Advanced Machine Learning
Lecture 4: Classification

Sandjai Bhulai
Vrije Universiteit Amsterdam

s.bhulai@vu.nl
15 September 2023

VU Vrije Universiteit Amsterdam
Faculty of Science

Linear models for classification

Advanced Machine Learning

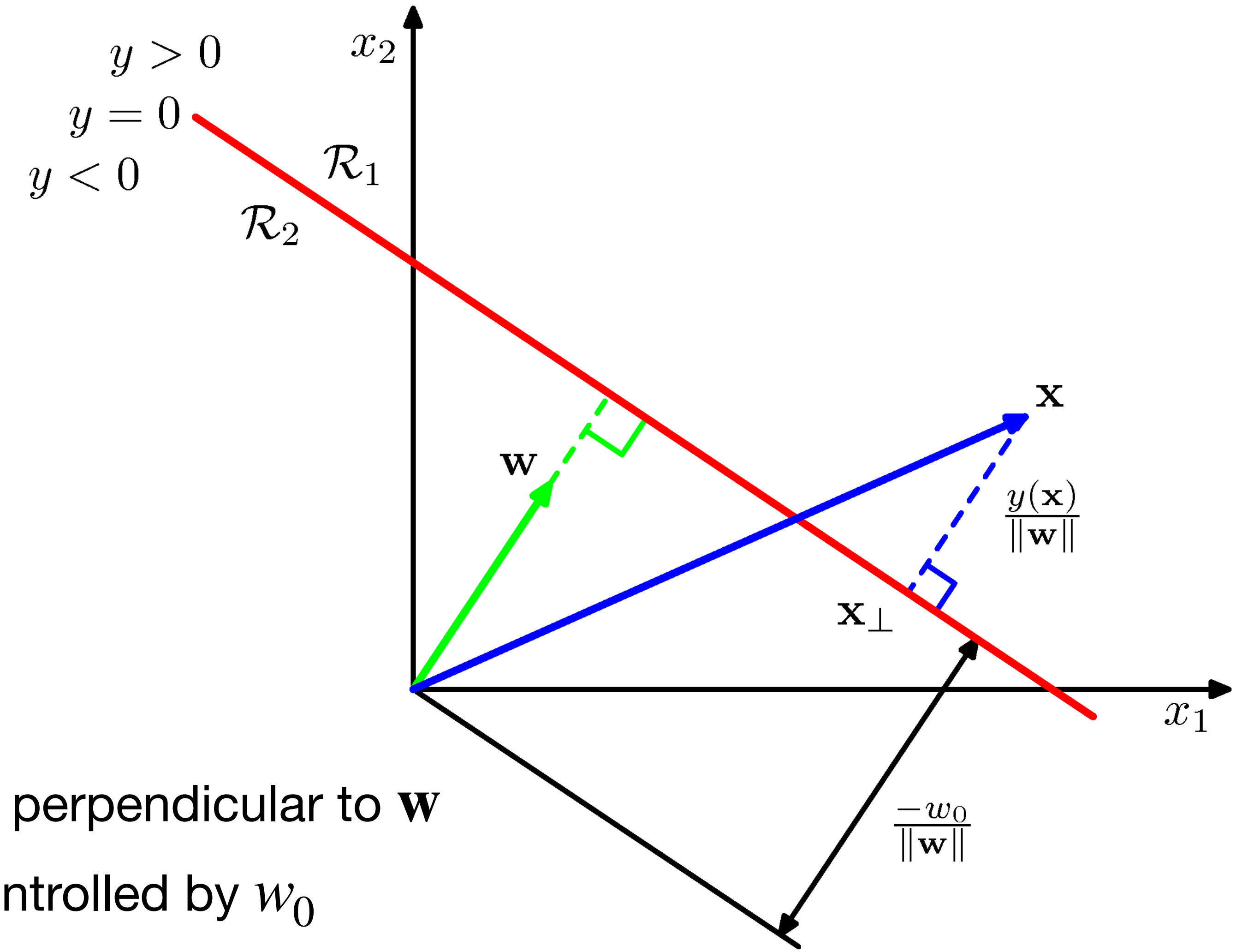
Classification with linear models

- Goal: take input vector x and map it onto one of K discrete classes
- Consider linear models: separable by $(D-1)$ dimensional hyperplanes in the D-dimensional input space
- Simplest linear regression model: $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$
- Use activation function $f(\,\cdot\,)$ to map function onto discrete classes $y(\mathbf{x}) = f(\mathbf{w}^\top \mathbf{x} + w_0)$
- Due to $f(\,\cdot\,)$, these models are no longer linear in the parameters

Discriminant functions

- The simplest case is the 2-class case: $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$, where $\mathbf{w}$ is a weight vector and w_0 is the bias
- Decision boundary is 0
- Consider 2 points $\mathbf{x}_a$ and $\mathbf{x}_b$ lie on the decision surface. Because $y(\mathbf{x}_a) = y(\mathbf{x}_b) = 0$, we have $\mathbf{w}^\top(\mathbf{x}_a - \mathbf{x}_b) = 0$.
- Thus, vector $\mathbf{w}$ is orthogonal to every vector lying within the decision surface
- If $\mathbf{x}$ is on the decision surface, then $y(\mathbf{x}) = 0$, indicating that $\dfrac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|} = -\dfrac{w_0}{\|w\|}$

