Linear Classifiers II, and Tangent Space for $k - NN$

Tomáš Svoboda

Vision for Robots and Autonomous Systems, Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague

May 23, 2022

Notes

Outline

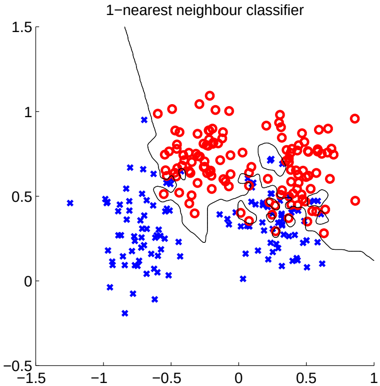
- $k - NN$, Tangent distance measure, invariance to rotation
- Better etalons by applying Fischer linear discriminator analysis.
- LSQ formulation of the learning task.

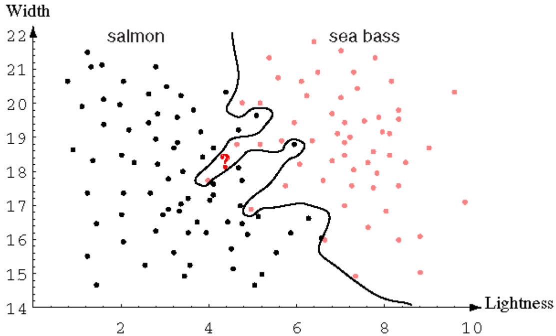
Notes

K-Nearest neighbors classification

For a query $\mathbf{x}$:

- Find K nearest $\mathbf{x}$ from the training (labeled) data.
- Classify to the class with the most exemplars in the set above.





Notes

Some properties:

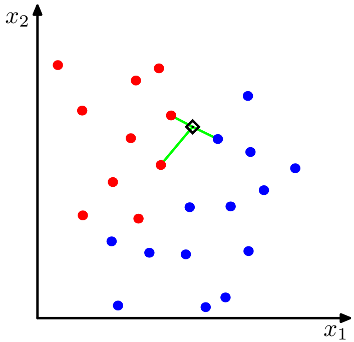
- A *nonparametric* method – does not assume anything about the distribution (that it is Gaussian etc.).
- Can be used for classification or regression. Here: classification.
- Training: Only store feature vectors and their labels.
- Very simple and suboptimal. With unlimited nr. prototypes, error never worse than twice the Bayes rate (optimum).
- *instance-based* or *lazy* learning – function only approximated locally; computation only during inference.
- Limitations
  - Curse of dimensionality - for every additional dimension, one needs exponentially more points to cover the space.
  - Comp. complexity - has to look through all the samples all the time. Some speed-up is possible. E.g., storing data in a K-d tree.
  - Noise. Missclassified examples will remain in the database....

$K-$ Nearest Neighbor and Bayes $j^* = \text{argmax}_j P(s_j|\mathbf{x})$

Assume data:

- N points $\mathbf{x}$ in total.
- N_j points in s_j class. Hence, $\sum_j N_j = N$.

We want to classify $\mathbf{x}$. Draw a sphere centered at $\mathbf{x}$ containing K points irrespective of class. V is the volume of this sphere. $P(s_j|\mathbf{x}) = ?$



$$P(s_j|\mathbf{x}) = \frac{P(\mathbf{x}|s_j)P(s_j)}{P(\mathbf{x})}$$

K_j is the number of points of class s_j among the K nearest neighbors.

