence in feature space $\phi(x)$

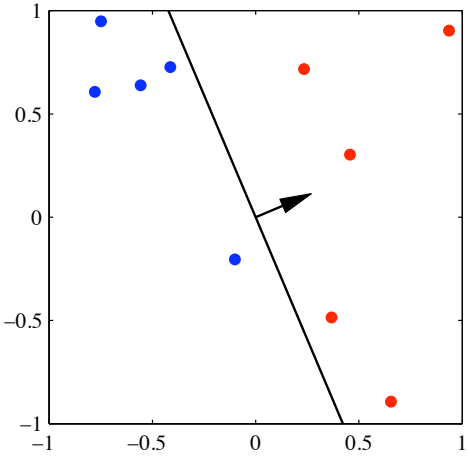
2-d Feature space (ϕ_1, ϕ_2) and Two-classes





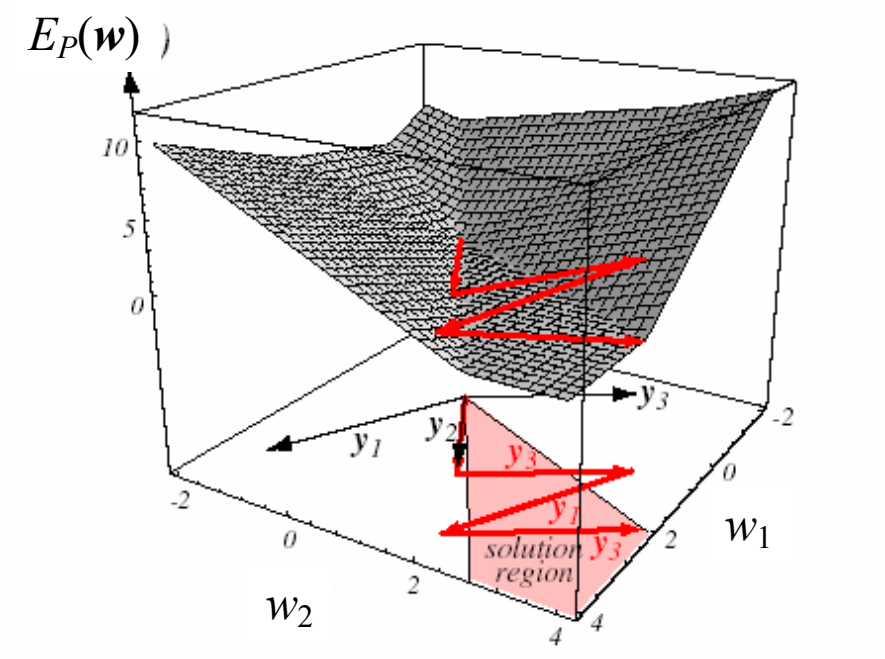


Black arrow: initial parameter vector $\boldsymbol{w}$
Black line: decision boundary
Arrow points to red class
Circled point in green is misclassified
Which is added to current $\boldsymbol{w}$ to give
new decision boundary shown in next plot



Data points
Correctly classified

Perceptron Solution in weight space $\boldsymbol{w}$



Criterion function plotted as a function of weights w_1 and w_2

Three samples.

Weight vector starts at origin.. Sequentially add to it "normalized" misclassified samples themselves

Sequence is y_2, y_3, y_1, y_3

Second update by y_3 takes solution farther from solution region than first update by y_3

Perceptron Convergence

Samples y_1, y_2, y_3 considered cyclically
Misclassified samples are marked

$$\overset{\downarrow}{\mathbf{y}_1},\ \mathbf{y}_2,\ \overset{\downarrow}{\mathbf{y}_3},\ \overset{\downarrow}{\mathbf{y}_1},\ \overset{\downarrow}{\mathbf{y}_2},\ \mathbf{y}_3,\ \mathbf{y}_1,\ \overset{\downarrow}{\mathbf{y}_2},\ \ldots$$

$$\begin{array}{ll} \mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k & k \geq 1 \end{array}$$