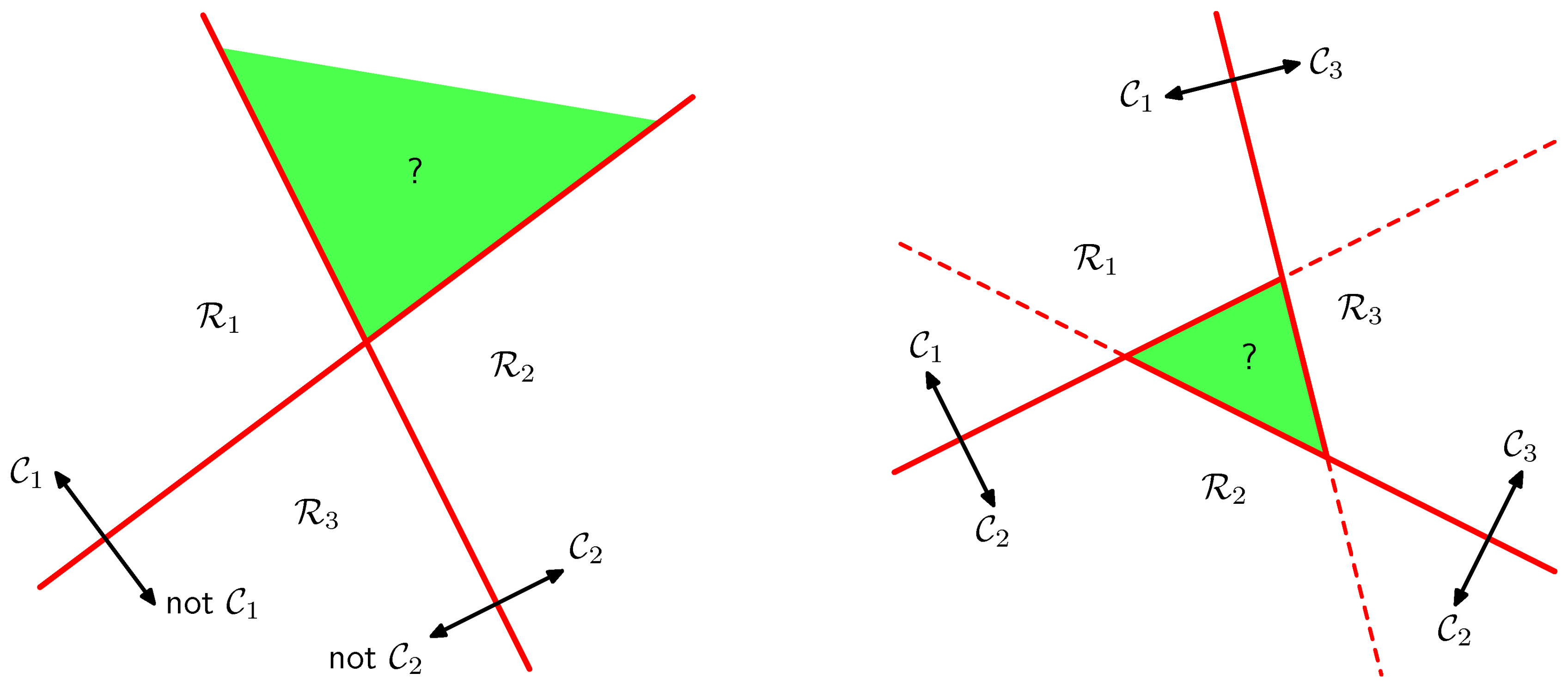
Geometry of linear discriminants



- Decision surface is perpendicular to $\mathbf{w}$
- Displacement is 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



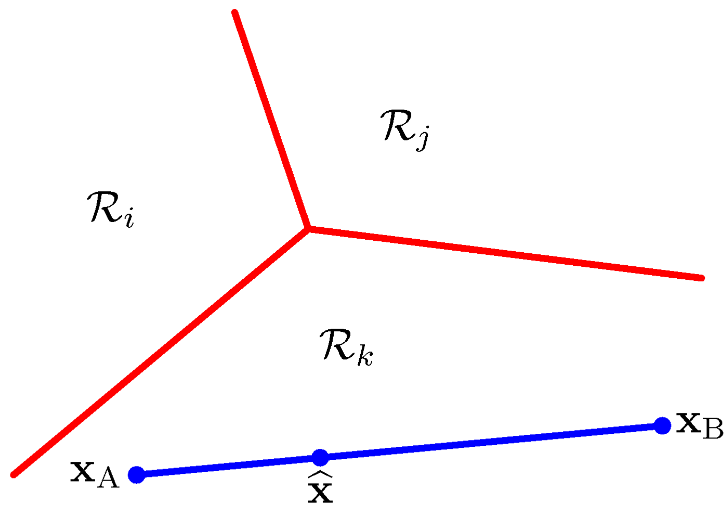
Single K-class classifier

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

- Point $\mathbf{x}$ belongs to class C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$

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

- Decision region singly connected and convex (due to linearity of discriminant functions)



Perceptron algorithm

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

$$y(\mathbf{x}) = f(\mathbf{w}^\top \varphi(\mathbf{x})) \qquad\qquad f(a) = \begin{cases} +1, & a \geq 0, \\ -1, & a < 0 \end{cases}$$

- Train using perceptron criterion (here $t_n \in \{-1,1\}$)

