Music and Machine Learning (IFT6080 Winter 08)
Prof. Douglas Eck, Université de Montréal

These slides follow closely the (English) course textbook
Pattern Recognition and Machine Learning
by Christopher Bishop

Linear Models for Classification

- Goal: take input vector x and map it onto one of K discrete classes
- Input space divided into *decision regions* divided by *decision surfaces* or *decision boundaries*
- Consider *linear models*: those separable by (D-1) dimensional hyperplanes in the D-dimensional input space.
- Perfect separation: *linearly-separable*
- Recall simplest linear regression model: $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$
- Use *activation function f(·)* to map function onto discrete classes

$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0\right)$$

- Due to *f(·)* these models are no longer linear in the parameters

Discriminant Functions

- 2 class case, simplest: $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$
  where **w** is a weight vector and w_0 is the bias

- Decision boundary is 0.

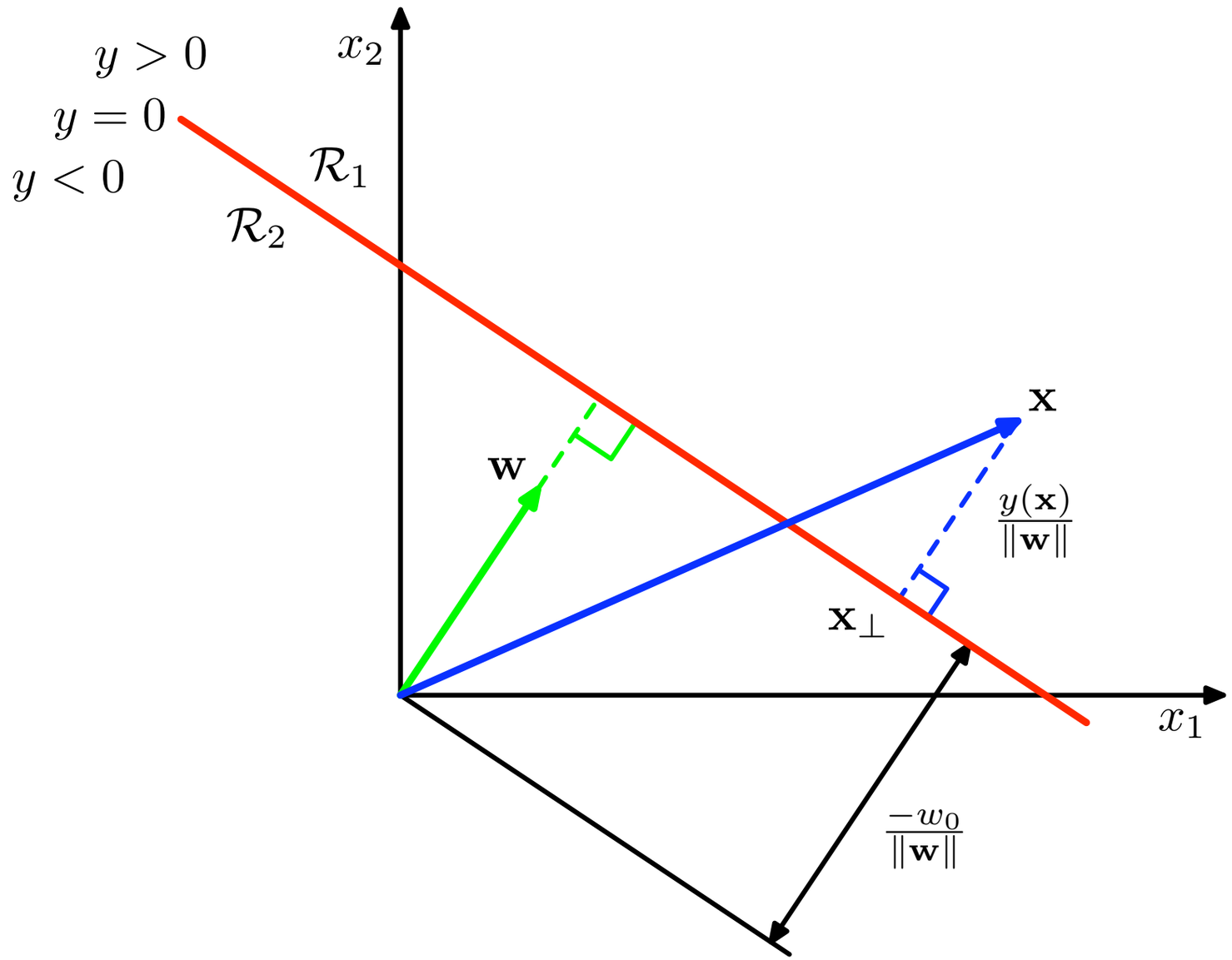
- Consider 2 points x_a and x_a lying on decision surface. Because $y(x_a) = y(x_b) = 0$ we have $\mathbf{w}^{\mathrm{T}}(\mathbf{x}_A - \mathbf{x}_B) = 0$
  thus vector w is orthogonal to every vector lying within decision surface

- If x is on decision surface then y(x)=0 indicating that:

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

  We thus see that bias w_0 decides location of decision surface

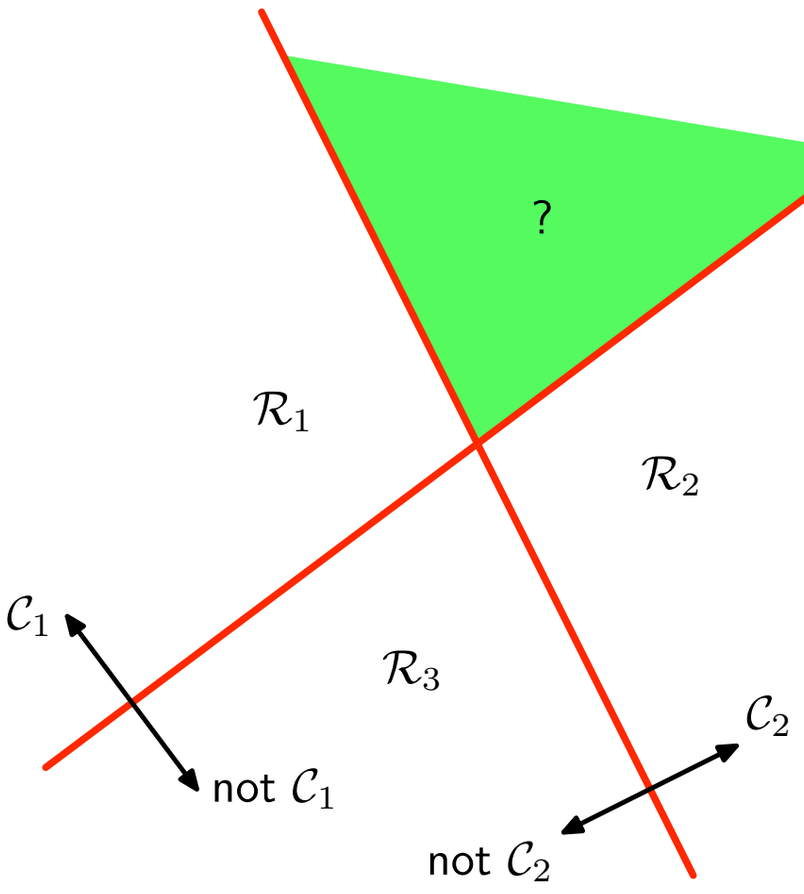
Geometry of linear discriminant

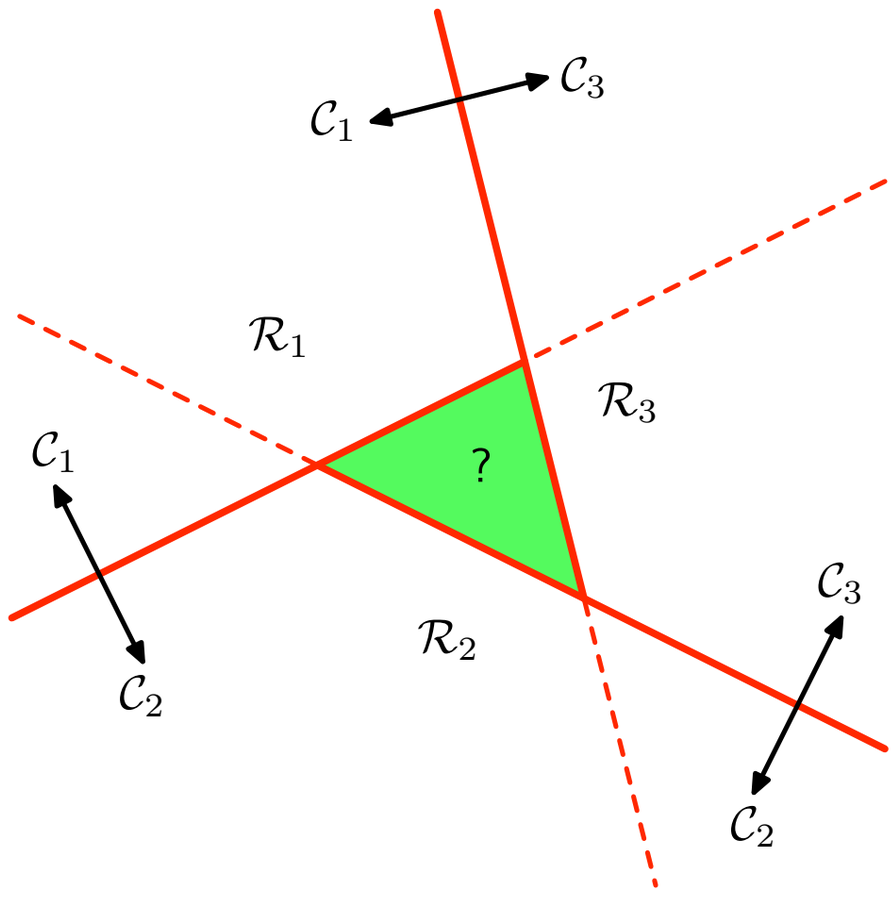


Decision surface is perpendicular to w. Displacement controlled by w_0.

Multiple Classes

- Not generally good idea to use multiple 2-class classifiers to do K-class classification
- Leads to ambiguous regions. See figure:





Left: distinguish points in C_k from those not in C_k. Right: pairwise separation of classes C_k and C_j. Both lead to ambiguous regions.

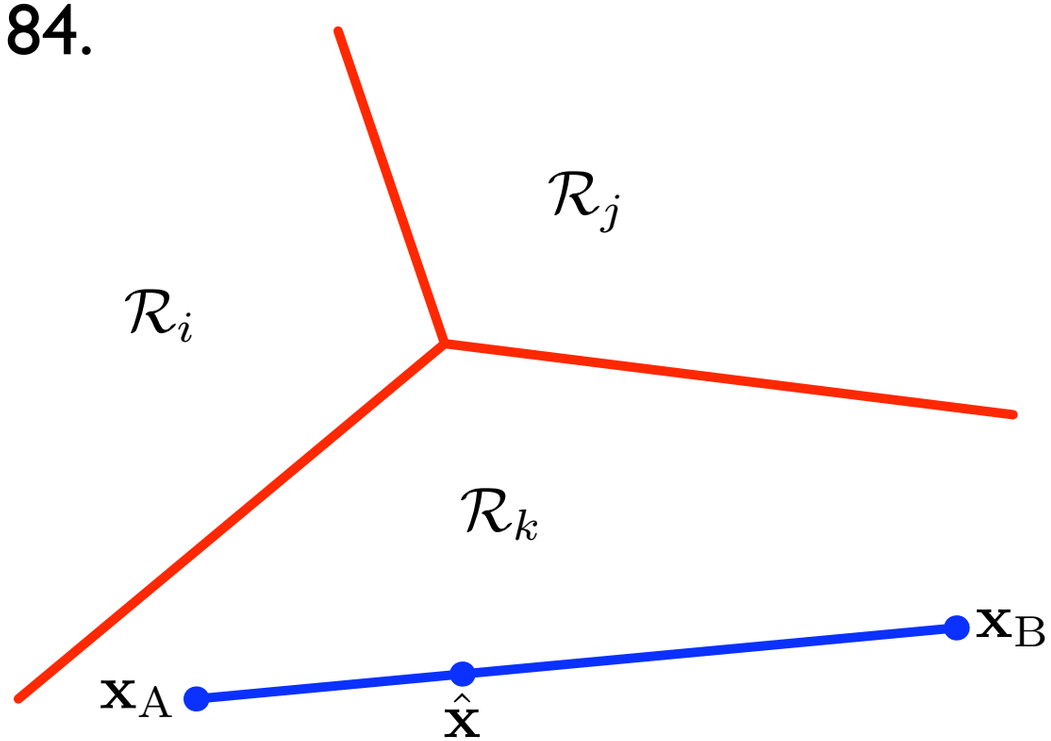
Single K-class classifier

- Single discriminant comprising K linear functions of form
$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$