$$P(s_j) = \frac{N_j}{N}$$

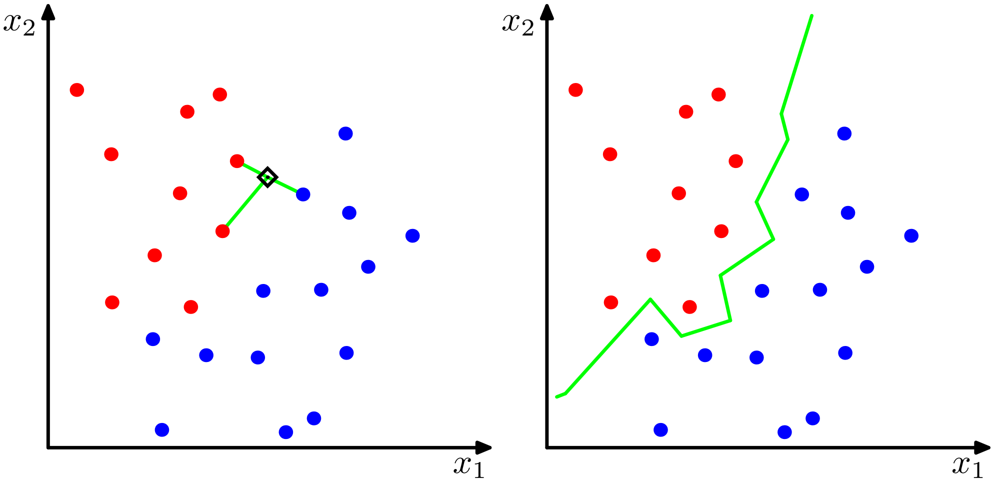
$$P(\mathbf{x}) = \frac{K}{NV}$$

$$P(\mathbf{x}|s_j) = \frac{K_j}{N_j V}$$

$$P(s_j|\mathbf{x}) = \frac{P(\mathbf{x}|s_j)P(s_j)}{P(\mathbf{x})} = \frac{K_j}{K}$$

Notes

NN classification example



(a) (b) [1]

---

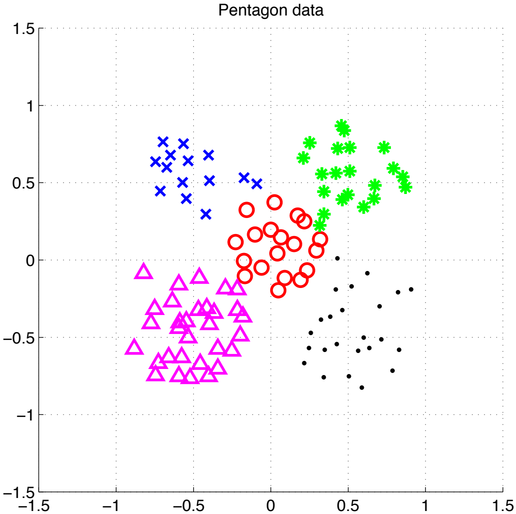
[1] Figs from [1]

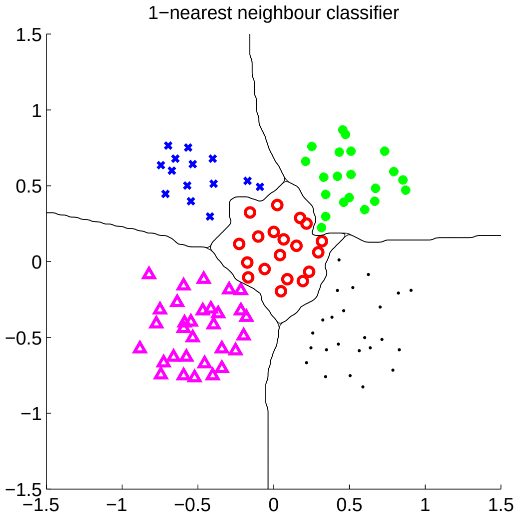
Notes

Left: k = 3.

Right: Decision boundary for k = 1.

NN classification example





Notes

Fast on "learning", very slow on decision.

There are ways for speeding it up, search for NN editing – making training data sparser, keeping only representative points.

What is *nearest*? Metrics for NN classification ...

Metrics : a function D which is

- nonnegative,
- reflexive,
- symmetrical,
- satisfying triangle inequality:

$$D(\mathbf{x_1}, \mathbf{x_2}) \geq 0$$
$$D(\mathbf{x_1}, \mathbf{x_2}) = 0 \text{ iff } \mathbf{x_1} = \mathbf{x_2}$$
$$D(\mathbf{x_1}, \mathbf{x_2}) = D(\mathbf{x_1}, \mathbf{x_2})$$
$$D(\mathbf{x_1}, \mathbf{x_2}) + D(\mathbf{x_2}, \mathbf{x_3}) \geq D(\mathbf{x_1}, \mathbf{x_3})$$

$D(\mathbf{x_1}, \mathbf{x_2}) = \|\mathbf{x_1} - \mathbf{x_2}\|$ just fine, but
...