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

  where $\mathcal{M}$ is the set of misclassified patterns

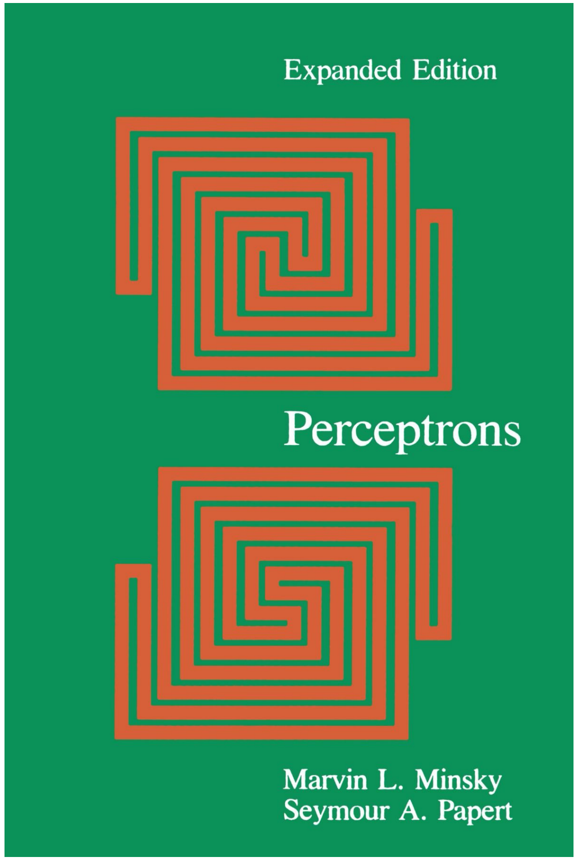
- Note that direct misclassification using total number of misclassified patterns will not work because of non-linear $f(\cdot)$

Perceptron algorithm

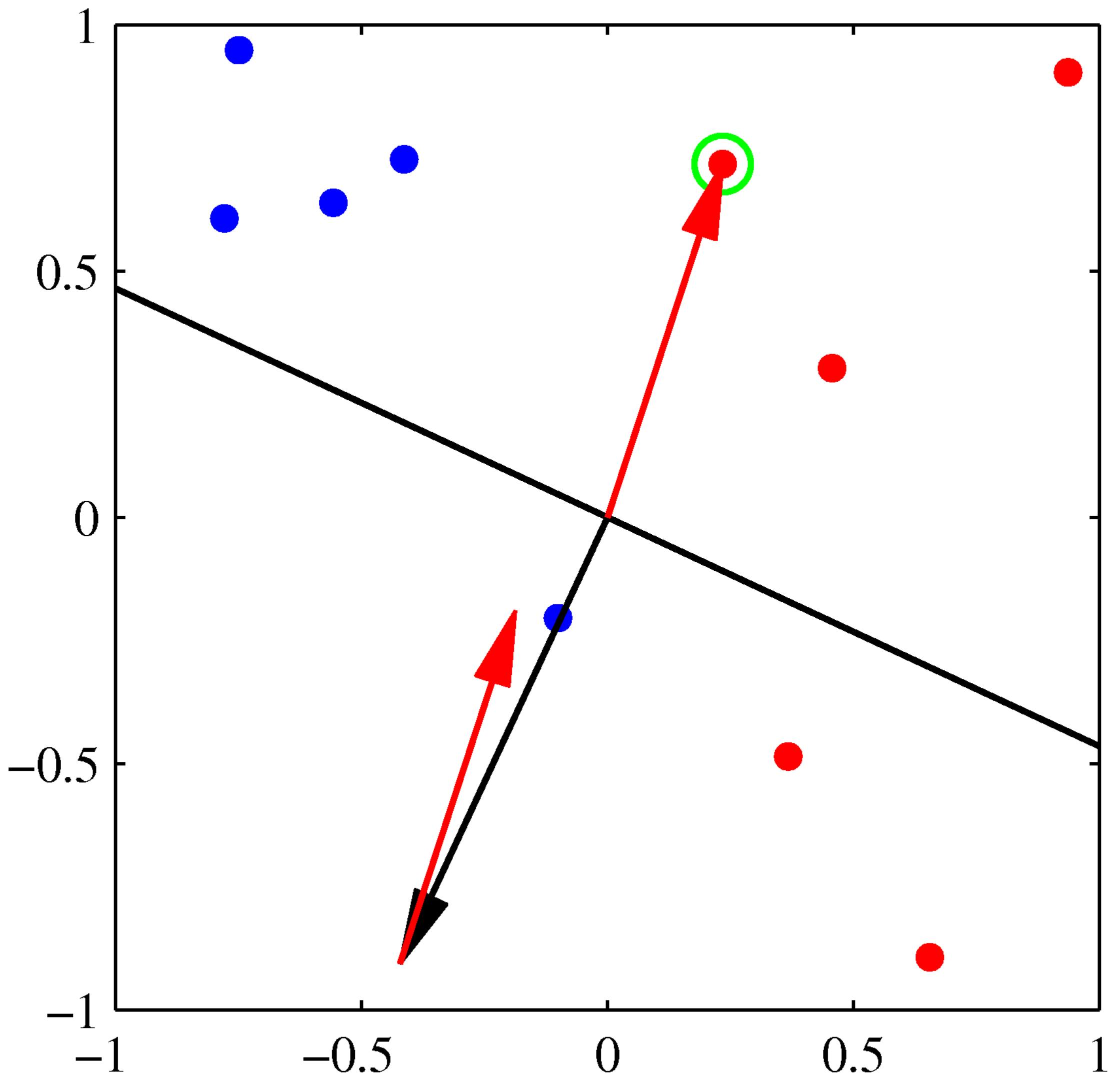
- Total error function is piecewise linear
- Stochastic gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \varphi_n t_n$$

