ECEN 649 Pattern Recognition

Perceptrons

Ulisses Braga-Neto

ECE Department

Texas A&M University

Adjustable Discriminant Rules

- Both parametric and nonparametric classification rules are plug-in rules, that is, they involve some form of distribution estimation using training data.
- We will consider now a different idea:
  - Assume a set of discriminants (decision boundaries)
  - Search for the discriminant that best fits the data by optimizing some objective criterion (*“learning”*)
- This is the basic idea behind many popular “machine learning” classification rules:
  - Perceptrons
  - Support Vector Machines
  - Neural Networks
  - Decision Trees

Rosenblatt's Perceptron

- This was the first adjustable discriminant rule, proposed in the late 50's by F. Rosenblatt .
- Assume a linear discriminant function

$$g_n(x) = a_0 + \sum_{i=1}^{d} a_i x_i$$

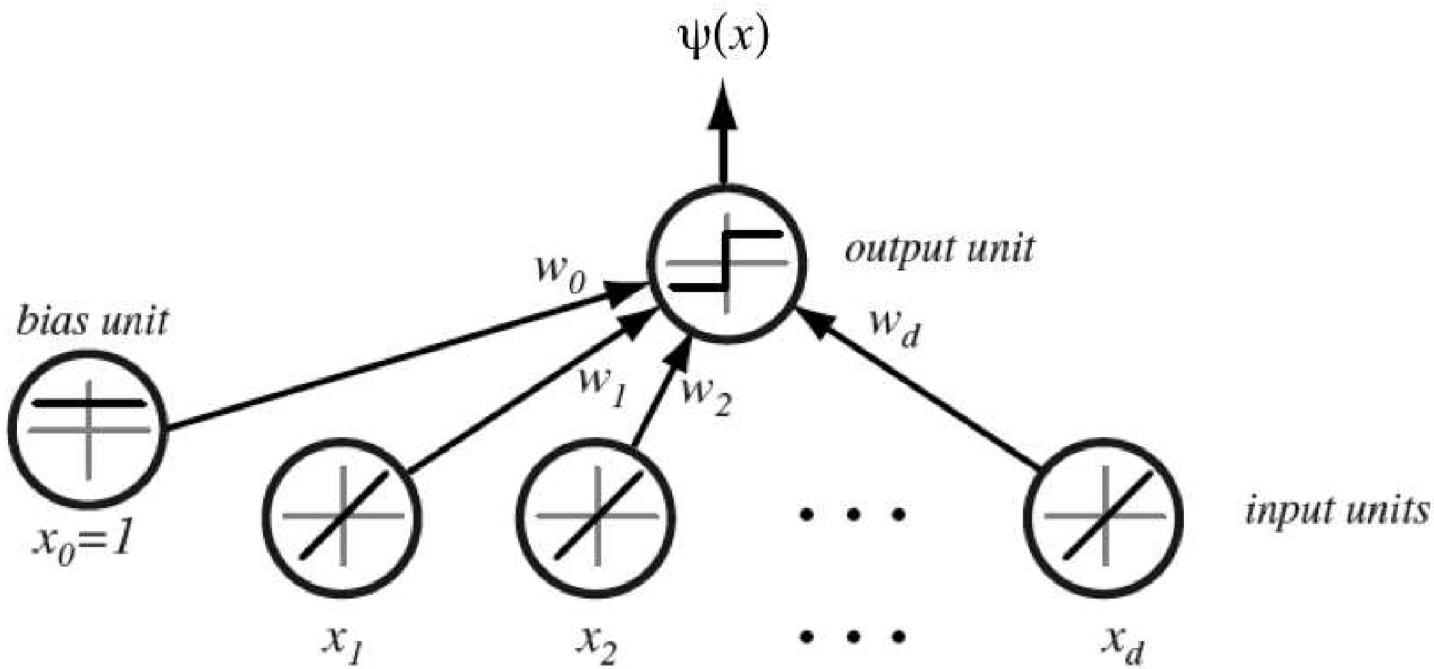
so that the designed classifier is given by

$$\psi_n(x) = \begin{cases} 1\,, & \text{if } a_0 + \sum a_i x_i \geq 0 \\ 0\,, & \text{otherwise} \end{cases}$$

- Adjust ("learn") the parameters $a_0, a_1, \ldots, a_d$ based on the training data.