- Point x belongs in class C_k if $y_k(x) > y_j(x)$ for all $j \neq k$

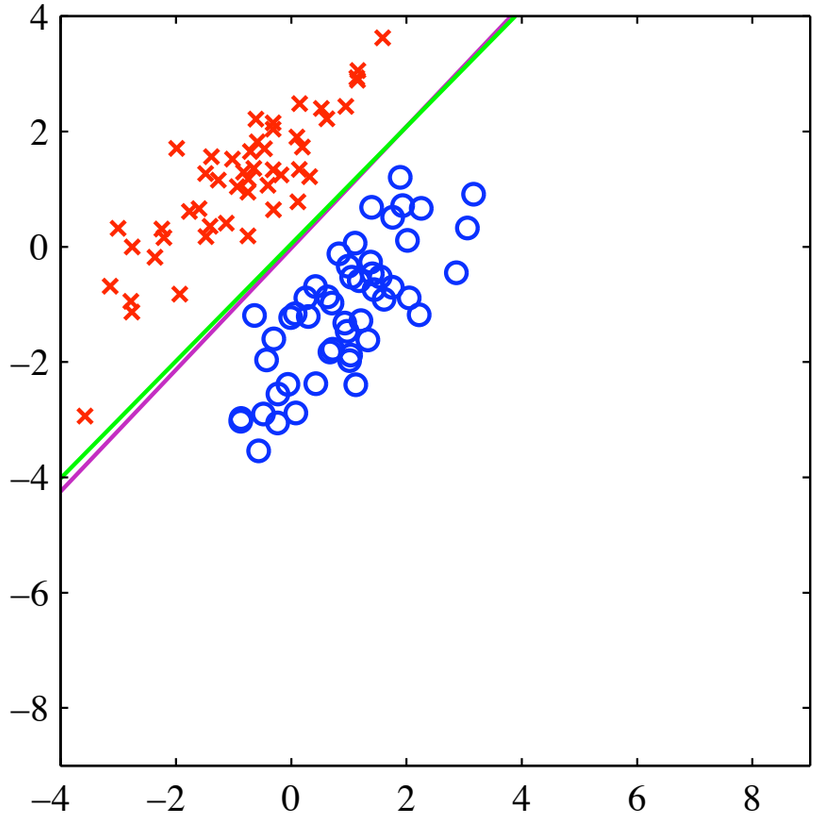
- Decision boundary between C_k and C_j is had by $y_k(x) = y_j(x)$ and corresponds to (D-1)-dimensional hyperplane $(\mathbf{w}_k - \mathbf{w}_j)^{\mathrm{T}}\mathbf{x} + (w_{k0} - w_{j0}) = 0$

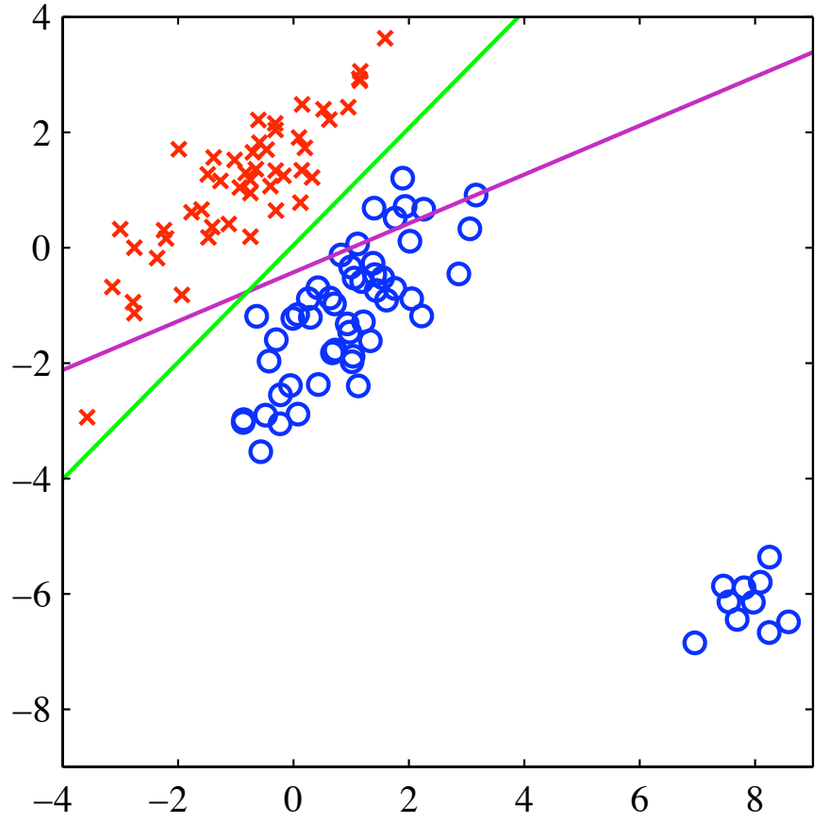
- Decision region singly connected and convex (due to linearity of discriminant functions. See Bishop pg 184.



Least Squares Classification

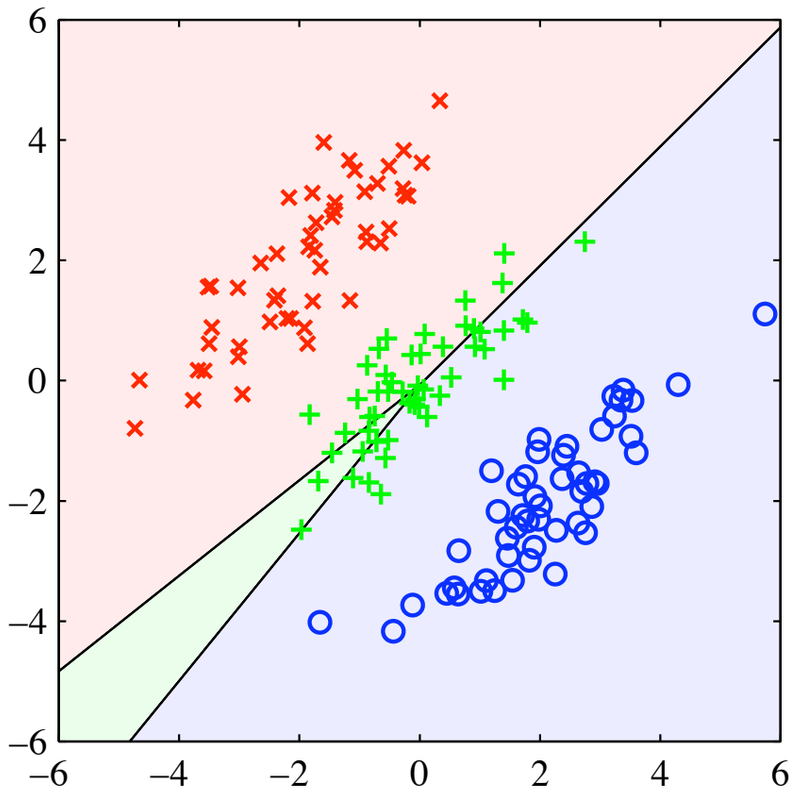
- Approximates conditional expectation E[t|x]
- Each class C_k described by its own linear model $y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$
- Minimize sum-of-squared errors on set of models
- Works very poorly. E.g. very sensitive compared to logistic regression to outliers.