Notes

Note, the minimum distance calculation can be reformulated into maximum similarity obtained by a dot product between the feature vector and the training examples.
When taking $\mathbf{x}$ as all the intensities, a “5” shifted 3 pixels left is farther from its etalon than to etalon of “8”. One could consider preprocessing:

1. shift query image to all possible positions and compute min distances
2. take the min(min(distance))
3. perform NN classification

Costly ...

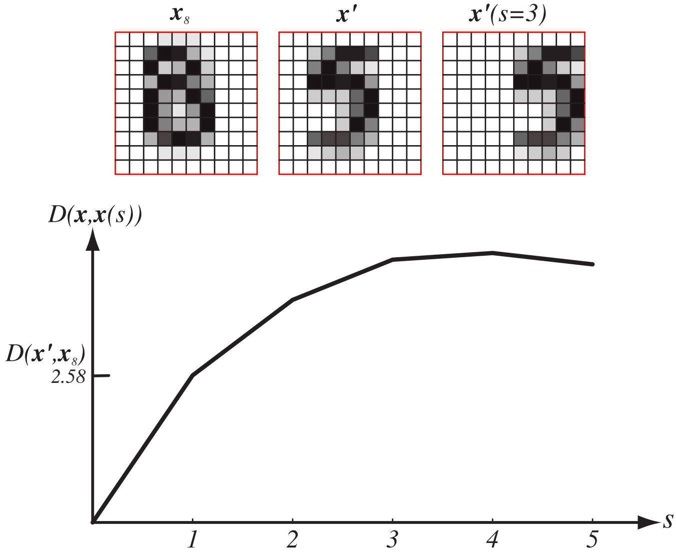
What is *nearest*? Metrics for NN classification ...

Metrics : a function D which is

- nonnegative,
- reflexive,
- symmetrical,
- satisfying triangle inequality:

$$D(\mathbf{x}_1, \mathbf{x}_2) \geq 0$$
$$D(\mathbf{x}_1, \mathbf{x}_2) = 0 \text{ iff } \mathbf{x}_1 = \mathbf{x}_2$$
$$D(\mathbf{x}_1, \mathbf{x}_2) = D(\mathbf{x}_1, \mathbf{x}_2)$$
$$D(\mathbf{x}_1, \mathbf{x}_2) + D(\mathbf{x}_2, \mathbf{x}_3) \geq D(\mathbf{x}_1, \mathbf{x}_3)$$

$D(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|$ just fine, but ...



Invariance to geometrical transformations? (figure from [2])

Notes

Note, the minimum distance calculation can be reformulated into maximum similarity obtained by a dot product between the feature vector and the training examples.
When taking $\mathbf{x}$ as all the intensities, a "5" shifted 3 pixels left is farther from its etalon than to etalon of "8". One could consider preprocessing:

1. shift query image to all possible positions and compute min distances
2. take the min(min(distance))
3. perform NN classification

Costly ...

Tangent space

Consider continuous tranformation:
e.g. rotation or translation not mirror reflection.

$\mathbf{x} = [x_1, x_2]^\top$ move along manifold $\mathcal{M}$
α is a tranformation parameter (e.g. angle)
Tangent vector $\boldsymbol{\tau}$ is a linearization