Rosenblatt's Perceptron - II

- This corresponds to the (single-layer) perceptron, depicted as follows.



Augmented Feature Vector

- Given the feature vector $x \in R^d$, consider the augmented vector:

$$x' = \begin{bmatrix} 1 \\ x \end{bmatrix} \in R^{d+1}$$

so that the perceptron classifier is given by

$$\psi_n(x') = \begin{cases} 1\,, & \text{if } a^T x' \geq 0 \\ 0\,, & \text{otherwise} \end{cases}$$

where $a = [a_0, a_1, \ldots, a_d]^T \in R^{d+1}$ is the parameter vector.

Augmented Feature Vector - II

- Given training data point (x_i, y_i), define likewise the augmented vector x_i', for $i = 1, \ldots, n$.
- Correct classification of x_i happens when

$$a^T x_i' \geq 0 \quad \text{if} \quad y_i = 1$$

$$a^T x_i' < 0 \quad \text{if} \quad y_i = 0$$

- Trick: flip the sign of x_i' if $y_i = 0$ so that

$$\text{correct classification of } x_i \quad \Leftrightarrow \quad a^T x_i' \geq 0$$

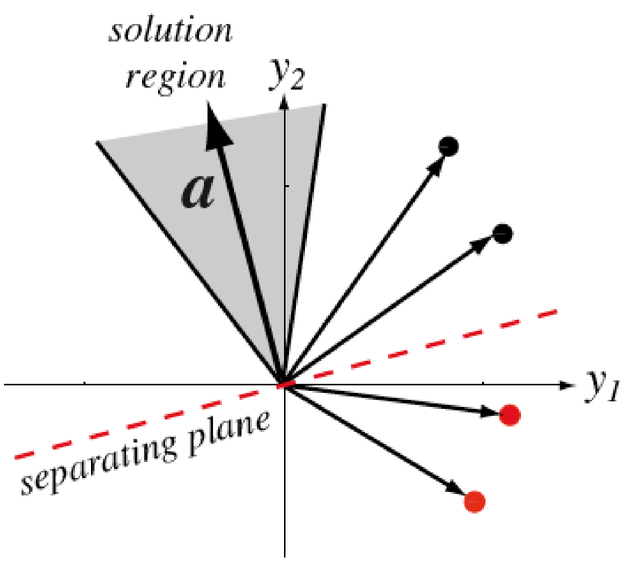
- For convenience, the prime will be omitted from the augmented vector in what follows.

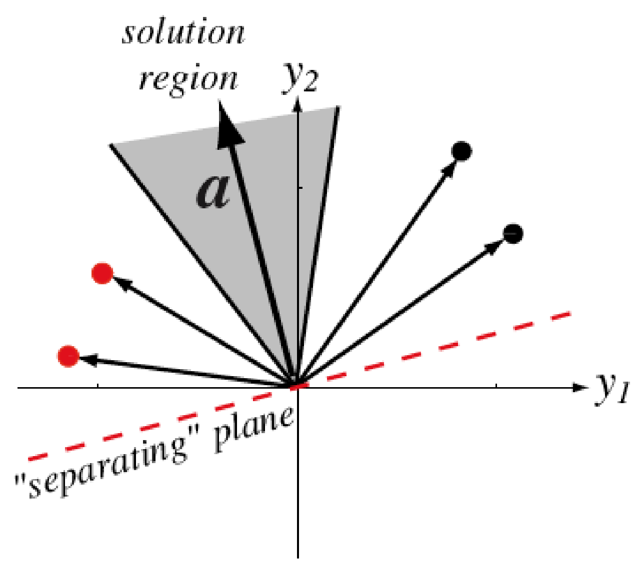
Linearly Separable Case

- The data is said to be *linearly separable* is there is a perceptron with zero apparent error, that is, one can find a (not unique) such that

$$a^T x_i \geq 0\,, \quad i = 1, \ldots, n$$

- Example: $n = 4$, left: raw data, right: “flipped" data.