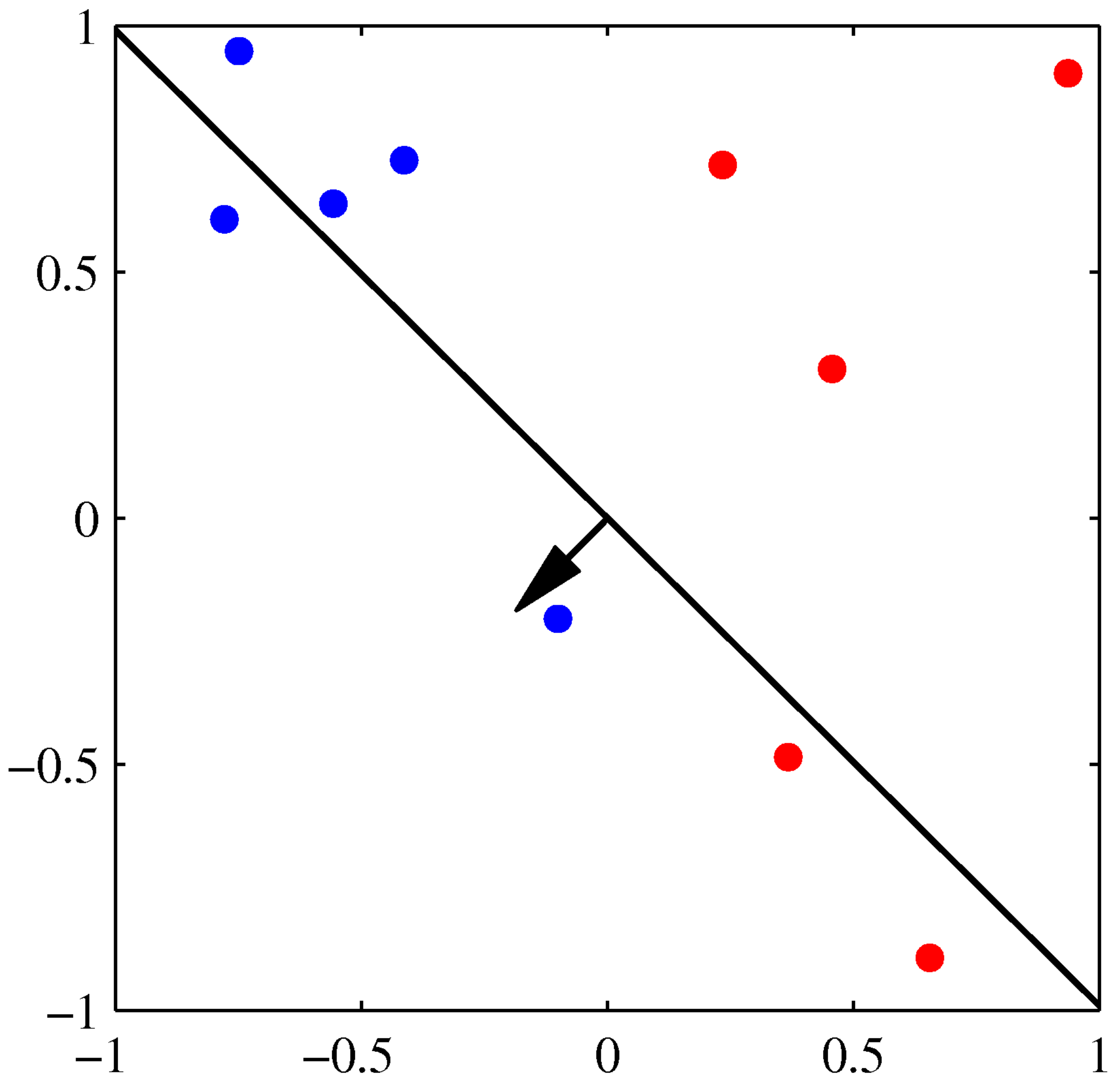
- Update is not a function of $\mathbf{w}$, thus η can be equal to 1
- Perceptron convergence theorem: if there exists and exact solution, then PA will find a solution in a finite number of steps
- Attacked by Minsky and Papert in *Perceptrons* (1969). Attack valid only for single-layer perceptrons. Consequence: research stopped in neural computation for nearly a decade