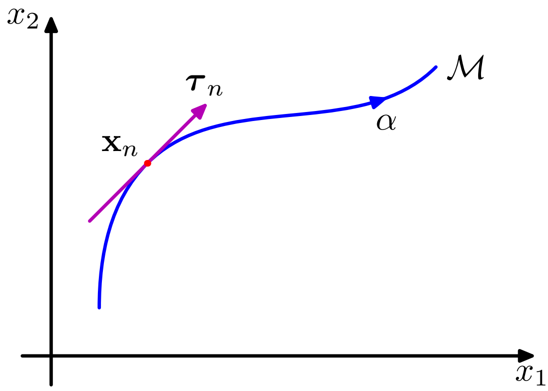
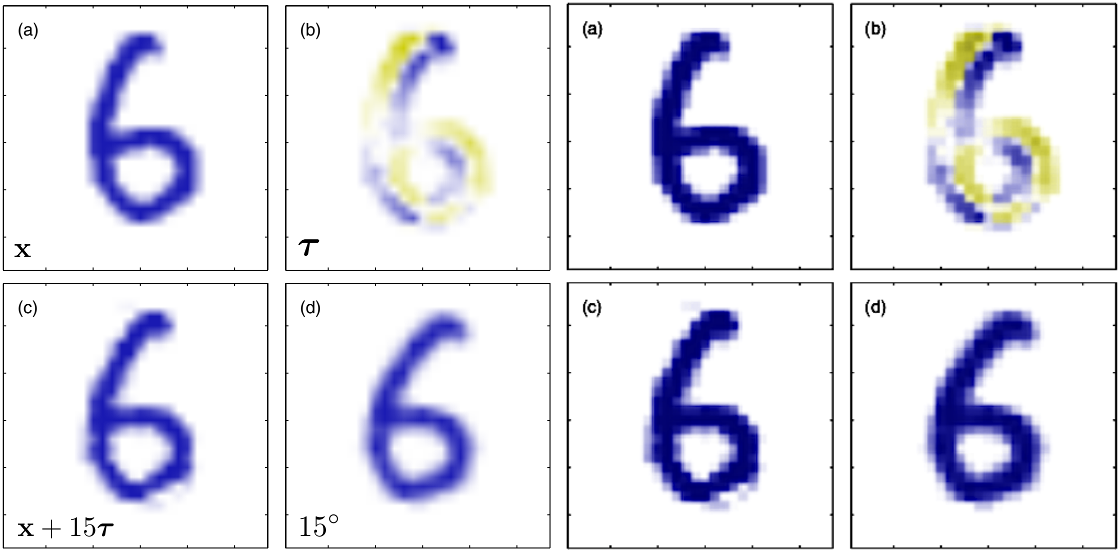


Figure from [1], slightly adapted

Notes

Approximating image rotation by adding tangent vector



Figures from [1], slighly adapted.

Notes

The right figure is just enhnanced version of the left one, for better visualisation.

a) orginal image $\mathbf{x}$

b) tangent vector $\boldsymbol{\tau}$ corresponding to clockwise rotation

c) result of $\mathbf{x} + 15\boldsymbol{\tau}$; simulating 15° rotation

d) actually rotated image, clockwise 15°.

Combining more transformations

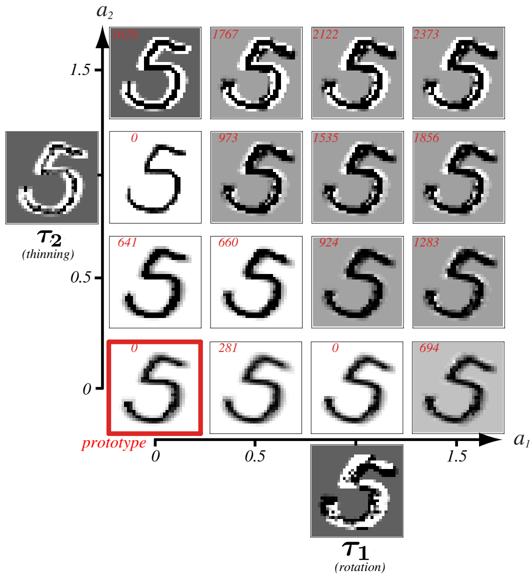
Approximate derivative by difference.
For all exemplars $\mathbf{x}'$
and all r tranformations $\mathcal{F}_i$

- $\boldsymbol{\tau_i} = \mathcal{F}_i(\mathbf{x}', \alpha_i) - \mathbf{x}'$

For each exemplar we have $d \times r$ matrix $\mathtt{T}$

$$\mathtt{T} = [\boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \cdots, \boldsymbol{\tau_r}]$$