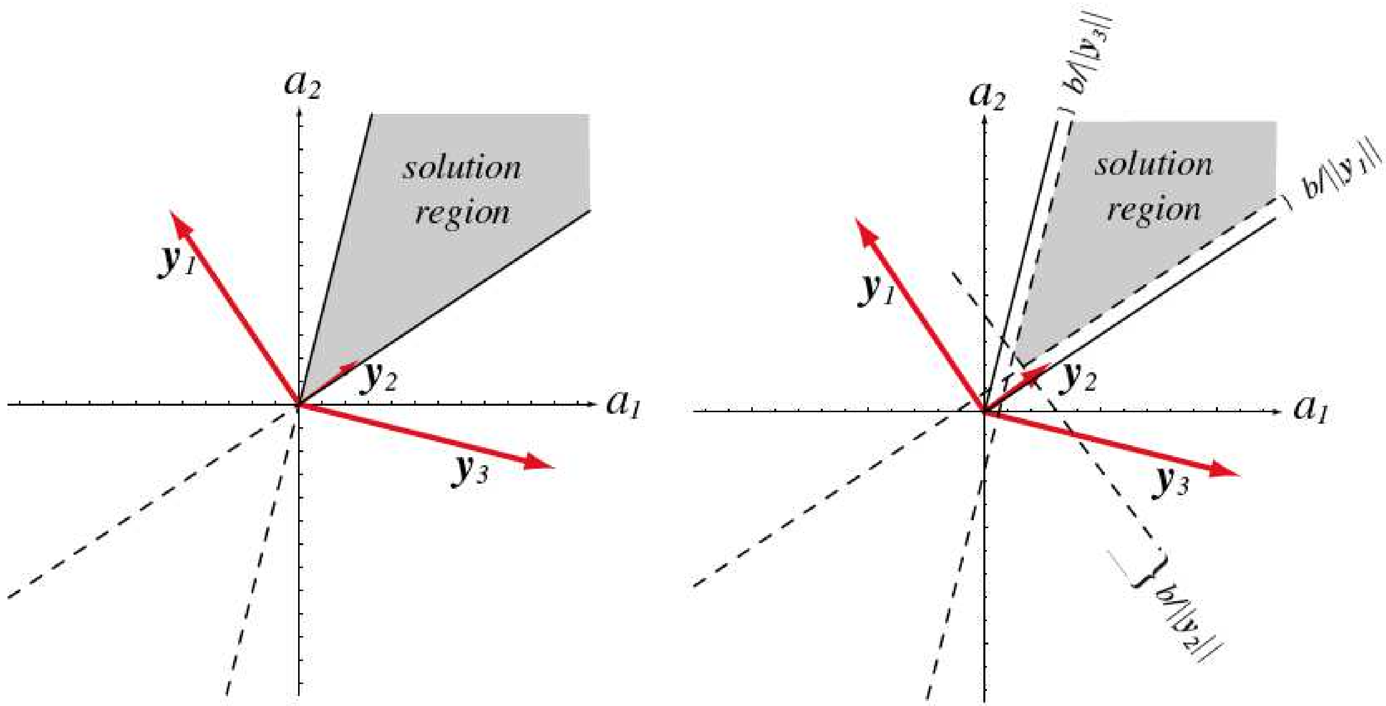
Perceptron with Margin

- Another possibility is to look for a vector a satisfying

$$a^T x_i \geq b\,, \quad i = 1, \ldots, n$$

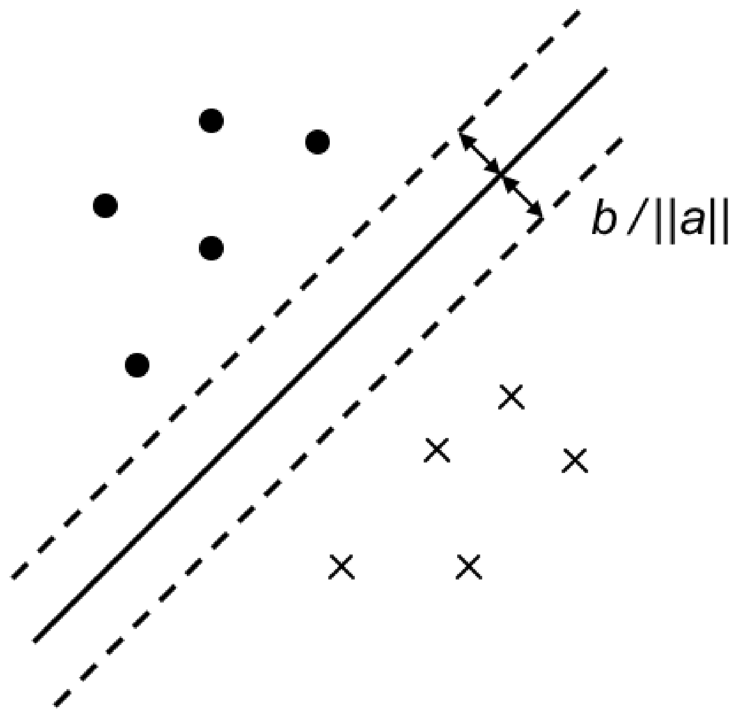
where $b > 0$ is the *margin*.

- Example: left: no margin, right: positive margin.



Perceptron with Margin - II

- The distance of point x_i to the decision hyperplane is $|g(x_i)|/||a|| = |a^T x_i|/||a||$. The perceptron with margin thus guarantees that all data points are at a distance at least $b/||a||$ from the hyperplane (this is also the main idea behind support vector machines, as will be seen).