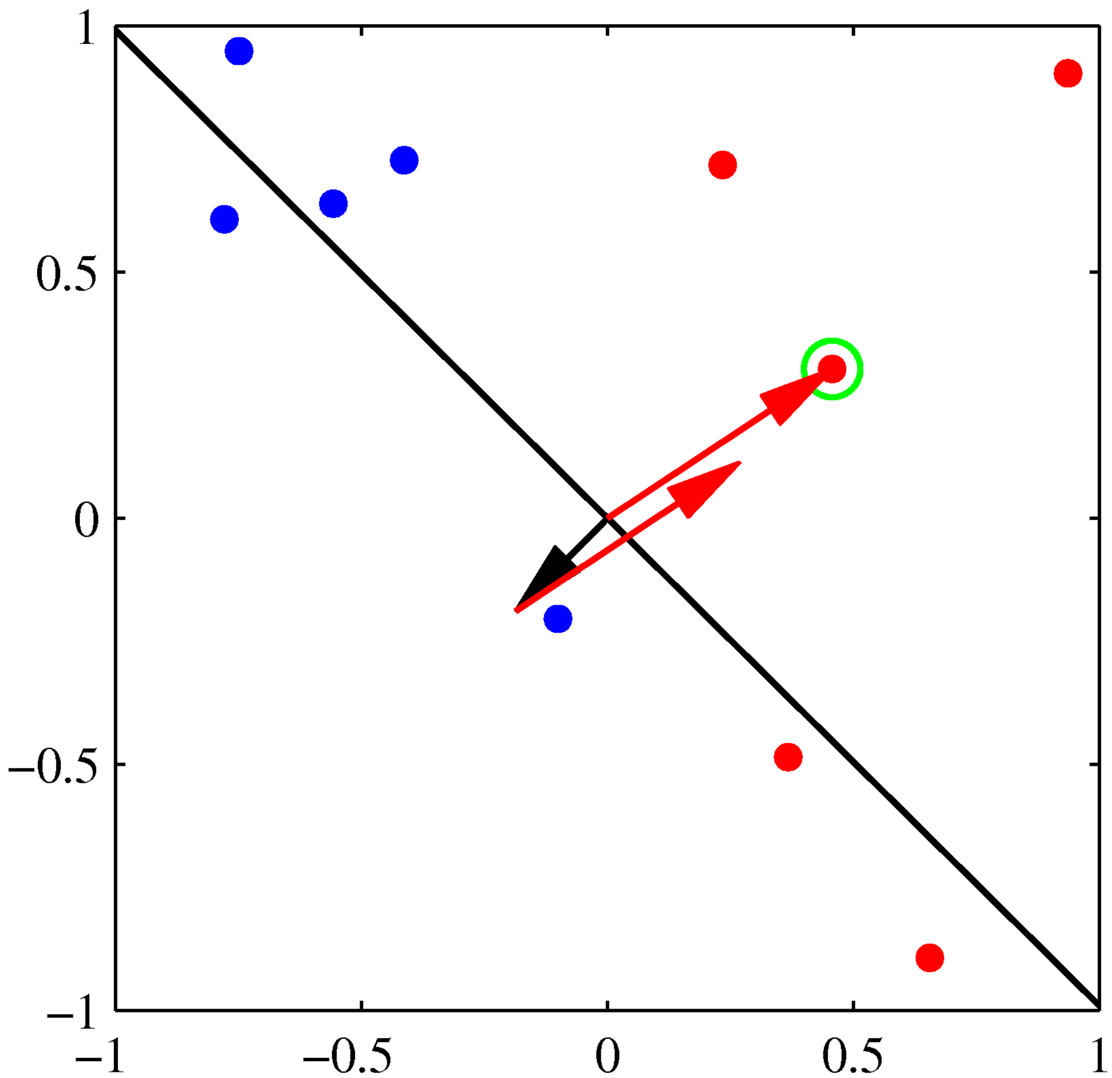
Perceptron algorithm



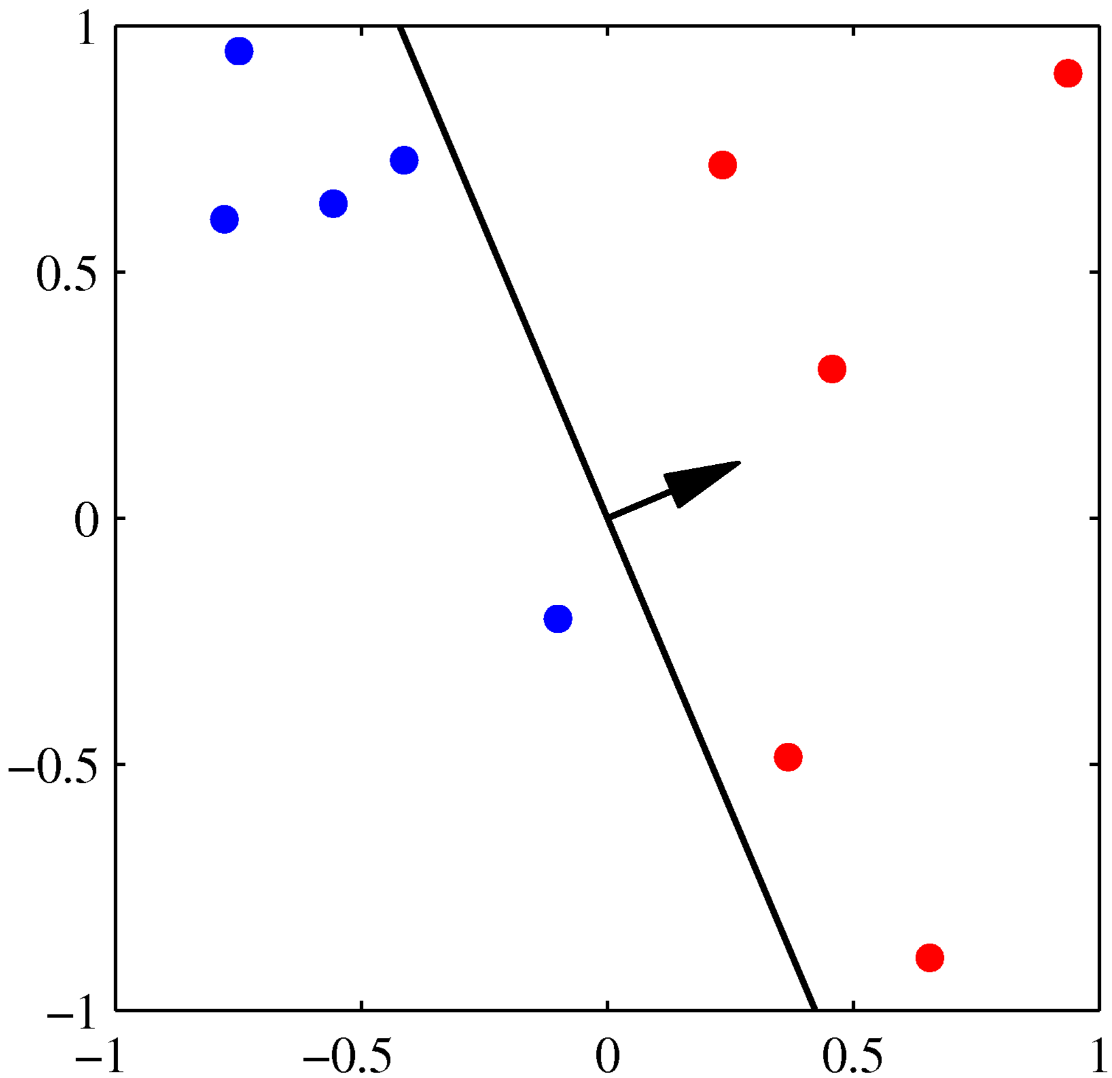
Perceptron algorithm



Perceptron algorithm



Perceptron algorithm



Probabilistic generative models

- Model class-conditional densities $p(\mathbf{x} \mid C_k)$
- Posterior probability for class C_1:

$$p(C_1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_1)p(C_1)}{p(\mathbf{x} \mid C_1)p(C_1) + p(\mathbf{x} \mid C_2)p(C_2)} = \frac{1}{1 + \exp(-a)} = \sigma(a)$$

where have defined $a = \ln \dfrac{p(\mathbf{x} \mid C_1)p(C_1)}{p(\mathbf{x} \mid C_2)p(C_2)}$

- σ is the logistic sigmoid function
- The inverse of σ is the logit function $a = \ln \left(\dfrac{\sigma}{1 - \sigma} \right)$

Probabilistic generative models

- Generalization to multiple classes:

$$p(C_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_k)p(C_k)}{\sum_j p(\mathbf{x} \mid C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln\big(p(\mathbf{x} \mid C_k)p(C_k)\big)$