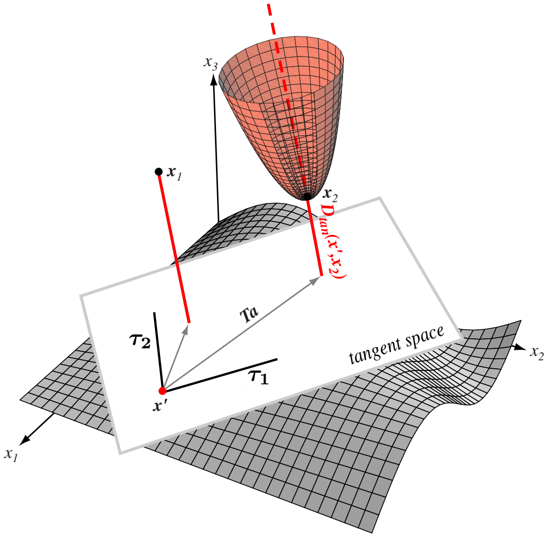
Grouping coefficients $\mathbf{a} = [a_1, a_2]^\top$
Right image visualizes $\mathbf{x}' + \mathtt{T}\mathbf{a}$



Figures from [2], slighly adapted.

Notes

Minimizing distance to tangent space



Figures from [2], slighly adapted.

$$D_{tan}(\mathbf{x}', \mathbf{x}) = \min_{\mathbf{a}} \|(\mathbf{x}' + \mathtt{T}\mathbf{a}) - \mathbf{x}\|$$

Gradient descent will do.

Notes

Linear classifiers II

$$g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

Decide s_1 if $g(\mathbf{x}) > 0$ and s_2 if $g(\mathbf{x}) < 0$

Figure from [2]

Notes

$g(\mathbf{x}) = 0$ is the *separating hyperplane*. Its dimension is one less that that of the input space – for 2D space, it is a line. (This is a bit counterintuitive - "hyper" normally means above, more...)
What is the geometric meaning of the weight vector $\mathbf{w}$?

One could mention the metaphor of the biological neuron here.

Linear classifiers II

$$g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

Decide s_1 if $g(\mathbf{x}) > 0$ and s_2 if $g(\mathbf{x}) < 0$

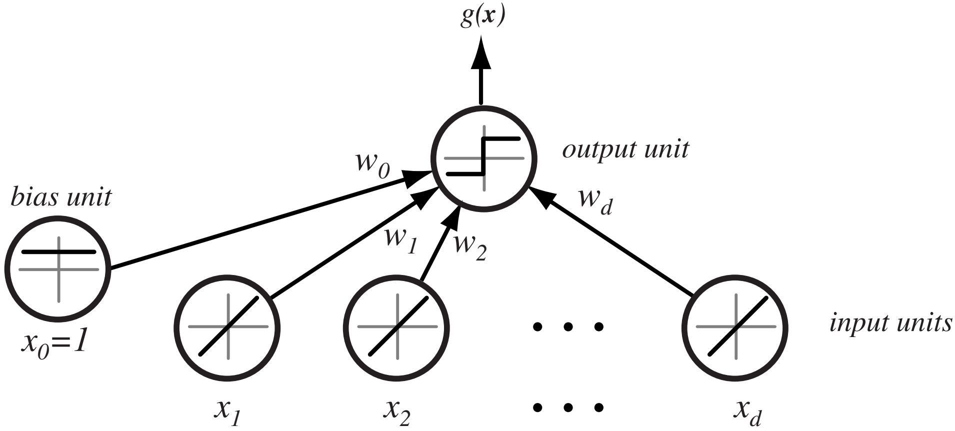


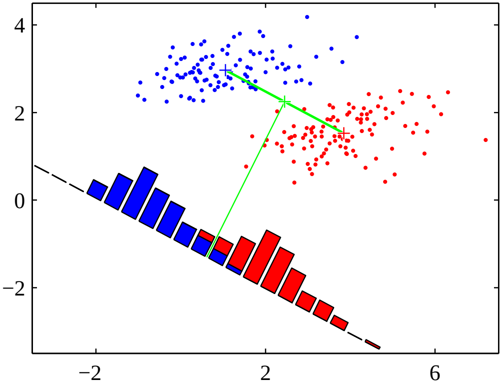
Figure from [2]