Gradient Descent

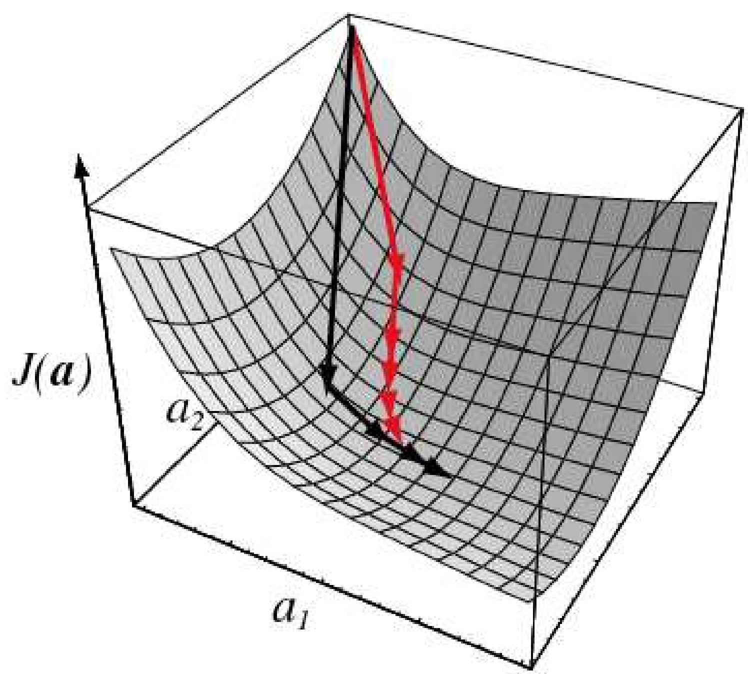
- Classical iterative optimization technique.
- Find a criterion function $J(a) \geq 0$ such that $J(a)$ is minimal ($J(a) = 0$) whenever a in the solution region.
- Gradient descent algorithm ("follow the gradient"):
  - Let $a(0) = a_0$ (initial guess).
  - At step $k \geq 1$,

$$\begin{aligned} \nabla J(a(k)) &\leftarrow \text{ gradient of } J \text{ at } a(k) \\ \ell(k) &\leftarrow \text{ step length} \\ a(k+1) &\leftarrow a(k) - \ell(k)\nabla J(a(k)) \end{aligned}$$