- This is known as the softmax function, because it is a smoothed version of the max

- Different representations for class-conditional densities yield different consequences in how classification is done

Continuous inputs

- First assume that all classes share the same covariance matrix and that there are only 2 classes.
- We have

$$p(C_1 \,|\, \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + w_0)$$

$$p(\mathbf{x} \,|\, C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \mu_k)^\top \Sigma^{-1} (\mathbf{x} - \mu_k) \right\}$$

where

$$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2}\mu_1^\top \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_2^\top \Sigma^{-1} \mu_2 + \ln \frac{p(C_1)}{p(C_2)}$$

Continuous inputs

- Quadratic term from Gaussian vanishes. The priors $p(C_k)$ only enter via the bias parameter
- For the general case of K classes, we have

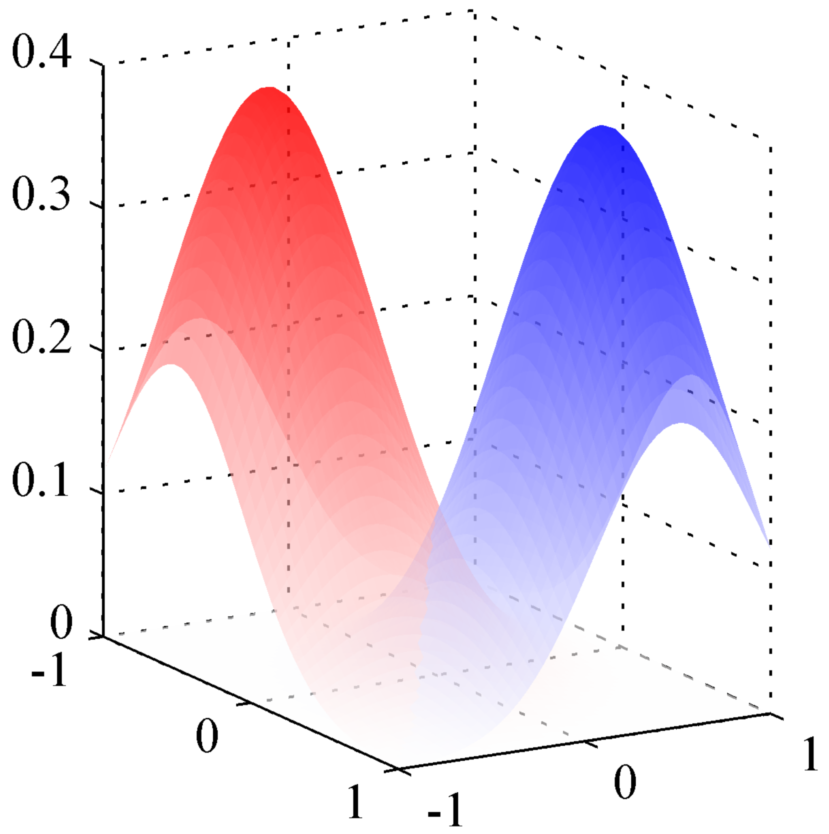
$$a_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

