

CS4770

Pattern Recognition

Linear Discriminant Functions

P. J. Narayanan
Monsoon 2005

Discriminant Functions

- Discriminant functions classify patterns directly.
- When the classes are Gaussians with identical covariance matrices, the discriminant functions were **linear**.
- If the space of \mathbf{x} is not linear, the space of a function y of \mathbf{x} could be linear.
- Assuming a linear form for discriminant functions, how do we learn them from the examples?
- Pose it as a minimization of a criterion function, often the training error.

Linear Discriminant Functions

- $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$. \mathbf{w} is the weight vector, w_0 is the threshold or the bias.
- In the 2-category case, the decision boundary is: $g(\mathbf{x}) = 0$. Positive side belongs to ω_1 and the negative side to ω_2 .
- For 2 points on the decision boundary: $\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$.
- \mathbf{w} is the normal to the decision boundary (a hyperplane). The positive side is region R_1 and the negative side is region R_2 .
- $r = g(\mathbf{x}) / \|\mathbf{w}\|$: distance from \mathbf{x} to the hyperplane.

- $w_0/||\mathbf{x}||$: distance from origin. If the bias is positive, origin is in R_1 else in R_2 .
- Multicategory case: define c discriminant functions g_i . Choose $j = \arg \max_i g_i(\mathbf{x})$. (**Linear Machine**)
- Boundary between 2 categories: hyperplane H_{ij} : $g_i(\mathbf{x}) = g_j(\mathbf{x})$. Also given by: $(\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{x} + (w_{i0} - w_{j0}) = 0$.
- $(\mathbf{w}_i - \mathbf{w}_j)$ is the normal. Distance to H_{ij} : $\frac{g_i(\mathbf{x}) - g_j(\mathbf{x})}{||\mathbf{w}_i - \mathbf{w}_j||}$.
- Decision regions are *convex* and *simply connected* since they are intersections of hyperplanes.
- Of the $c(c - 1)/2$ pairs, only a few H_{ij} s matter.

Augmented Vectors

- Can write the discriminant function as: $g(\mathbf{y}) = \mathbf{a}^T \mathbf{y}$ where $\mathbf{a} = [w_0 \ w_1 \ w_2 \ \dots \ w_d]^T$ and $\mathbf{y} = [1 \ x_1 \ x_2 \ \dots \ x_d]^T$.
- \mathbf{a} and \mathbf{y} are now **augmented** weight vector and feature vector respectively.
- Discriminant functions are homogeneous equations. The hyperplane passes through the origin of the augmented space, but not necessarily the original feature space.
- Distance to hyperplane: $\mathbf{a}^T \mathbf{y} / \|\mathbf{a}\|$.

Generalized Linear Discriminants

- Quadratic discriminants:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^d w_i x_i + \sum_{i,j} w_{ij} x_i x_j.$$

- Similarly, general polynomial discriminant, etc.
- Define **generalized linear discriminant function** as:

$$g(\mathbf{x}) = \mathbf{a}^T \mathbf{y} = \sum_{i=0}^{d'} a_i y_i(\mathbf{x})$$

- Let $y_i(\mathbf{x})$, $i = 1 \dots d'$ be arbitrary functions $\varphi_i(\mathbf{x})$ of \mathbf{x} .
- Feature space of \mathbf{x} transformed via $y_i = \varphi_i(\mathbf{x})$ to the space of \mathbf{y} , where discriminant is a linear function.

- The separating hyperplanes in y -space can map to complex regions in x -space.
- The discriminant function is given by $\mathbf{a}^T \mathbf{y} = 0$ in y -space.
- Example 1: Separating boundary of a circle. Quadratic in XY space, but a line ($r = r_1$) in the polar coordinates.
- Example 2: ω_1 : 1st and 3rd quadrants , ω_2 : 2nd and 4th. A transformation to a 1-D space where $y' = xy$ will linearly separate the regions!

2-Class, Linear Classifier Design

- Our problem is classifier design or estimation of the hyperplane normal \mathbf{a} given the training samples.
- Correct classification: $\mathbf{a}^T \mathbf{y} > 0$ and \mathbf{y} labelled ω_1 or $\mathbf{a}^T \mathbf{y} < 0$ and \mathbf{y} labelled ω_2 .
- Bring training points from ω_1 and ω_2 to a common form.
- If we negate all y_i classified as ω_2 , then $\mathbf{a}^T \mathbf{y} > 0$ for all correct classifications.
- The \mathbf{a} vector in this case is called the **separating vector**.