  - Stop when $J(a(k)) = 0$ or $|\ell(k)\nabla J(a(K))| < \tau$

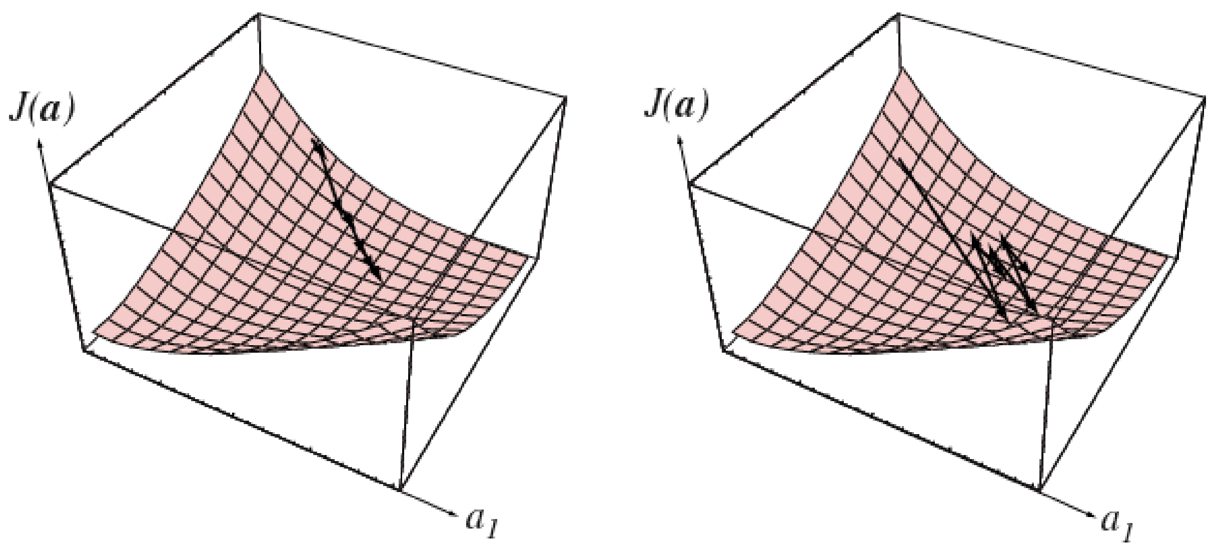
Gradient Descent - II

- Graphical example: gradient descent is depicted in red below. In black is a different kind of descent algorithm, called Newton's algorithm, which converges faster but requires matrix inversion and so may be unstable.



Learning Rate

- The step lengths $\ell(k)$ have to be selected carefully to ensure convergence: one wants $l(k) \to 0$ as $k \to \infty$, but not too fast ("overrelaxation") nor too slow ("underrelaxation"). In both cases, the algorithm may fail to converge in a reasonable number of steps.
- Example: left: overrrelexation. right: underrelaxation.