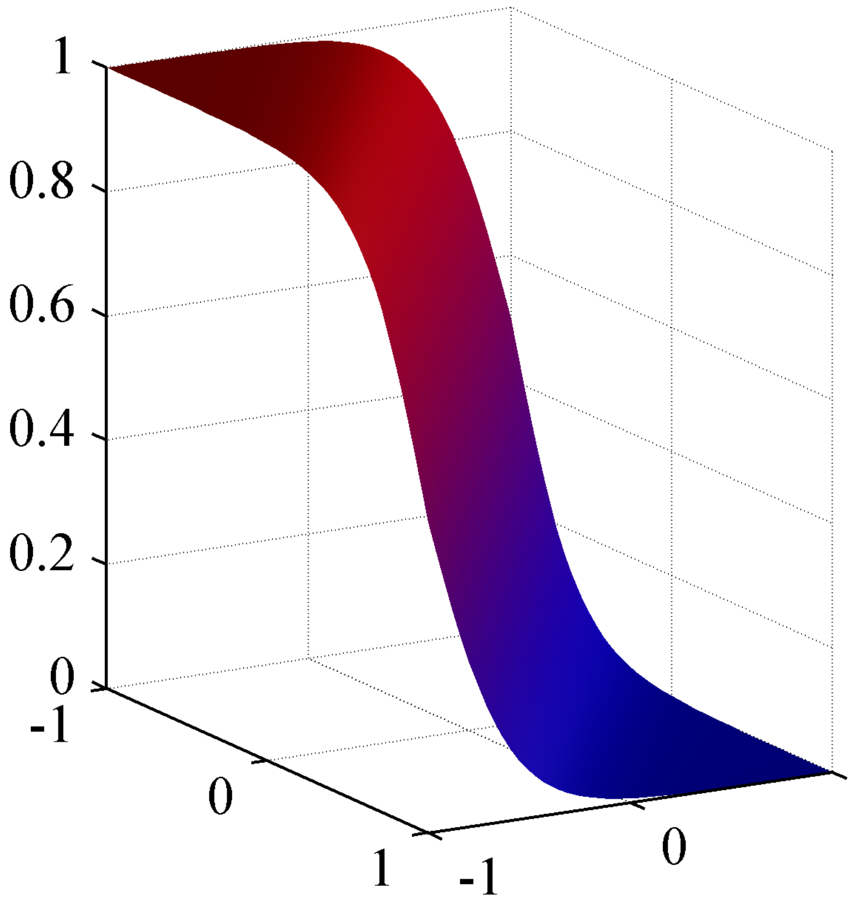
where

$$\mathbf{w}_k = \Sigma^{-1}\mu_k$$

$$w_{k0} = -\frac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \ln p(C_k)$$

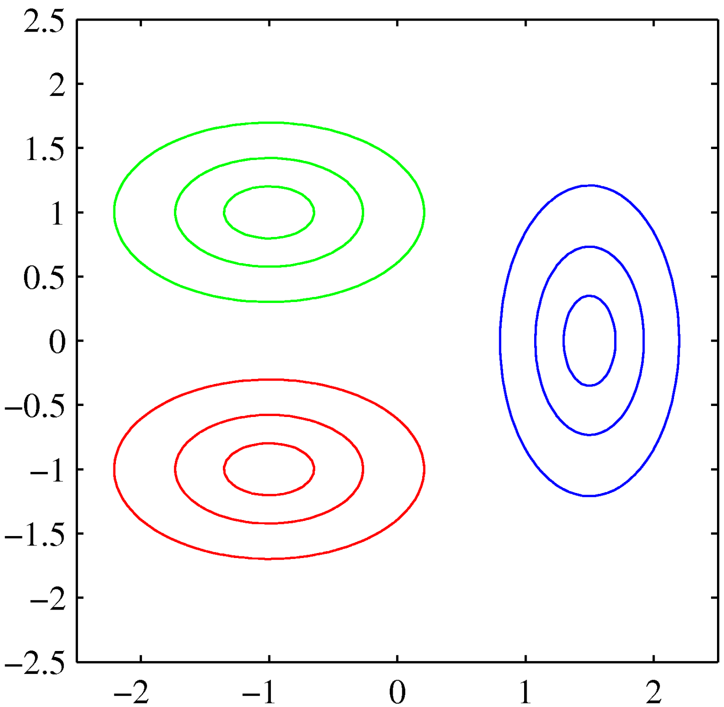
Continuous inputs

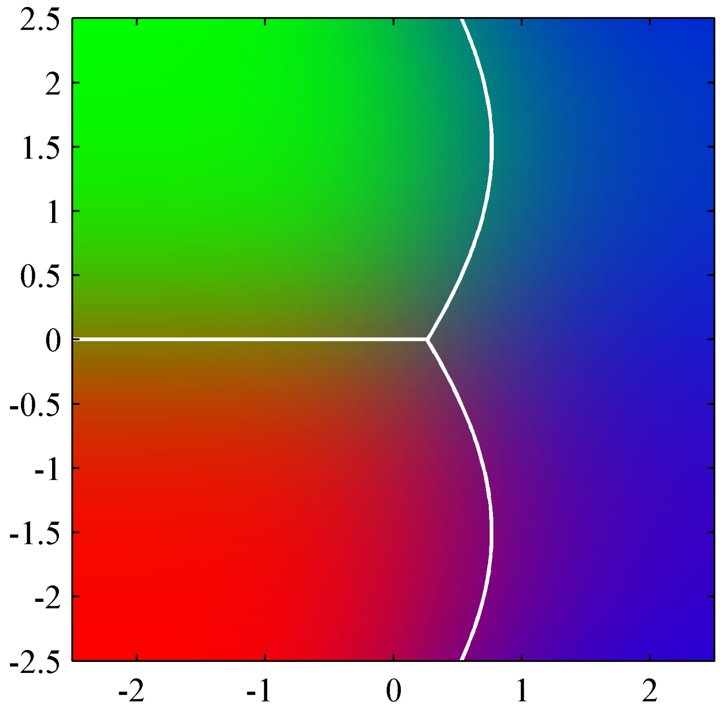




Linear versus quadratic

- When covariance is shared by classes, the decision boundary is linear
- When covariances are unlinked, the decision boundary is quadratic