- Fixed increment single sample correction

```
1 begin initialize a, k ← 0
2         do k ← (k + 1) mod n
3            if y^k is misclassified by a then a ← a + y^k
4         until all patterns properly classified
5    return a
6 end
```

Perceptron Convergence Theorem

Theorem (Perceptron Convergence)

If training samples are linearly separable, then the sequence of weight vectors given by *Fixed Increment Error Correction Algorithm* will terminate at a solution vector

Proof for Single-Sample Correction

Let $\hat{\mathbf{a}}$ be any solution vector, so that $\hat{\mathbf{a}}^t\mathbf{y}_i$ is strictly positive for all i, and let α be a positive scale factor. From Eq. 20 we have

$$\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}} = (\mathbf{a}(k) - \alpha\hat{\mathbf{a}}) + \mathbf{y}^k$$

and hence

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 = \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 + 2(\mathbf{a}(k) - \alpha\hat{\mathbf{a}})^t\mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

Proof for Single-Sample Correction

Because $\mathbf{y}^k$ was misclassified, $\mathbf{a}^t(k)\mathbf{y}^k \leq 0$ and thus

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\hat{\mathbf{a}}^t\mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

Because $\hat{\mathbf{a}}^t\mathbf{y}^k$ is strictly positive, the second term will dominate the third if α is sufficiently large. In particular, if we let β be the maximum length of a pattern vector,

$$\beta^2 = \max_i \|\mathbf{y}_i\|^2, \tag{21}$$

and let γ be the smallest inner product of the solution vector with any pattern vector, that is,

$$\gamma = \min_i \left[\hat{\mathbf{a}}^t\mathbf{y}_i\right] > 0, \tag{22}$$

Proof for Single-Sample Correction

then we have the inequality

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\gamma + \beta^2.$$

If we choose

$$\alpha = \frac{\beta^2}{\gamma}, \tag{23}$$

we obtain

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - \beta^2.$$

Thus, the squared distance from $\mathbf{a}(k)$ to $\alpha\hat{\mathbf{a}}$ is reduced by at least β^2 at each correction, and after k corrections we obtain

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(1) - \alpha\hat{\mathbf{a}}\|^2 - k\beta^2. \tag{24}$$

Because this squared distance cannot become negative, it follows that the sequence of corrections must terminate after no more than k_0 corrections, where

$$k_0 = \frac{\|\mathbf{a}(1) - \alpha\hat{\mathbf{a}}\|^2}{\beta^2}. \tag{25}$$

Bound on no.of corrections

$$k_0 = \frac{\alpha^2 \|\hat{\mathbf{a}}\|^2}{\beta^2} = \frac{\beta^2 \|\hat{\mathbf{a}}\|^2}{\gamma^2} = \frac{\max_i \|\mathbf{y}_i\|^2 \|\hat{\mathbf{a}}\|^2}{\min_i [\mathbf{y}_i^t \hat{\mathbf{a}}]^2}.$$

Some Direct Generalizations

Correction whenever $a^t(k)\, y^k$ fails to exceed a margin

$$\begin{aligned} &\mathbf{a}(1) && \text{arbitrary} \\ &\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k && k \geq 1, \end{aligned}$$

Algorithm 5. (Variable-Increment Perceptron with Margin)

```
1 begin initialize a, threshold θ, margin b, η(·), k ← 0
2         do k ← (k + 1) mod n
3            if a^t y^k ≤ b then a ← a + η(k)y^k
4         until a^t y^k > b for all k
5    return a
6 end
```

It can be shown that if the samples are linearly separable and if

$$\eta(k) \geq 0, \tag{28}$$

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = \infty \tag{29}$$

and

$$\lim_{m\to\infty} \frac{\sum_{k=1}^{m} \eta^2(k)}{\left(\sum_{k=1}^{m} \eta(k)\right)^2} = 0, \tag{30}$$

then $\mathbf{a}(k)$ converges to a solution vector $\mathbf{a}$ satisfying $\mathbf{a}^t\mathbf{y}_i > b$ for all i (Problem 19).