Perceptron Criterion Function

- The choice of criterion function $J(a)$ is fundamental.
- A naive criterion function such as the number of errors

$$J(a) = \sum_{i=1}^{n} I_{\{a^T x_i < 0\}}$$

  cannot work, because it is piecewise constant ($J(a) \in \{0, 1, \ldots, n\}$), and therefore unsuitable for gradient descent.
- Consider instead the criterion function

$$J_p(a) = \sum_{i=1}^{n} -(a^T x_i) I_{\{a^T x_i < 0\}}$$

Perceptron Criterion Function - II

- This is called the *perceptron criterion function*.
- It is piecewise linear and suitable for gradient descent.
- Note that

$$\nabla J_p(a) = \sum_{i=1}^{n} -x_i I_{\{a^T x_i < 0\}}$$

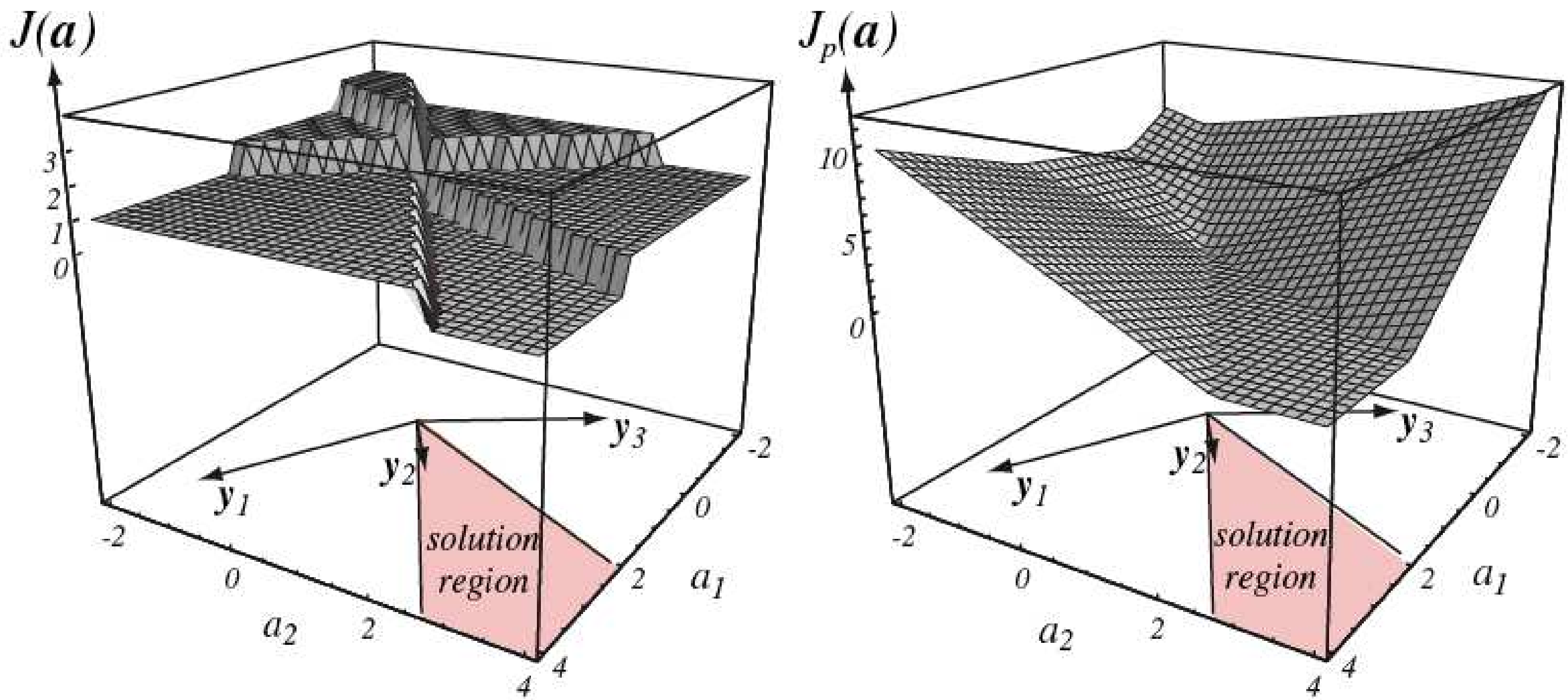
- Therefore, the basic iterative step in the gradient descent algorithm becomes

$$a(k+1) \leftarrow a(k) + \ell(k) \sum_{i=1}^{n} x_i I_{\{a^T x_i < 0\}}$$

that is, at each step, the current vector a is “corrected” in the direction of the misclassified data points.