Solution Space

- $\mathbf{a}^T \mathbf{y} > 0$ for all correct classification.
- Consider the solution space of values a_i . \mathbf{y} is the normal to the hyperplane in that space. All points on the positive half-space are solutions!
- With n training samples, the solution region is restricted to the intersection of n positive half-spaces.

General Problem

- Find the weight vector \mathbf{a} such that $\mathbf{a}^T \mathbf{y} > 0$.
- Search in the weight space for the best solution. There could be a solution region and a “middle solution” is preferred.
- Add a *margin* b by requiring: $\mathbf{a}^T \mathbf{y} \geq b > 0$. The solution is offset to the middle from the boundary by $b/\|\mathbf{y}\|$.
- Can pose it as: Minimize a criterion function $J(\mathbf{a})$ with respect to \mathbf{a} in the weight space.
- Several forms of J are possible.

Gradient, Jacobian, Hessian

- **Gradient:** Derivative of a scalar function with respect to a vector. Result is a vector.

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_d} \right]^T$$

- **Jacobian:** Derivative of a vector-valued function with respect to a vector. Result is a matrix.

$$\mathbf{J}_{\mathbf{x}}(\mathbf{f}(\mathbf{x})) = [\nabla_{\mathbf{x}} f_1(\mathbf{x}) \cdots \nabla_{\mathbf{x}} f_k(\mathbf{x})]^T = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_d} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_k}{\partial x_1} & \frac{\partial f_k}{\partial x_2} & \cdots & \frac{\partial f_k}{\partial x_d} \end{bmatrix}$$

- **Hessian:** Second derivative of a scalar function with respect to a vector. Result is a square matrix.

$$\mathbf{H}_{\mathbf{x}}(f(\mathbf{x})) = \nabla_{\mathbf{x}}^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial^2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial^2 \partial x_d} \end{bmatrix}$$

- **Series expansion:**

$$f(\mathbf{x}) = f(\mathbf{x}_0) + [\nabla_x f(\mathbf{x})]^\top \Delta \mathbf{x} + \frac{1}{2} [\Delta \mathbf{x}]^\top \mathbf{H}_{\mathbf{x}}(f(\mathbf{x})) \Delta \mathbf{x} + \cdots$$

Gradient Descent Method

- Adjust \mathbf{a} iteratively along its gradient direction towards the minimum.
- If gradient is +ve, function increases with \mathbf{a} , so we need to decrease \mathbf{a} . If gradient is -ve, function decreases with \mathbf{a} , so we need to increase \mathbf{a} .
- In either case, adjust \mathbf{a} in the opposite direction to the gradient!
- $\mathbf{a}(k + 1) = \mathbf{a}(k) - \eta(k) \nabla J(\mathbf{a}(k))$ for positive η will take it closer to the minimum of J .

- $J(\mathbf{a}(k + 1)) = J(\mathbf{a}(k)) + \nabla J(\mathbf{a}(k))\Delta\mathbf{a}$ using Taylor series expansion.
- $J(\mathbf{a}(k + 1)) - J(\mathbf{a}(k)) = \nabla J(\mathbf{a}(k))[-\eta(k)\nabla J(\mathbf{a}(k))] = -\eta(k)\|\nabla J(\mathbf{a}(k))\|^2.$
- J moves towards the minimum in each step slowly, based on the **learning rate** η .
- Large η : giant strides. Can oscillate around the minimum.
Small η : crawl slowly towards the minimum.

Newton's Method

- $J(\mathbf{a}) = J(\mathbf{a}(k)) + \nabla J^T(\mathbf{a} - \mathbf{a}(k)) + \frac{1}{2}(\mathbf{a} - \mathbf{a}(k))^T \mathbf{H}(\mathbf{a} - \mathbf{a}(k))$
- To minimize, differentiate with respect to \mathbf{a} and set to 0.
- $\nabla J + \mathbf{H}(\mathbf{a} - \mathbf{a}(k)) = 0$. Thus, $\mathbf{a} - \mathbf{a}(k) = -\mathbf{H}^{-1} \nabla J$
- Therefore, $\mathbf{a} = \mathbf{a}(k) - [\mathbf{H}(J(\mathbf{a}(k)))]^{-1} \nabla J$
- Iteratively adjust \mathbf{a} . Moves closer to minimum at a faster rate than gradient descent.
- Valid only when the Hessian is positive definite, non-singular, etc.
- Also, a matrix inversion is required in each step!