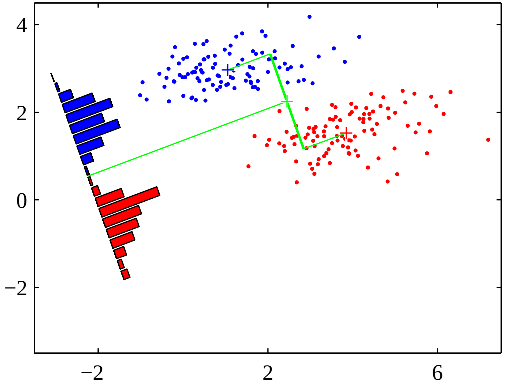
Notes

$g(\mathbf{x}) = 0$ is the *separating hyperplane*. Its dimension is one less that that of the input space – for 2D space, it is a line. (This is a bit counterintuitive - "hyper" normally means above, more...)
What is the geometric meaning of the weight vector $\mathbf{w}$?

One could mention the metaphor of the biological neuron here.

Fischer linear discriminant



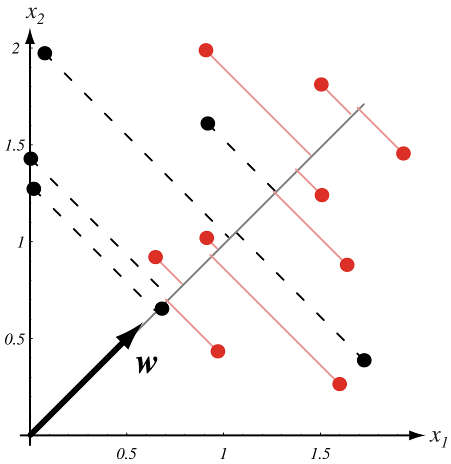


- Dimensionality reduction
- Maximize distance between means, . . .
- . . . and minimize within class variance. (minimize overlap)

Figures from [1]

Notes

Projections to lower dimensions $y = \mathbf{w}^\top \mathbf{x}$



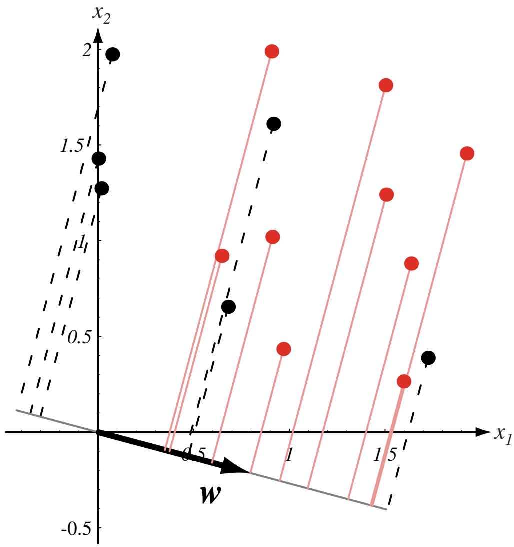


Figure from [2]

Notes

Keep in mind we assume normalized $\mathbf{w}$ hence, $\|\mathbf{w}\| = 1$.