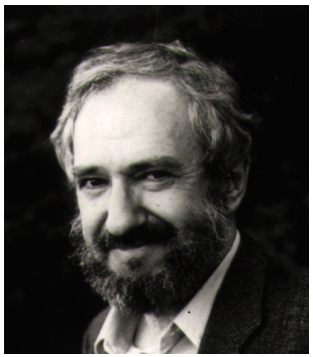
Some Direct Generalizations: original gradient descent algorithm for E_P

$$\mathbf{a}(1) \qquad \text{arbitrary}$$
$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y},$$

Algorithm 6. (Batch Variable Increment Perceptron)

```
1  begin initialize a, η(·), k ← 0
2        do k ← (k + 1) mod n
3           Y_k = {}
4           j = 0
5           do j ← j + 1
6              if y_j is misclassified then Append y_j to Y_k
7           until j = n
8           a ← a + η(k) Σ_{y∈Y_k} y
9        until Y_k = {}
10    return a
11 end
```

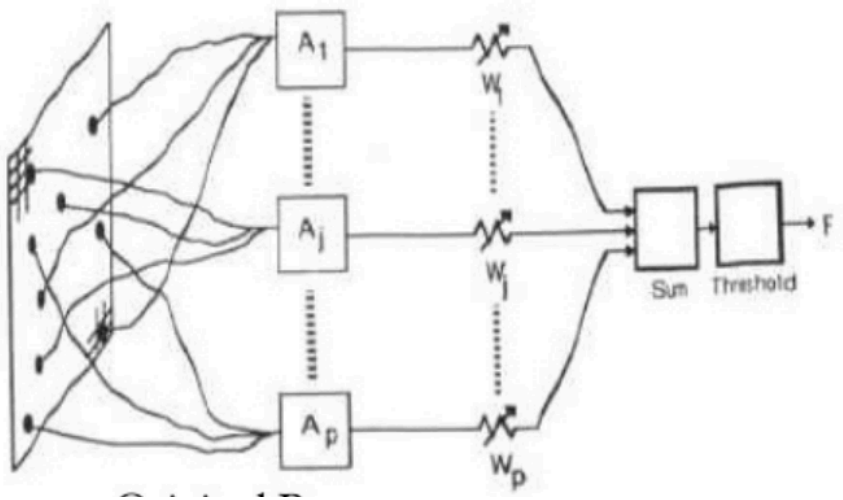
History of Perceptrons

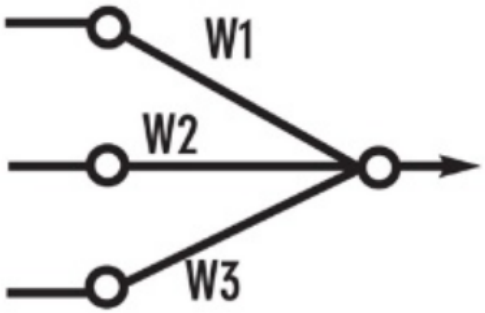


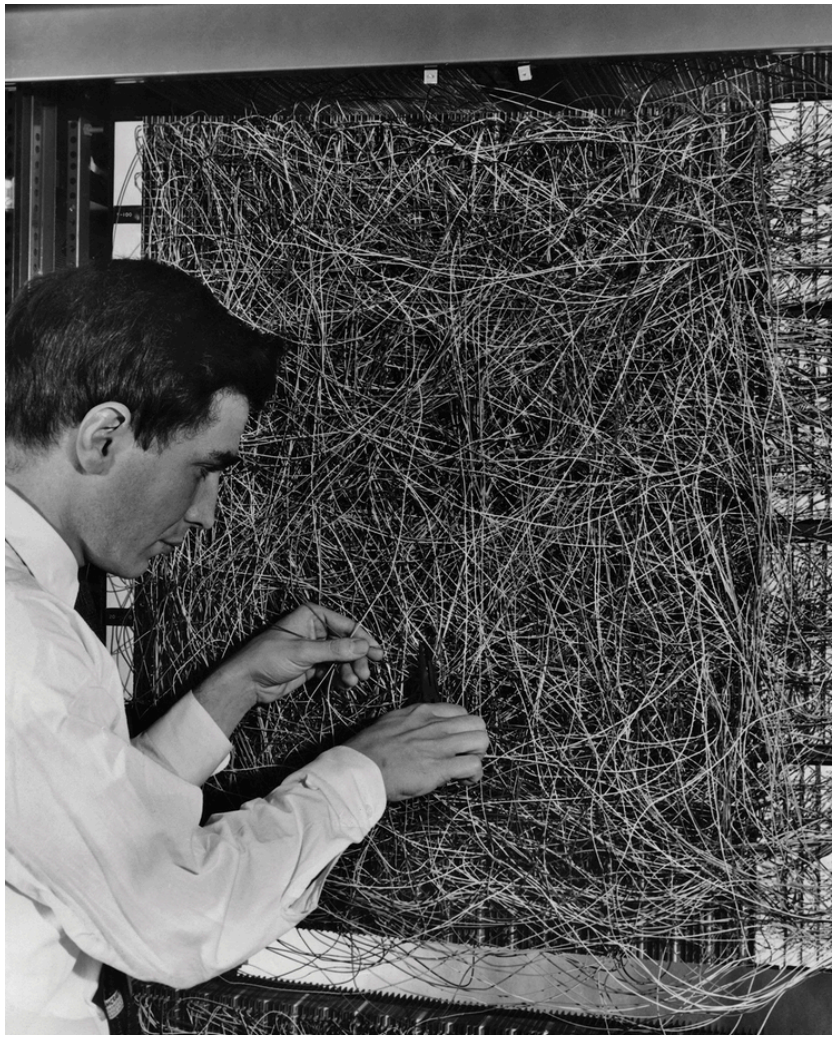
Perceptron
Invented at Calspan Buffalo, NY
Rosenblatt, Frank,
The Perceptron--a perceiving and recognizing automaton.
Report 85-460-1, 1957
Cornell Aeronautical Laboratory

Minsky and Papert
dedicated book to him
Minsky M. L. and Papert S. A. 1969
Perceptrons, MIT Press