Linear Perceptron

- A machine that implements any linear decision boundary.
- Given a vector of inputs \mathbf{y} , it has a weight vector \mathbf{a} and computes the function: $\mathbf{a}^T \mathbf{y}$.
- A perceptron can classify any pair of linearly separable classes.
- Classify \mathbf{x}_i as ω_1 if $\mathbf{w}^T \mathbf{x}_i > 0$. Else as ω_2 .
- $\mathbf{a}^T \mathbf{y} > 0$ for all correctly classified inputs.
 $\mathbf{a}^T \mathbf{y} < 0$ for misclassified inputs.

Choice of Criterion Function J

- $J(\mathbf{a})$ is an error measure since we seek its minimum.
- Number of misclassified samples: Piecewise constant, no gradients!
- Sum of $\mathbf{a}^T \mathbf{y}$ for misclassified samples: continuous and differentiable. Too smooth near boundaries, so can settle on it. Influenced by large sample vectors.
- Normalized, squared sum of $\mathbf{a}^T \mathbf{y}$ with margin: Continuous, settles away from the boundary.
- Guaranteed to converge on linearly separable training sets.

Perceptron Criterion Function

- Perceptron criterion function: $J_p(\mathbf{a}) = \sum_{\mathcal{Y}} (-\mathbf{a}^T \mathbf{y})$ where \mathcal{Y} is the set of misclassified samples.
- $\nabla J_p = \sum_{\mathcal{Y}} (-\mathbf{y})$.
- Gradient descent rule: $\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathcal{Y}} \mathbf{y}$
- **Batch learning:** Adjust weights after processing all inputs.
- **Sequential learning:** Adjust weights after each sample is encountered.

Geometric Interpretation

- In the solution (or dual) space, y_i is a hyperplane and the current $a(k)$ is a point.
- If y_i is misclassified, a is on the wrong side of hyperplane. Vector y_i points from a towards the hyperplane.
- Move closer to the hyperplane y_i by adding y_i to a .
- Previously correct samples can now be misclassified.
- Sequential learning: a zig-zag path to solution space.
- Batch learning: a smooth, “averaged” path.

Proof of Convergence

- For sequential learning for a linear separable case. We show the updated weights are closer to a solution vector.
- Let $\hat{\mathbf{a}}$ be a solution vector. Thus, $\hat{\mathbf{a}}^\top \mathbf{y}_i > 0, \forall i$.
- \mathbf{y}^k is the misclassified sample at step k . $\mathbf{a}(k)^\top \mathbf{y}^k < 0$.
We have: $\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k$ with simple $\eta = 1$.
- Consider: $E(k+1) = \|\mathbf{a}(k+1) - \alpha \hat{\mathbf{a}}\|^2 = (\mathbf{a}(k) - \alpha \hat{\mathbf{a}} + \mathbf{y}^k)^2$
with $\alpha > 0$.

$$\begin{aligned}
E(k+1) &= \|\mathbf{a}(k) - \alpha \hat{\mathbf{a}}\|^2 + 2(\mathbf{a}(k) - \alpha \hat{\mathbf{a}})^\top \mathbf{y}^k + \|\mathbf{y}^k\|^2 \\
&= E(k) + 2\mathbf{a}(k)^\top \mathbf{y}^k - 2\alpha \hat{\mathbf{a}}^\top \mathbf{y}^k + \|\mathbf{y}^k\|^2 \\
&\leq E(k) - 2\alpha\gamma + \beta^2
\end{aligned}$$

where $\beta^2 = \max_i \|\mathbf{y}_i\|^2$, $\gamma = \min_i (\hat{\mathbf{a}}^\top \mathbf{y}_i) > 0$, since $\mathbf{a}(k)^\top \mathbf{y}^k < 0$.