Maximum likelihood

- Since we have a parametric form for class-conditional densities $p(\mathbf{x} \mid C_k)$, we can determine values of the parameters and priors $p(C_k)$

$$p(\mathbf{x}_n, C_1) = p(C_1)p(\mathbf{x}_n \mid C_1) = q\mathcal{N}(\mathbf{x}_n \mid \mu_1, \Sigma)$$

$$p(\mathbf{x}_n, C_2) = p(C_2)p(\mathbf{x}_n \mid C_2) = (1 - q)\mathcal{N}(\mathbf{x}_n \mid \mu_2, \Sigma)$$

- Let $t_n \in \{0,1\}$, then the likelihood is then given by

$$p(\mathbf{t}, \mathbf{X} \mid q, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} \left[q\mathcal{N}(\mathbf{x}_n \mid \mu_1, \Sigma)\right]^{t_n} \left[(1 - q)\mathcal{N}(\mathbf{x}_n \mid \mu_2, \Sigma)\right]^{1-t_n}$$

Maximum likelihood

- The log-likelihood function with relevant terms for q is:

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

- Maximize with respect to q, yields

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

$$\boxed{p(\mathbf{t}, \mathbf{X} \mid q, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} \left[q \mathcal{N}(\mathbf{x}_n \mid \mu_1, \Sigma) \right]^{t_n} \left[(1 - q) \mathcal{N}(\mathbf{x}_n \mid \mu_2, \Sigma) \right]^{1 - t_n}}$$

Maximum likelihood

- The log-likelihood function with relevant terms for μ_1 is:

$$\sum_{n=1}^{N} t_n \ln \mathcal{N}(\mathbf{x}_n \mid \mu_1, \Sigma) = -\frac{1}{2}\sum_{n=1}^{N} t_n (\mathbf{x}_n - \mu_1)^{\top}\Sigma^{-1}(\mathbf{x}_n - \mu_1) + \text{const}$$

- Maximize with respect to μ_1, yields

$$\mu_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n$$

$$\boxed{p(\mathbf{t}, \mathbf{X} \mid q, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} \left[q\mathcal{N}(\mathbf{x}_n \mid \mu_1, \Sigma)\right]^{t_n} \left[(1-q)\mathcal{N}(\mathbf{x}_n \mid \mu_2, \Sigma)\right]^{1-t_n}}$$

Logistic regression

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

$$p(C_1 \mid \varphi) = y(\varphi) = \sigma(\mathbf{w}^\top \varphi) \qquad \frac{\mathrm{d}\sigma(a)}{\mathrm{d}a} = \sigma(a)(1 - \sigma(a))$$

- More compact than maximum likelihood fitting of Gaussians. For M parameters, Gaussian model uses $2M$ parameters for the means, and $M(M+1)/2$ parameters for the shared covariance matrix

- Maximum likelihood: $p(\mathbf{t} \mid \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$

Logistic regression

- Negative log of likelihood yields cross entropy

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

- Gradient with respect to $\mathbf{w}$

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

- Therefore, we have the same form for the gradient for the sum-of-squares error

$$\nabla \ln p(\mathbf{t} \mid \mathbf{w}, \beta) = \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^\top \varphi(x_n) \right\} \varphi(x_n)^\top$$

Iterative reweighted least squares

- Efficient iterative optimization: Newton-Raphson

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

  where H is the Hessian matrix (with second derivatives)

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

- For cross entropy we get a similar set of normal equations for weighted least squares, which depends on $\mathbf{w}$

- This dependency forces us to apply the update iteratively

Iterative reweighted least squares

- Apply this to linear regression

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^\top \varphi_n - t_n)\varphi_n = \Phi^\top \Phi \mathbf{w} - \Phi^\top \mathbf{t}$$

$$H = \nabla \nabla E(\mathbf{w}) = \sum_{n=1}^{N} \varphi_n \varphi_n^\top = \Phi^\top \Phi$$

- The Newton-Raphson update then takes

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - H^{-1} \nabla E(\mathbf{w}) = \mathbf{w}^{\text{old}} - (\Phi^\top \Phi)^{-1} \left\{ \Phi^\top \Phi \mathbf{w}^{\text{old}} - \Phi^\top \mathbf{t} \right\}$$

$$= (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

Iterative reweighted least squares

- Apply this to logistic regression

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\varphi_n = \Phi^\top(\mathbf{y} - \mathbf{t})$$

$$H = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1 - y_n)\varphi_n\varphi_n^\top = \Phi^\top R\Phi$$

with R as diagonal matrix with $R_{nn} = y_n(1 - y_n)$

Iterative reweighted least squares

- The Newton-Raphson update then takes

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - H^{-1}\nabla E(\mathbf{w}) = \mathbf{w}^{\text{old}} - (\Phi^\top R\Phi)^{-1}\Phi^\top(\mathbf{y}-\mathbf{t})$$

$$= (\Phi^\top R\Phi)^{-1}\left\{\Phi^\top R\Phi\mathbf{w}^{\text{old}} - \Phi^\top(\mathbf{y}-\mathbf{t})\right\}$$

$$= (\Phi^\top R\Phi)^{-1}\Phi^\top R\mathbf{z}$$

where

$$\mathbf{z} = \Phi\mathbf{w}^{\text{old}} - R^{-1}(\mathbf{y}-\mathbf{t})$$