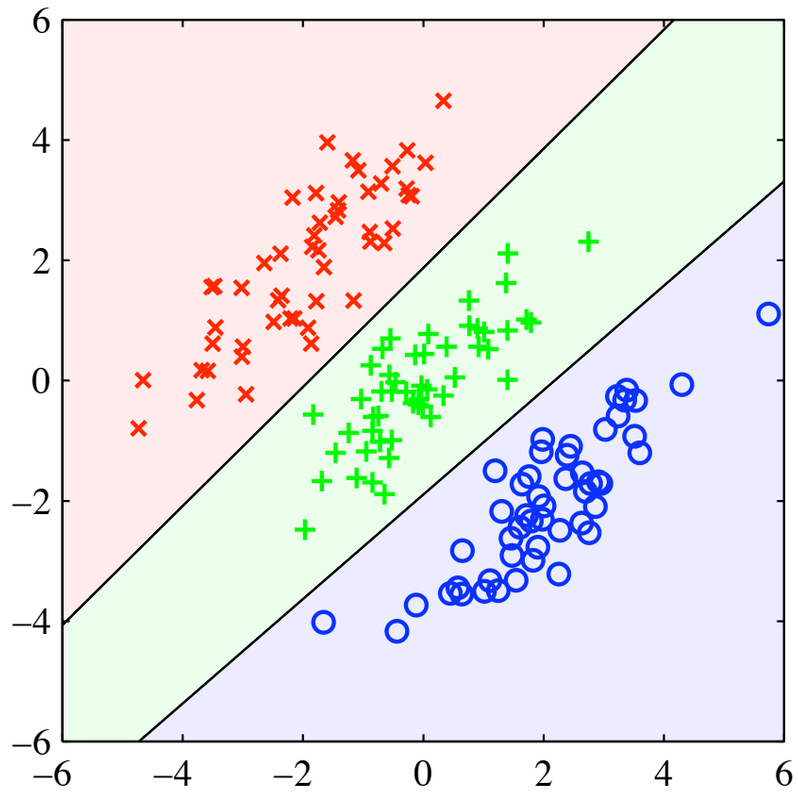




Least Squares Classification

- Second example: even for easy dataset with clear decision boundaries one class is underrepresented (compare to logistic regression below)
- Behavior not surprising when we consider that it corresponds to ML under assumption of Gaussian conditional distribution. Binary target vectors are far from Gaussian.





Fisher's linear discriminant

- Can look at classification as dimensionality reduction: take D dimensional input vector and project it down to 1 dimension using:
$y = \mathbf{w}^{\mathrm{T}}\mathbf{x}$

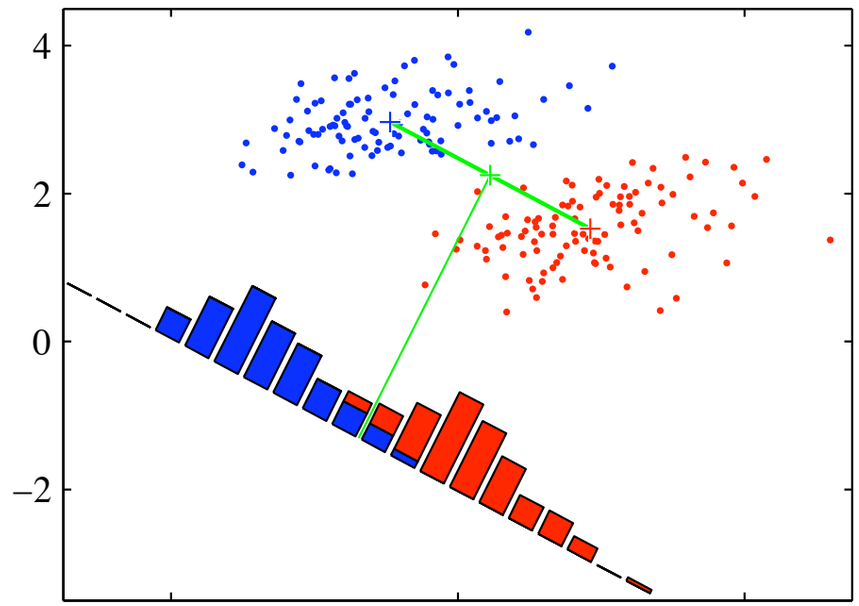
- Use threshold such that $y \geq w_0$ is classified as C_0 (Yields standard linear classifier from previous slide).

- Adjust weight vector to project with maximum class separation.
Consider 2-class version:
$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\mathbf{x}_n \qquad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\mathbf{x}_n$$

- Simplest measure of class separation is separation of class means.
Choose w to maximize $m_2 - m_1 = \mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1)$ where
$m_k = \mathbf{w}^{\mathrm{T}}\mathbf{m}_k$ si the mean of the projected data from class C_k.

Fisher's linear discriminant

- Still problematic: when there is strongly non-diagonal covariance between class distributions, separation is poor:



- Fishers: maximize separating between class means while minimizing variance within each class. Within class variance is:

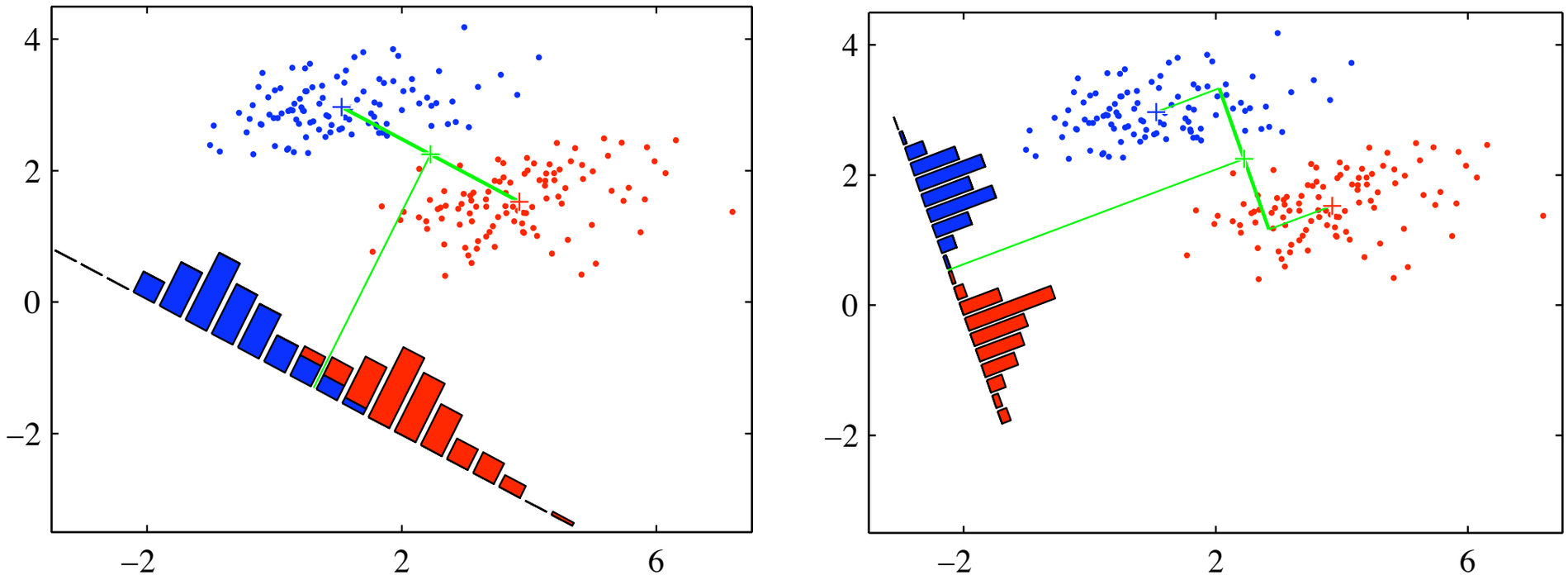
$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \text{ where } y_n = \mathbf{w}^{\mathrm{T}} \mathbf{x}_n$$

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

- Fisher criterion is ratio of the between-class mean and the within-class

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

Geometry of Fisher's



Left: samples from two classes along with histograms resulting from projection onto line joining the class means (reproduced from previous slide). Right: corresponding projection based on Fisher linear discriminant.