- Choosing $\alpha = \beta^2/\gamma$, we get $E(k+1) \leq E(k) - \beta^2$.
- Weights get closer to a solution vector with each iteration.
- After k steps, $E(k+1) \leq E(1) - k\beta^2$. Since $E(k+1)$ can't be negative, maximum for $k = k_0 = E(1)/\beta^2$.

- If $\mathbf{a}(1) = \mathbf{0}$, $k_0 = \frac{\alpha^2 \hat{\mathbf{a}}^2}{\beta^2} = \frac{\beta^2 \hat{\mathbf{a}}^2}{\gamma^2} = \frac{\|\hat{\mathbf{a}}\|^2 \max_i \|\mathbf{y}_i\|^2}{\min_i \|\hat{\mathbf{a}}^\top \mathbf{y}_i\|^2}$.
- Bounds the number of iterations, but in terms of unknown solution vector.
- The above is true for all $\hat{\mathbf{a}}$. The solution vector with a small margin (i.e., small γ , some points nearly on the hyperplane) will determine the convergence rate.
- Algorithms: **Batch Perceptron** learning, **Single Sample Perceptron** learning, **Variable-Increment Perceptron with Margin** learning, **Batch Variable-Increment Perceptron** learning, etc.

Relaxation Algorithm

- $J_q(\mathbf{a}) = \sum_{\mathcal{Y}} (\mathbf{a}^\top \mathbf{y})^2$.
- J_p is piecewise linear. J_q is differentiable everywhere.
But, the solution can **settle on the border**.
The solution is **influenced by large samples**.
- Add a bias term and normalize: $J_r(\mathbf{a}) = \frac{1}{2} \sum_{\mathcal{Y}} \frac{(\mathbf{a}^\top \mathbf{y} - b)^2}{\mathbf{y}^\top \mathbf{y}}$.
- Gradient $\nabla J_r = \sum_{\mathcal{Y}} \frac{(\mathbf{a}^\top \mathbf{y} - b)}{\mathbf{y}^\top \mathbf{y}} \mathbf{y}$.
- Use gradient descent using this and a learning rate $\eta(k)$.
- Single sample descent: Adjust weights for each misclassified sample. $\mathbf{a}(k+1) = \mathbf{a}(k) + \eta \frac{b - \mathbf{a}^\top(k) \mathbf{y}^k}{\|\mathbf{y}^k\|^2} \mathbf{y}^k$.

- $\frac{b - \mathbf{a}^\top(k)\mathbf{y}^k}{\|\mathbf{y}^k\|}$ is the distance from the point \mathbf{a} to the hyperplane \mathbf{y} . $\frac{\mathbf{y}^k}{\|\mathbf{y}^k\|}$ is the unit vector in that direction.
- If $\eta = 1$, the solution vector is moved to the hyperplane.
- $\mathbf{a}^\top(k+1)\mathbf{y}^k - b = (1 - \eta)(\mathbf{a}^\top(k)\mathbf{y}^k - b)$. If $\eta < 1$, still not classified correctly, etc. *Underrelaxed* if $\eta < 1$, *overrelaxed* if $1 < \eta < 2$.
- Can show that if the samples are linearly separable, $\mathbf{a}(k)$ will settle at a point on the decision boundary $\mathbf{a}^\top\mathbf{y} = b$.

Nonseparable Conditions

- What happens if the samples are **not** linearly separable?
- The perceptron criterion function and the relaxation procedures are *error-correcting*. Weights adjusted **only when there are misclassified points!**.
- When samples are not separable, the procedure may never terminate.
- Any set of $d - 1$ points are linearly separable! Large sets of points are almost never linearly separable.
- Can we make **all samples** count towards the learning?