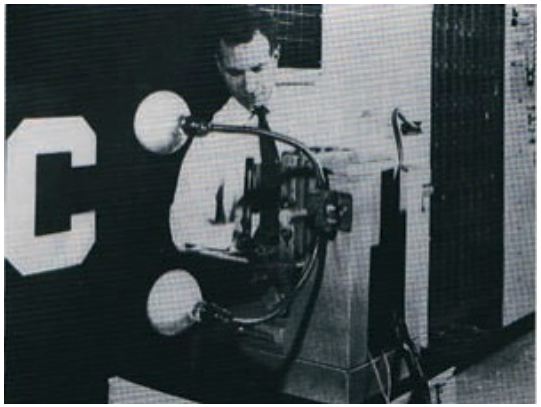
Perceptron







Perceptron Hardware (Analog)



Learning to discriminate shapes of characters
20x20 cell
Image of character

Patch-board to allow different configurations of input features φ

Racks of Adaptive Weights
Implemented as potentiometers

Known as Mark 1 Perceptron. It is now in the Smithsonian.

Disadvantages of Perceptrons

- Does not converge if classes not linearly separable
- Does not provide probabilistic output
- Not readily generalized to $K > 2$ classes

Summary

- Linear Discrimin. Funcs have simple geometry
- Extensible to multiple classes
- Parameters can be learnt using
  - Least squares
    - not robust to outliers, model close to target values
  - Fisher's linear discriminant
    - Two class is special case of least squares
    - Not easily generalized to more classes
  - Perceptrons
    - Does not converge if classes not linearly separable
    - Does not provide probabilistic output
    - Not readily generalized to $K>2$ classes