Fisher's linear discriminant

- Fisher's criterion: $J(\mathbf{w}) = \dfrac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$
- Can relate this to weights w by rewriting as: $J(\mathbf{w}) = \dfrac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_B\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_W\mathbf{w}}$

  where S_B is *between-class* covariance $S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}}$

  and where S_W is *within-class* covariance

  $\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}} + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^{\mathrm{T}}$

- Differentiating *J*(*w*) with respect to *w* reveals maximum when:
  $(\mathbf{w}^{\mathrm{T}}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^{\mathrm{T}}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$

- S_Bw is always in the direction of (m_2 - m_1). Don't care about magnitiude of w so drop scalar factors $(\mathbf{w}^{\mathrm{T}}\mathbf{S}_B\mathbf{w})$ and $(\mathbf{w}^{\mathrm{T}}\mathbf{S}_W\mathbf{w})$
  Multiply both sides by $\mathbf{S}_W^{-1}$ yields *Fisher's linear discriminant*

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

Fisher's linear discriminant

- Least squares approach to linear discriminant was to make model predictions as close as possible to target values
- Fisher was derived based on maximum *separation* of classes
- Fisher can still be derived as special case of least squares
- Trick is to recode target for C_1 to be N/N_1 where N_1 is the number of patterns in class C_1 and N is the total number of patterns. For C_2 target is $-N/N_2$.
- Approximates reciprocal of the prior for the class. Read 4.1.7
- Fisher's criterion can be extended to >2 classes. Read 4.1.6

Perceptron algorithm.

- Rosenblatt (1962)
- Linear model with step activation function:

$$y(\mathbf{x}) = f(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}))$$

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

- Train using *perceptron criterion*

$$E_{\mathrm{P}} = -\sum_{n \in \mathcal{M}} \mathbf{w}^{\mathrm{T}}\phi_n t_n$$

where *M* is the set of misclassified patterns. Note that direct misclassification using total number of miscalssified patterns will not work because of nonlinear f() (the gradient is ill-conditioned). Instead we seek a weight vector such that $\forall \mathbf{x}_n \in \mathcal{C}_1, \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) > 0$ and $\forall \mathbf{x}_n \in \mathcal{C}_2, \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) < 0$.

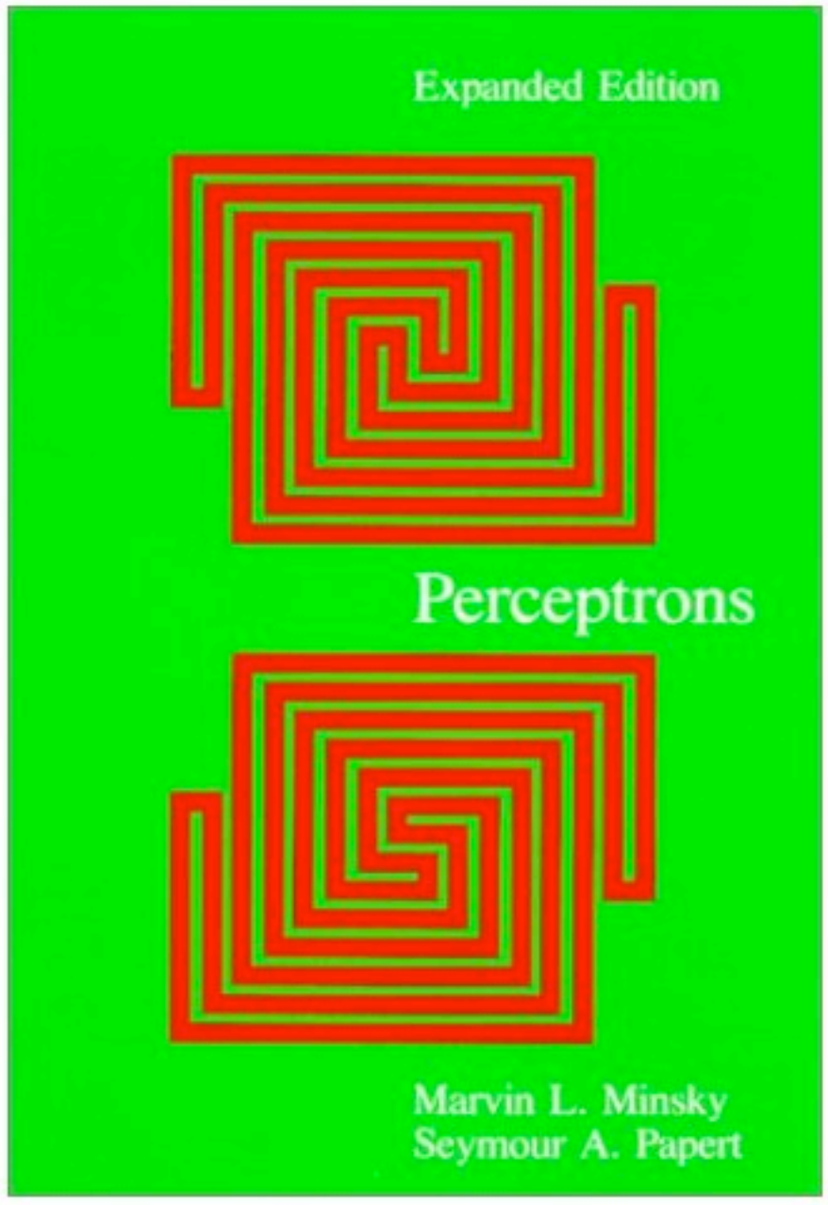
Perceptron algorithm.

- Total error function is piecewise linear (across all *misclassified* patterns). Stochastic gradient descent: $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi_n t_n$