Minimum Squared Error Procedure

- To find \mathbf{a} such that: $\mathbf{x}_i^T \mathbf{a} \geq 0, \quad i = 1, 2, \dots, n.$
- Adding a margin, the following solution works.
 $\mathbf{x}_i^T \mathbf{a} = b_i$ if $b_i > 0 \quad i = 1, 2, \dots, n.$
- Combine to write: $\mathbf{Y}\mathbf{a} = \mathbf{b}$ where \mathbf{Y} has \mathbf{x}_i^T as its rows and \mathbf{b} has b_i as the i 'th component.
- A linear set of equations. Overconstrained since $n > d.$
- Minimum squared error or least squared error solution:
Find \mathbf{a} that minimizes $J_s = \|\mathbf{Y}\mathbf{a} - \mathbf{b}\|^2 = \sum_i (\mathbf{x}_i^T \mathbf{a} - b_i)^2.$

MSE Procedure

- $\nabla_{\mathbf{a}} J_s(\mathbf{a}) = 2\mathbf{Y}^T(\mathbf{Y}\mathbf{a} - \mathbf{b}) = \mathbf{0}$.
- $\mathbf{Y}^T\mathbf{Y}\mathbf{a} = \mathbf{Y}^T\mathbf{b}$. Thus, $\mathbf{a} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{b} = \mathbf{Y}^\dagger\mathbf{b}$.
- $(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T = \mathbf{Y}^\dagger$ is called the **pseudoinverse** of the $n \times d$ matrix \mathbf{Y} . $\mathbf{Y}^\dagger\mathbf{Y} = \mathbf{I}$ but $\mathbf{Y}\mathbf{Y}^\dagger \neq \mathbf{I}$ in general.
- If \mathbf{Y} is square and nonsingular, $\mathbf{a} = \mathbf{Y}^{-1}\mathbf{b}$.
- We can choose \mathbf{b} to get a reasonable solution.
Separable solution may not exist for arbitrary \mathbf{b} .

MSE Procedure

- $J_s = \|\mathbf{Y}\mathbf{a} - \mathbf{b}\|^2$. $\nabla_{\mathbf{a}} J_s(\mathbf{a}) = 2\mathbf{Y}^T(\mathbf{Y}\mathbf{a} - \mathbf{b}) = \mathbf{0}$.
- $\mathbf{a} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{b} = \mathbf{Y}^\dagger\mathbf{b}$.
- MSE solution always exists. $\mathbf{Y}^\dagger\mathbf{b} = \lim_{\epsilon \rightarrow 0} (\mathbf{Y}^T\mathbf{Y} + \epsilon\mathbf{I})^{-1}\mathbf{Y}^T\mathbf{b}$ converges to an MSE solution.
- We can choose \mathbf{b} to get a reasonable solution. Separable solution may not be found for arbitrary \mathbf{b} even for linearly separable situation.
- The MSE or Least-Squares hold important properties, however.

MSE and Fisher's Discriminant

- If \mathbf{X}_1 are the samples labelled ω_1 and \mathbf{X}_2 are the samples labelled ω_2 , use:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{1}_1 & \mathbf{X}_1 \\ -\mathbf{1}_2 & -\mathbf{X}_2 \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{n}{n_1} \mathbf{1}_1 \\ \frac{n}{n_2} \mathbf{1}_2 \end{bmatrix}$$

- Write $\mathbf{Y}^T \mathbf{Y} \mathbf{a} = \mathbf{Y}^T \mathbf{b}$. Expand using means $\mu_i = \frac{1}{n_i} \sum_{D_i} \mathbf{x}$, $\mu = \frac{1}{n} \sum \mathbf{x}$ and $\mathbf{S}_w = \sum_i \sum_{D_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T$.
- We get $w_0 = -\mu^T \mathbf{w}$ and $\left[\frac{1}{n} \mathbf{S}_w + \frac{n_1 n_2}{n} (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \right] \mathbf{w} = (\mu_1 - \mu_2)$.
- Solution: $\mathbf{w} = \alpha n \mathbf{S}_w^{-1} (\mu_1 - \mu_2)$, Fisher Discriminant!

MSE and Bayes Classifier

- With $\mathbf{b} = \mathbf{1}_n$, minimum MSE error corresponds to the minimum mean squared error between $\mathbf{a}^\top \mathbf{y}$ and the Bayes optimal discriminant $g(\mathbf{x})$.
- Thus, MSE asymptotically approximates the Bayes optimal discriminant $g(\mathbf{x}) = P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x})$.