Projection to lower dimension $\mathbf{y} = \mathtt{W}^\top \mathbf{x}$

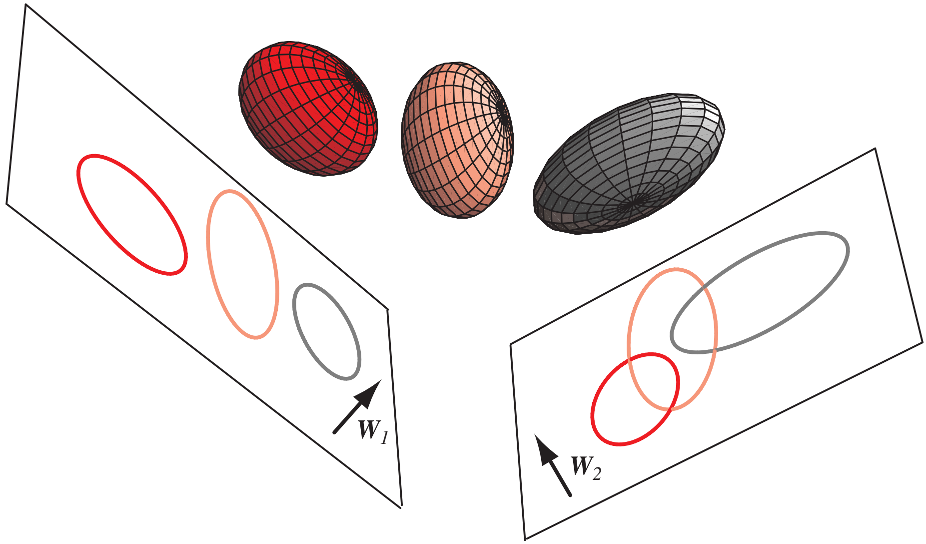
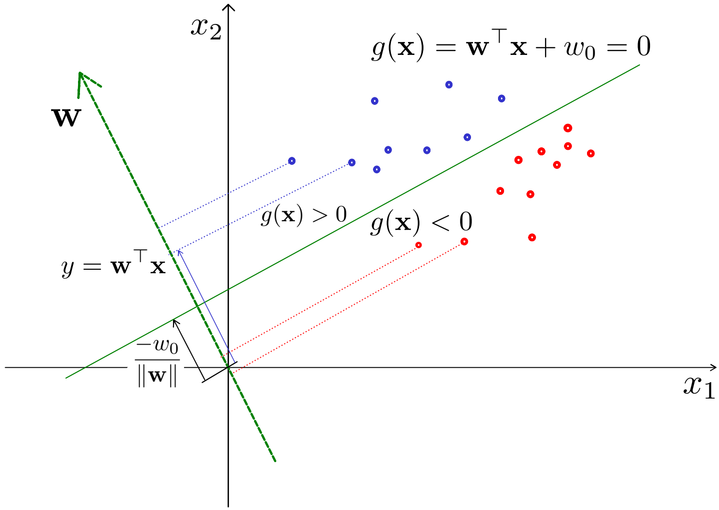
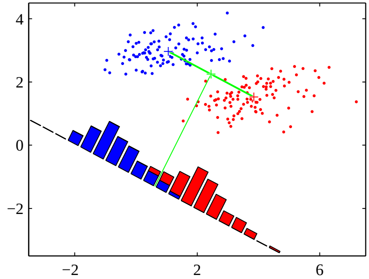


Figure from [2]

Notes

Finding the best projection $y = \mathbf{w}^\top \mathbf{x}$, $y \geq -w_0 \Rightarrow C_1$, otherwise C_2





Notes

This is just to make sure we understand geometric meaning of $\mathbf{w}$, w_0 and the separating hyperplane. Remind the vector notation $\mathbf{w}$ means the same as $\vec{w}$.

Finding the best projection $y = \mathbf{w}^\top \mathbf{x}$, $y \geq -w_0 \Rightarrow C_1$, otherwise C_2

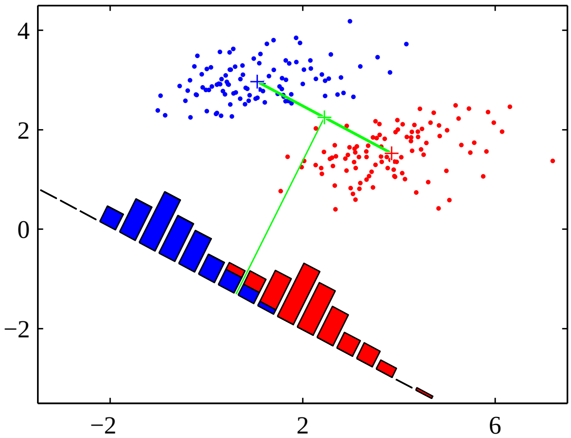
$$m_2 - m_1 = \mathbf{w}^\top(\mathbf{m}_2 - \mathbf{m}_1)$$

Within class scatter of projected samples

$$s_i^2 = \sum_{y \in C_i} (y - m_i)^2$$

Fischer criterion:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$



Notes

Fischer criterion, max or min?