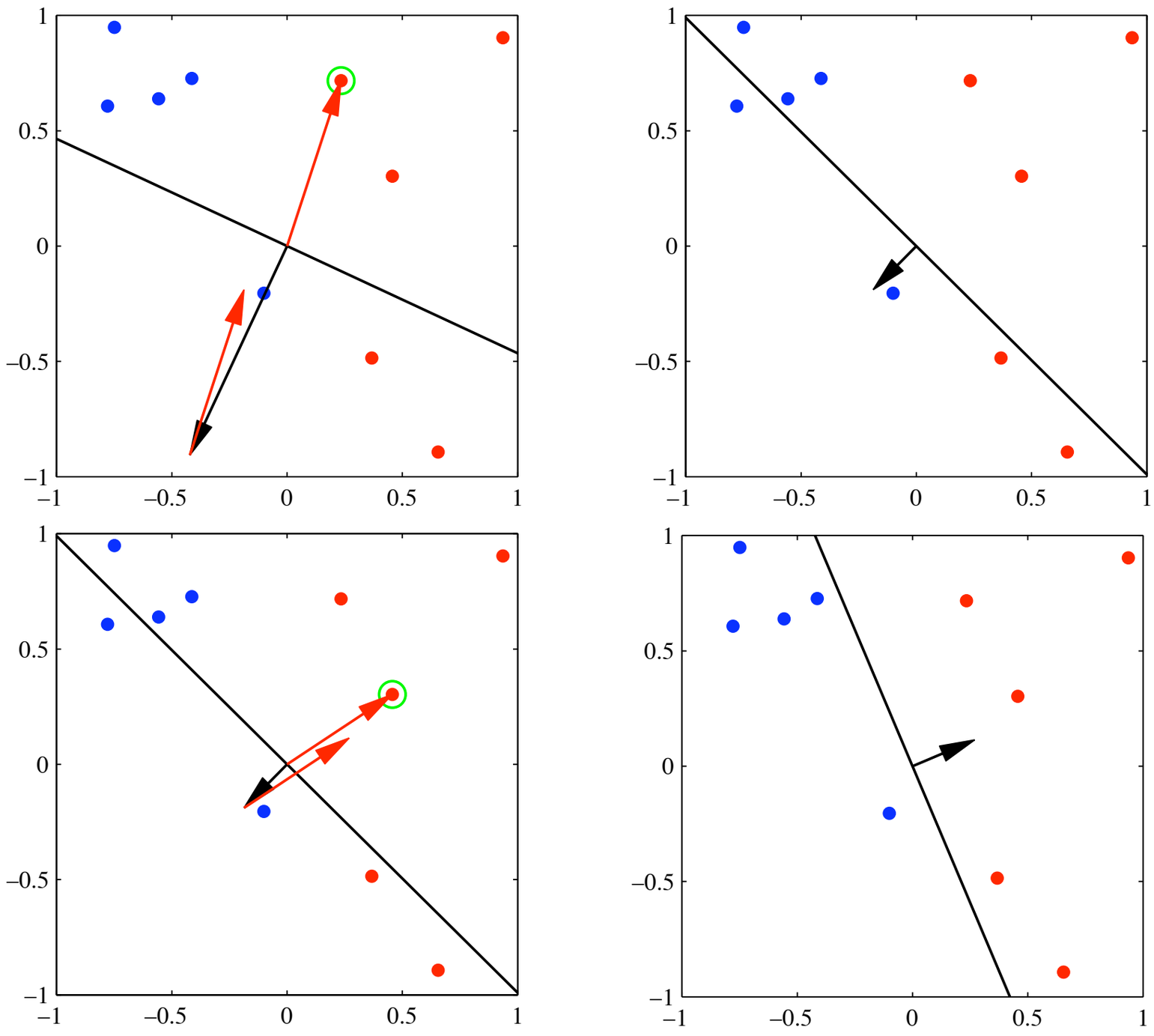
- Update is not a function of w thus η can be equal to 1

- *Perceptron convergence theorem*: If there exists an exact solution (i.e. if training set is indeed linearly separable) then PA will find a solution in finite number of steps.

- Attacked by Minsky and Papert in *Perceptrons* (1969). Attack valid only for single-layer perceptrons. Still, stopped research in neural computation for nearly a decade.



Geometry of Perceptron



Convergence for 2 classes (red and blue). Black arrow = w; black line is decision boundary. On top left, the point circled in green is misclassified thus its feature vector is added to w. This is repeated on the bottom

Probabilistic Generative Models

- Model class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$
- Posterior probability for class C_1:

$$p(\mathcal{C}_1|x) = \frac{p(x|\mathcal{C}_1)p(\mathcal{C}_1)}{p(x|\mathcal{C}_1)p(\mathcal{C}_1) + p(x|\mathcal{C}_2)p(\mathcal{C}_2)}$$
$$= \frac{1}{1+\exp(-a)} = \sigma(a)$$

where we have defined $a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$

- σ is the *logistic sigmoid;* Note that $\sigma(-a) = 1 - \sigma(a)$
  and that the inverse of σ is the *logit* function

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

representing ratio of probabilities $a = \ln[p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})]$

Probabilistic Generative Models

- Generalization to multiple classes: *normalized exponential function*

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

$$= \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$

Also known as *softmax* function because it is a smoothed version of "max".

- Different representations for class-conditional densities yield different consequences in how classification is done [see following slides...]

Continuous inputs

- First assume all classes share same covariance matrix and only 2 classes. This yields:

$$p(\mathcal{C}_1|x) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

where:

$$\begin{aligned} \mathbf{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

- Quadratic term from Gaussians vanishes. priors p(C_k) only enter via bias parameter. Thus make parallel shift in decision boundry.

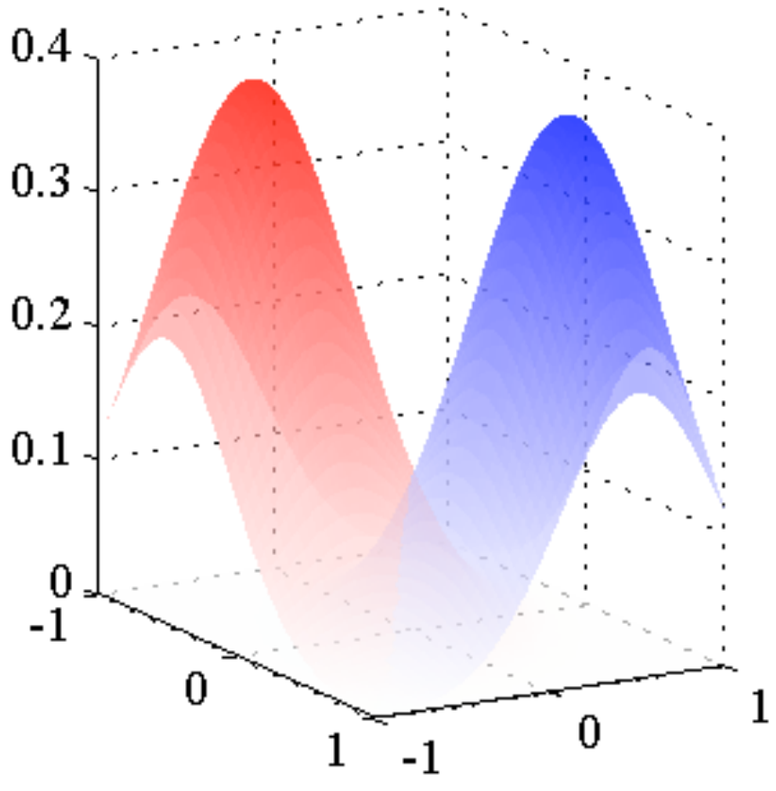
- For general case of *K* classes we have:

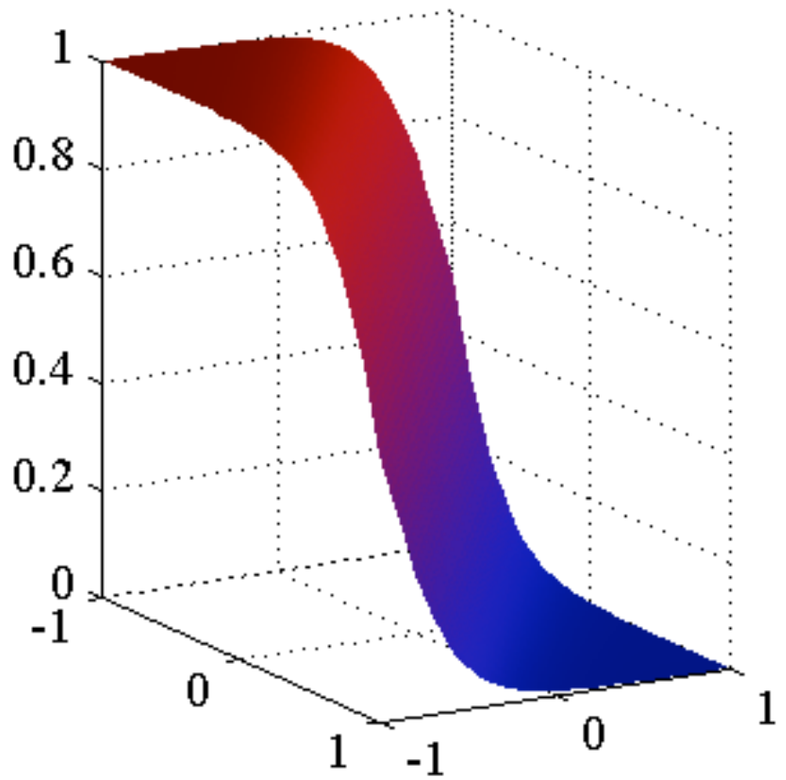
$$a_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

where:

$$\begin{aligned} \mathbf{w}_k &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k) \\ w_{k0} &= -\frac{1}{2}\boldsymbol{\mu}_k^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k) \end{aligned}$$

Continuous inputs

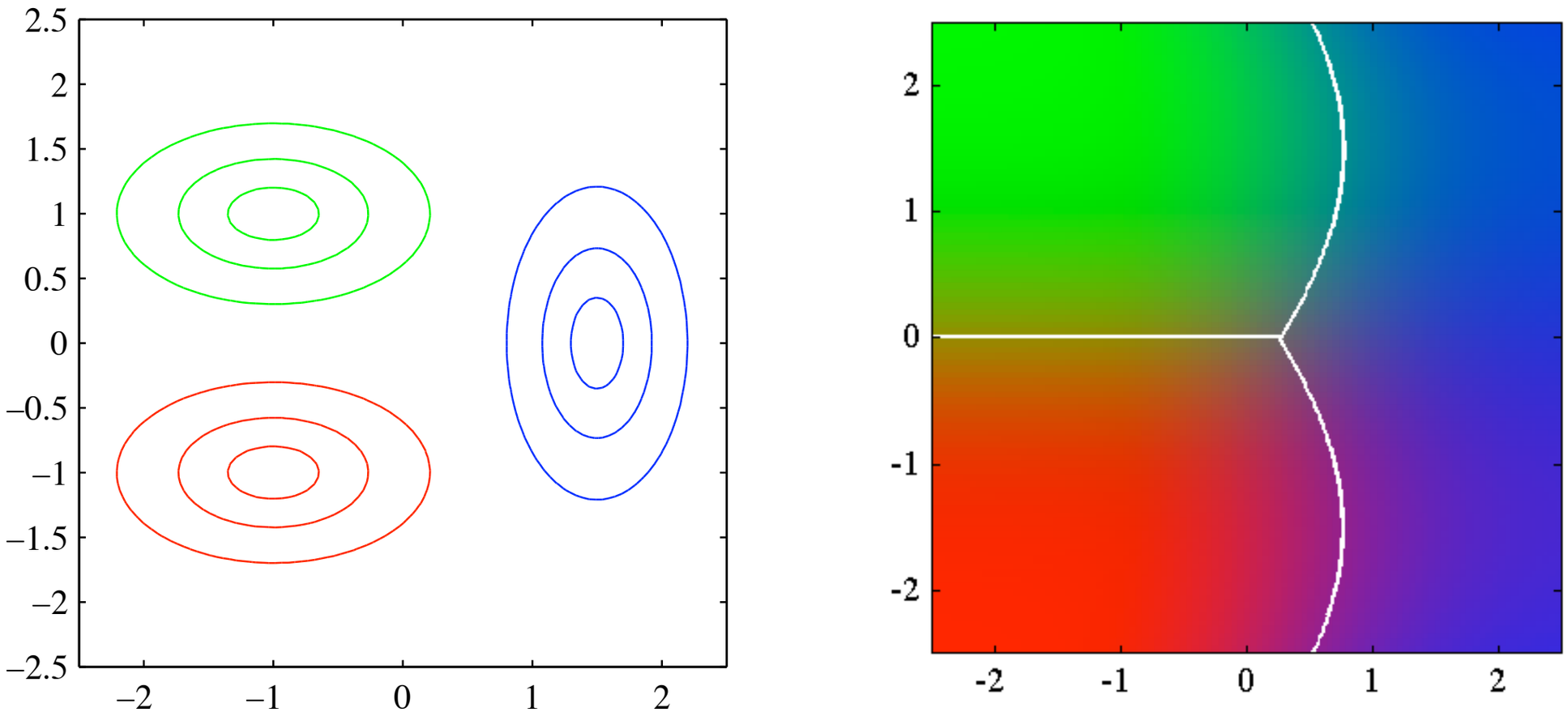




Left: class-conditional densities for two classes, red and blue. Right, corresponding posterior probability $p(C_1|x)$ which is given by a logistic sigmoid of a linear function of x. On right: proportion of red yields $p(C_1|x)$, proportion of blue yields $p(C_2|x) = 1-p(C_1|x)$

Linear versus quadratic

- When covariance is shared by classes, decision boundary is linear. When covariances are unlinked, decision boundary is quadratic. See Bishop p198-199 for details



Left: class conditional densities for three Gaussians. Green and red have same covariance matrix. Right: Decision boundary is linear where covariances are the same, quadratic where covariances differ.

Maximum likelihood

- Now have a parametric form for class-conditional densities p(x|C_k). Can now determine values of parameters and priors p(C_k).

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- Yields likelihood:

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} [\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

- Maximize log-likelihood. For π relevant terms are:

$$\sum_{n=1}^{N} \{t_n \ln \pi + (1-t_n) \ln(1-\pi)\}$$

set derivative with respect to π to 0 yields:

$$\pi = \frac{1}{N}\sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

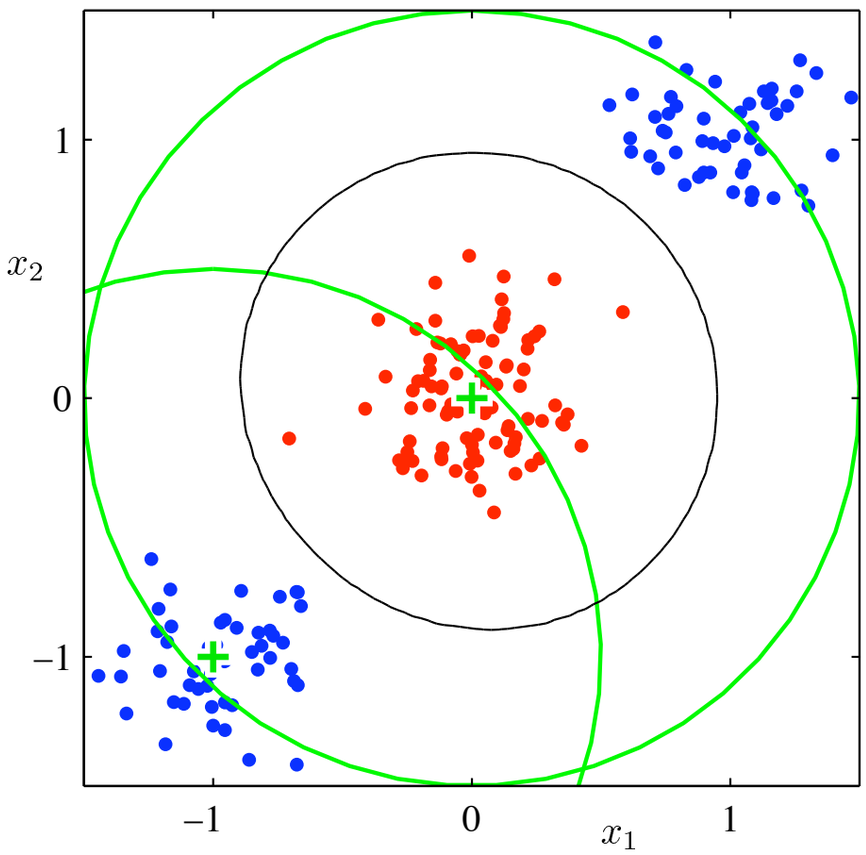
Probabilistic Discriminative Models

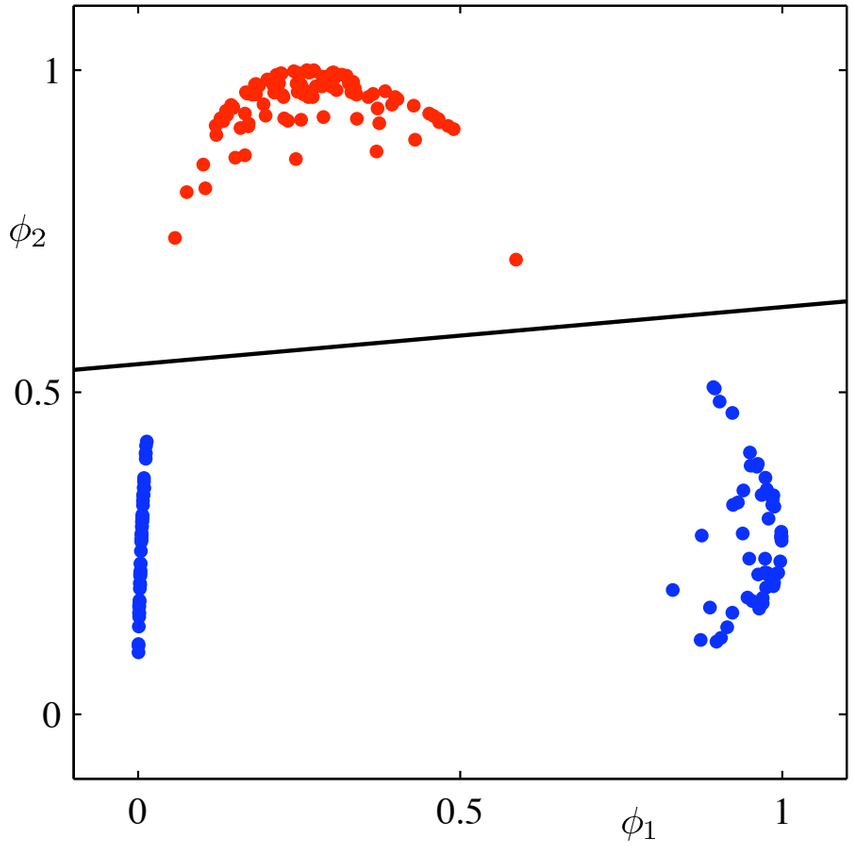
- ***Generative approach:***
  For two-class classification, posterior of C_1 is logistic sigmoid acting on linear function of x for many distributions $p(x|C_k)$. Used ML to determine parameters of densities as well as class priors $p(C_k)$. Then use Bayes to determine posterior class probabilities

- ***Discriminiative approach:***
  Learn parameters of general linear model explicitly using ML. Maximize likelihood of conditional $p(C_k|x)$ directly.

- Advantages of discriminiative approach: fewer parameters, better performance when class-conditional densities do not correspond well to true distributions

Fixed Basis Functions





Left: original data is not linearly separable. Two Gaussian basis functions ϕ_1 and ϕ_1 are defined, as shown by green lines. Right: New linearly-separable space as created by the Gaussians. Line is nonlinear in the original variables.

Logistic regression

- Posterior probability of class C_1 written as a logistic sigmoid acting on linear function of feature vector ϕ

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi})$$

where $\sigma(\cdot)$ is the logistic sigmoid

- Called *logistic regression* but really a model of classification

- More compact than ML-fitting of Gaussians: for M parameters, Gaussian model uses 2M parameters for the means and M(M + 1) /2 parameters for shared covariance matrix. Grows quadratically.

- Use ML to determine parameters. Recall derivitive of logistic sigmoid is:

$$\frac{\mathrm{d}\sigma}{\mathrm{d}a} = \sigma(1-\sigma)$$

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}\left\{1-y_n\right\}^{1-t_n}$$

Logistic regression

- Negative log of likelihood yields *cross entropy*:

$$E(\mathbf{w}) = -\ln p(\mathbf{t}, \mathbf{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1 - t_n) \ln(1 - t_n) \right\}$$

- Gradient with respect to *w* yields:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n) \boldsymbol{\phi}_n$$

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

- Takes same form as the gradient for sum-of-squares error (eqn B3.13):

$$\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$$

- However logistic sigmoid means no closed-form quadratic solution (here is sum of squared error for comparison:)

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

where Φ is the design matrix

Iterative reweighted least squares (IRLS)

- Efficient iterative optimization: *Newton-Raphson*

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mathbf{H}^{-1}\nabla E(\mathbf{w})$$

  where H is the Hessian matrix comprising the second derivatives of E(w) with respect to w.

- For sum-of-squares error this can be done in one step because the error function is quadratic (see p207)

- For cross-entropy we get a similar set of normal equations for *weighted* least squares where the weighting depends on w

- This dependancy forces us to apply the update iteratively. (see p208)

Final notes on chapter 4

- Logistic regression suffers from overfitting. ML solution corresponds to $\sigma=0.5$ which is where $w^T\phi=0$. Magnitude of w goes to infinity; slope of the sigmoid becomes infinitely steep. All positive examples take a posterior of $p(C_k|x)=1$.

- Not resolved by having more data. But is resolved using (a)regularization or (b) a Bayesian approach involving an appropriate prior over w.

- (4.3.4) *Multiclass logistic regression* is an easy extension of the two-class formulation. Replace logistic sigmoid with softmax and use maximum likelihood.

- (4.3.5) *Probit regression* is of theoretical interest but is in practice similar to logistic regression.

- (4.3.6) *Canonical link functions* provide a principled framework for matching activation functions and error functions. Based on assumptions about the conditional distribution for the target variable.

- 4.3.4-6 are not discussed further. Please read them.