MSE and Bayes Classifier

- Use $\mathbf{b} = \mathbf{1}_n$ for MSE error procedure. If $g_0(\mathbf{x}) = P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x})$ is the Bayesian decision boundary, write error between the two boundaries as:

$$\epsilon^2 = \int (\mathbf{a}^\top \mathbf{y} - g_0(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x}$$

- $J(\mathbf{a}) = \sum_{\omega_1} (\mathbf{a}^\top \mathbf{y} - 1)^2 + \sum_{\omega_2} (\mathbf{a}^\top \mathbf{y} + 1)^2$
 $= n \left(\frac{n_1}{n} \frac{1}{n_1} \sum_{\omega_1} (\mathbf{a}^\top \mathbf{y} - 1)^2 + \frac{n_2}{n} \frac{1}{n_2} \sum_{\omega_2} (\mathbf{a}^\top \mathbf{y} + 1)^2 \right)$

- As $n \rightarrow \infty$, using conditional expectations E_{ω_1}

$$\bar{J}(\mathbf{a}) = \frac{J(\mathbf{a})}{n} = P(\omega_1) E_{\omega_1} (\mathbf{a}^\top \mathbf{y} - 1)^2 + P(\omega_2) E_{\omega_2} (\mathbf{a}^\top \mathbf{y} - 1)^2$$

- It can be shown that:

$$\bar{J}(\mathbf{a}) = \int (\mathbf{a}^T \mathbf{y} - g_0(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x} + \left[1 - \int g_0^2(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \right]$$

- Second term is independent of \mathbf{a} . So, the error between the two decision boundaries is minimized when MSE is minimized!
- Discriminant function $\mathbf{a}^T \mathbf{y}$ gives direct information about the posteriors! The boundaries of $g(\mathbf{x})$ and $g_0(\mathbf{x})$ match wherever data exists.

Least Mean Squared Procedure

- Pseudoinverse is cumbersome and time-consuming.
- Use gradient descent to minimize $J_s(\mathbf{a})$:
 1. Initialize weights $\mathbf{a}(1)$ to arbitrary values.
 2. Update using: $\mathbf{a}(k + 1) = \mathbf{a}(k) - \eta(k)\mathbf{Y}^\top(\mathbf{Y}\mathbf{a}(k) - \mathbf{b})$.
- Perform this sample by sample: **Widrow-Hoff** or **Least Mean Squared (LMS)** procedure.
- Updation: $\mathbf{a}(k + 1) = \mathbf{a}(k) + \eta(k)(b - \mathbf{a}^\top(k)\mathbf{y}^k)\mathbf{y}^k$.
Repeat the samples till correction is small.
- May not stop at a separating hyperplane even if it exists.

Varying Margin Vector

- There exists a $\mathbf{b} = \hat{\mathbf{b}}$ for which MSE works if the samples are separable.
- Treat \mathbf{a} , \mathbf{b} as variable and optimize over them!
- $J_s(\mathbf{a}, \mathbf{b}) = \|\mathbf{Y}\mathbf{a} - \mathbf{b}\|^2$, $\mathbf{b} > 0$.
 $\nabla_{\mathbf{a}} J_s = 2\mathbf{Y}^T(\mathbf{Y}\mathbf{a} - \mathbf{b})$ and $\nabla_{\mathbf{b}} J_s = -2(\mathbf{Y}\mathbf{a} - \mathbf{b})$
- $\mathbf{a} = \mathbf{Y}^\dagger \mathbf{b}$. Use gradient descent for \mathbf{b} with $\mathbf{e} = \mathbf{Y}\mathbf{a} - \mathbf{b}$.
- Start with $\mathbf{b} > \mathbf{0}$ and do **not** reduce its components!

- Use $\mathbf{e}^+ = \mathbf{e} + |\mathbf{e}|$ where $|e|_i = \text{abs}(e_i)$, a positive vector.
- **Ho-Kashyap Procedure:**
 1. Start with $\mathbf{b} > \mathbf{0}$, $\mathbf{a} = \mathbf{Y}^\dagger \mathbf{b}$.
 2. $\mathbf{e} = \mathbf{Y}\mathbf{a} - \mathbf{b}$, $\mathbf{e}^+ = \mathbf{e} + |\mathbf{e}|$, $\mathbf{b} = \mathbf{b} + \eta(\mathbf{k})\mathbf{e}^+$, $\mathbf{a} = \mathbf{Y}^\dagger \mathbf{b}$
 Until $|\mathbf{e}| < \text{threshold}$ or maximum iterations reached.
- If all components of \mathbf{e}^+ are 0, no more corrections. If $\mathbf{e} \neq \mathbf{0}$, the samples are not linearly separable.
- A modification is to use: $\mathbf{a}(k+1) = \mathbf{a}(k) + \eta \mathbf{Y}^\dagger |\mathbf{e}(k)|$