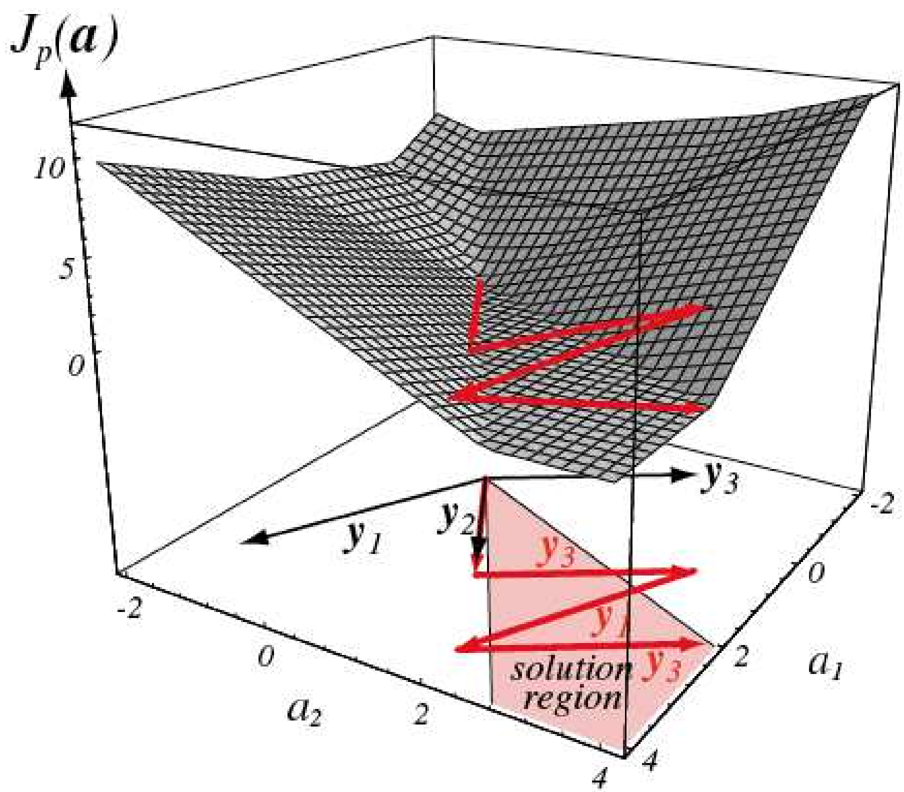
Perceptron Criterion Function - III

- Example: naive (left) and perceptron (right) criterion functions.



Perceptron Criterion Function - IV

- Example of gradient descent algorithm with perceptron criterion function.



Alternative Criterion Functions

- Quadratic criterion function:

$$J_q(a) = \sum_{i=1}^{n} (a^T x_i)^2 I_{\{a^T x_i < 0\}}$$

This is too smooth near the boundary of the solution region (so one can get the trivial solution $a = 0$). It is also dominated by large data vectors.