Finding the best projection $y = \mathbf{w}^\top \mathbf{x}$, $y \geq -w_0 \Rightarrow C_1$, otherwise C_2

$$m_2 - m_1 = \mathbf{w}^\top(\mathbf{m}_2 - \mathbf{m}_1)$$

$$s_i^2 = \sum_{y \in C_i} (y - m_i)^2$$

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0$$

$$\mathtt{S}_i = \sum_{x \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^\top$$

$$\mathtt{S}_W = \mathtt{S}_1 + \mathtt{S}_2$$

$$\mathtt{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top$$

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathtt{S}_B \mathbf{w}}{\mathbf{w}^\top \mathtt{S}_W \mathbf{w}}$$

Notes

$\mathtt{S}_B$ stands for the *between* class scatter matrix. Remind

$$\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$$

hence we seek:

$$2\mathtt{S}_B\mathbf{w}(\mathbf{w}^\top \mathtt{S}_W \mathbf{w}) = (\mathbf{w}^\top \mathtt{S}_B \mathbf{w}) 2\mathtt{S}_W \mathbf{w}$$

the expressions within bracket are (unknown) scalars

$$\mathtt{S}_B \mathbf{w} = \lambda \mathtt{S}_W \mathbf{w}$$

leading to eigenvalue problem

$$\mathtt{S}_W^{-1} \mathtt{S}_B \mathbf{w} = \lambda \mathbf{w}$$

However, $\mathtt{S}_B\mathbf{w}$ is always in direction $(\mathbf{m}_2 - \mathbf{m}_1)$, and scale is not important

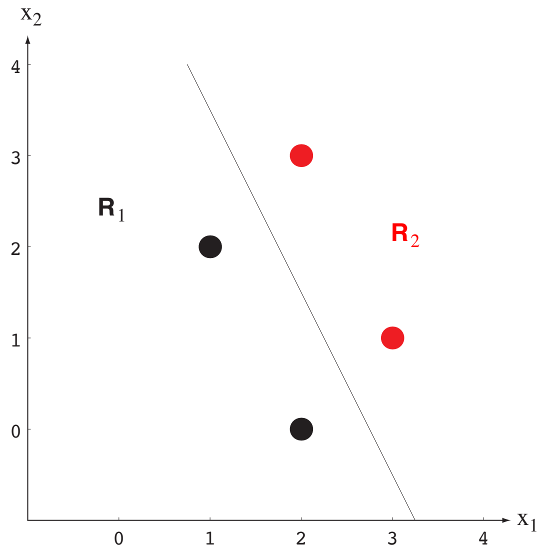
$$\mathbf{w} = \mathtt{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

LSQ approach to linear classification

$$\mathbf{w} = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

$$\mathtt{X}\mathbf{w} = \mathbf{b}$$

$$J(\mathbf{w}) = \|\mathtt{X}\mathbf{w} - \mathbf{b}\|^2$$



Notes

Write dimensions to each symbol, n may stand for the number of points, d for dimensionality of the feature space.
Solving

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0$$

yields $\mathbf{w} = (\mathtt{X}^\top \mathtt{X})^{-1}\mathtt{X}^\top \mathbf{b}$ Try to solve the above figure. We are looking for a separating hyperplane

$$\mathbf{w}^\top \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = 0$$

and we want points in training set distant from the hyperplane

$$\mathtt{X} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{bmatrix}$$

$$\mathbf{b} = [1\ 1\ 1\ 1]^\top$$

Linear least squares not guaranteed to correctly classify everything on the training set. It's objective function not perfect for classification. Margins $\mathbf{b}$ were set quite arbitrarily.

Outliers can shift the decision boundary.

LSQ approach, better margins $\mathbf{b}$?

$$\mathtt{X} = \begin{bmatrix} \mathtt{1}_1 & \mathtt{X}_1 \\ -\mathtt{1}_2 & -\mathtt{X}_2 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} \frac{n}{n_1}\mathtt{1}_1 \\ \frac{n}{n_2}\mathtt{1}_2 \end{bmatrix}$$

Notes

After some derivation it can be shown the LSQ solution is equivalent to Fisher linear discriminant
insert into intermediate result when solving $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0$

$$\mathtt{X}^\top \mathtt{X}\mathbf{w} = \mathtt{X}^\top \mathbf{b}$$

References I

Further reading: Chapter 4 of [1], or chapter 3 and 5 of [2].

[1] Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer Science+Bussiness Media, New York, NY, 2006.
https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf.

[2] Richard O. Duda, Peter E. Hart, and David G. Stork.
Pattern Classification.
John Wiley & Sons, 2nd edition, 2001.

Notes