Extension to Multiple Categories

- Define $g_i(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_{i0} = \mathbf{a}_i^T \mathbf{y}$, $i = 1, \dots, c$.
- Vector \mathbf{x} is assigned to ω_i if $g_i(\mathbf{x})$ is greater than $g_j(\mathbf{x})$ for all j .
- Decide ω_i if $\mathbf{a}_i^T \mathbf{y} > \mathbf{a}_j^T \mathbf{y}$, $\forall j \neq i$.

Kesler's Construction

- Convert it to a 2 class problem.
Write $\mathbf{a}_i^\top \mathbf{y} - \mathbf{a}_j^\top \mathbf{y} > 0$, $j = 2, \dots, c$.
- Define a (cd) -dimensional vector: $\alpha = [\mathbf{a}_1^\top \ \mathbf{a}_2^\top \ \dots \ \mathbf{a}_c^\top]^\top$ and (cd) -dimensional samples:
 $\eta_{ij} = [\mathbf{0}^\top \ \dots \ \mathbf{y}^\top \ \dots \ -\mathbf{y}^\top \ \dots \ \mathbf{0}^\top]^\top$ with a \mathbf{y} at the i th position and $-\mathbf{y}$ at the j th position.
- Now a linear problem: $\alpha^\top \eta_{ij} > 0$, for all $j \neq i$.
- Create η_{ij} for each $\mathbf{y} \in \omega_i$. Number of samples: $(c - 1)n$.
- Dimensionality: cd .

Error-Correction Rule

- Objective: Achieve $\alpha^T \eta_{ij} \geq 0$.
- Using the Perceptron criterion function:
 $\alpha(k+1) = \alpha(k) + \eta_{ij}^k$ for a misclassified vector η^k .
- $\mathbf{y}^k \in \mathcal{Y}_i$ requires correction. $\mathbf{a}_i^T \mathbf{y}^k \leq \mathbf{a}_j^T \mathbf{y}^k$ for some j .
- This translates to the Fixed Increment rule:
$$\begin{aligned} \mathbf{a}_i(k+1) &= \mathbf{a}_i(k) + \mathbf{y}^k && \text{for } i, \\ \mathbf{a}_j(k+1) &= \mathbf{a}_j(k) - \mathbf{y}^k && \text{for } j, \\ \mathbf{a}_l(k+1) &= \mathbf{a}_l(k) && \text{for others.} \end{aligned}$$

Generalization of MSE

- Kesler construction generalizes error-correction rules well.
- For MSE, let us assume $\mathbf{a}_i^T \mathbf{y}$ to be an estimate of the posterior probability $P(\omega_i | \mathbf{x})$.
- Thus, $\mathbf{a}_i^T \mathbf{y} = 1$ if $\mathbf{y} \in \mathcal{Y}_i$ and 0 otherwise.
- Let $\mathbf{Y} = [\mathbf{Y}_1^T \quad \mathbf{Y}_2^T \quad \cdots \quad \mathbf{Y}_c^T]^T$ be the partitioned training samples and $\mathbf{A} = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_c]$ the weights.

- $\mathbf{YA} = \mathbf{B} = [\mathbf{B}_1^\top \ \cdots \ \mathbf{B}_c^\top]^\top$ where \mathbf{B}_i has its i th row all 1's and rest 0's.
- The squared “error” $(\mathbf{YA} - \mathbf{B})^\top(\mathbf{YA} - \mathbf{B})$ is minimized when $\mathbf{A} = \mathbf{Y}^+\mathbf{B}$.
- If \mathbf{B}_i contains the risks λ_{ij} , the mean-squared-error approximation to the minimum risk solution!

Linear Discriminants: Summary

Criterion Function	Method	Remarks
Perceptron	Gradient Descent	Error Correcting
Perceptron error	Gradient Descent	Error Correcting
Sum of Squared Error (SSE)	Using pseudoinverse	Minimum Squared Error
SSE	Gradient Descent	Least Mean Sq.
SSE	Gradient Descent	Ho-Kashyap