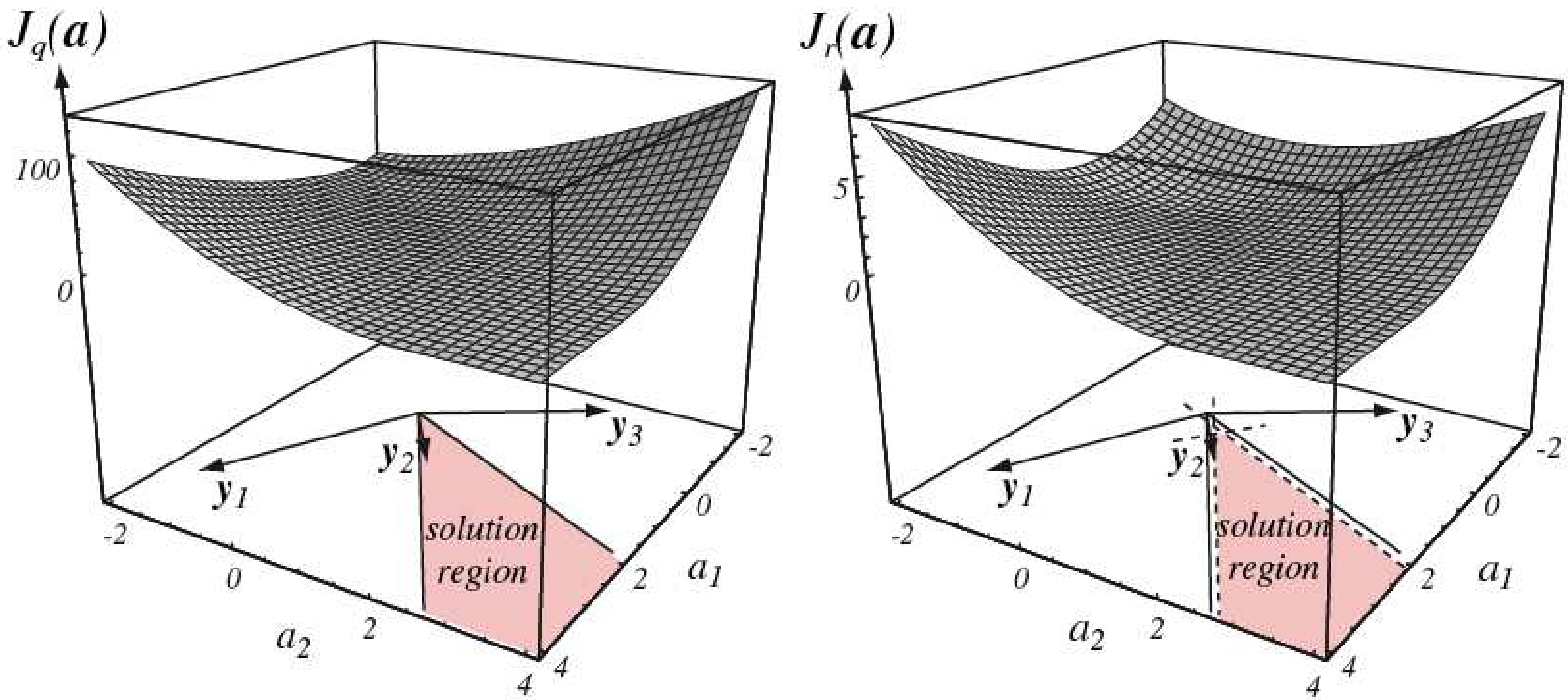
- Normalized quadratic criterion function with margin:

$$J_r(a) = \frac{1}{2} \sum_{i=1}^{n} \frac{(a^T x_i - b)^2}{||x_i||^2} I_{\{a^T x_i < b\}}$$

This avoids boundary points with the margin and also normalizes against large data vectors.

Alternative Criterion Functions - II

- Example: quadratic (left) and normalized with margin (right) criterion functions.



Nonseparable Case

- If the data is nonseparable, then the solution region is empty and there is no linear classifier that can achieve zero apparent error.
- This means that $J(a)$ is never zero and the gradient descent algorithm may run indefinitely.
- A sensible stopping rule then becomes essential.

Overfitting

- All adjustable-discriminant classification rules, as we will see, are based on *iterative* search. With increasing number of iterations, the apparent error on the training data decreases, but the true classification error may increase after a certain point.

- This leads to *overfitting*, which greatly affects adjustable discriminant rules.

- One recalls that LDA can also be seen as fitting a linear discriminant to the data based on an optimality criterion (e.g. Fisher's criterion), but in the case there is a closed formula solution and no iterative adjustment.

Overfitting - II

- Graphical interpretation of overfitting.