Machine Learning

Session 9 Linear Models for Classification

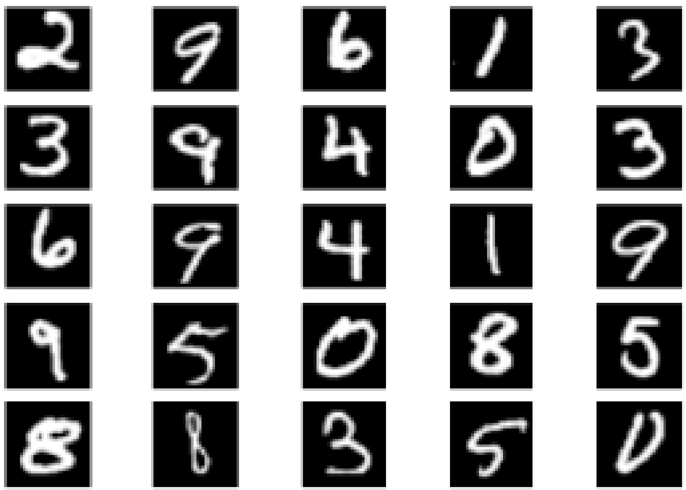
Introduction

Logistic Regression

Multi-class (soft-max) Logistic Regression

Material taken from Bishop (chapter 4.1.1; 4.1.2,4.1.3, 4.2 (only introduction),4.3 , 4.3.2, 4.3.4 and

Classification in context



What digit is this?
How can I predict this? What are my input features?

Is this a dog?
How can I predict this? What are my input features?

Am I going to pass the exam?
How can I predict this? What are my input features?

Linear Models for Classification

- **Discriminative approach:**
  - Learn the boundary parameters directly
  - Examples: Support Vector Machines, logistic regression, SoftMax …

- **Generative approach:**
  - Model the distribution of inputs characteristic of the class $p(\mathbf{x}|y = k)$ and apply Bayes rule
  - Better interpretability
  - Usually requires more parameters to learn
  - Examples: Gaussian models with shared covariances, Naïve Bayes

Linear Models for Classification

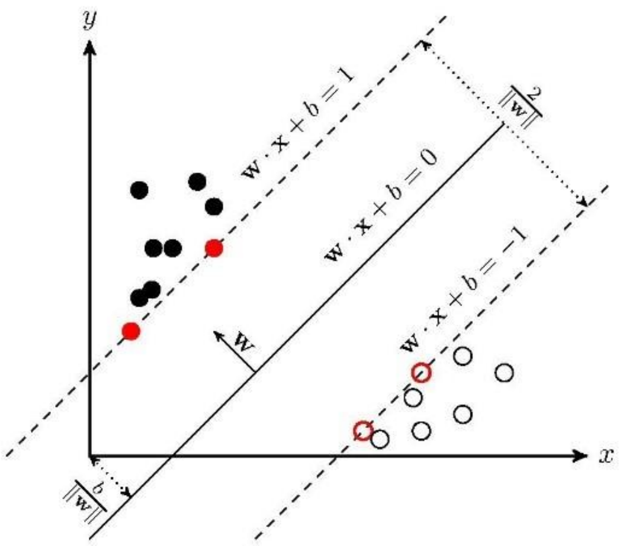
- **Linear discriminant functions:**
  - Decision function

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

  - Classification

    if $h_{\mathbf{w}}(\mathbf{x}) \geq 0$ then $\mathbf{x}$ belongs to class $+1$

    if $h_{\mathbf{w}}(\mathbf{x}) < 0$ then $\mathbf{x}$ belongs to class -1



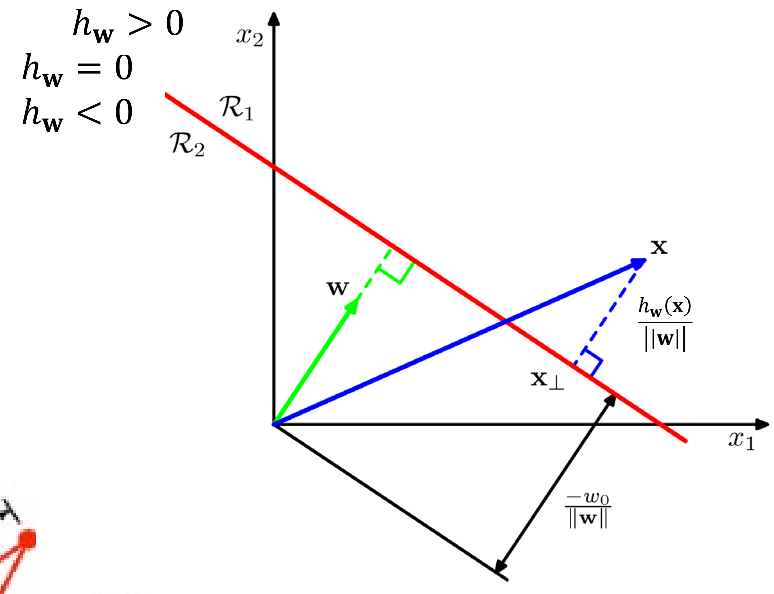
  - Decision surface is a hyperplane of $D-1$ dims, equation $h_{\mathbf{w}}(\mathbf{x}) = 0$
  - $\mathbf{w}$ is the normal vector to the plane, pointing to the $+1$ class
  - w_0 determines de location of the decision surface

Linear Models for Classification

- **Linear discriminant functions:**
  - Two dimensional example: $h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1$

- The value $h_{\mathbf{w}}(\mathbf{x})$ gives a **signed** measure of the distance r of $\mathbf{x}$ to the decision surface

$$r = \frac{h_{\mathbf{w}}(\mathbf{x})}{\|\mathbf{w}\|}$$

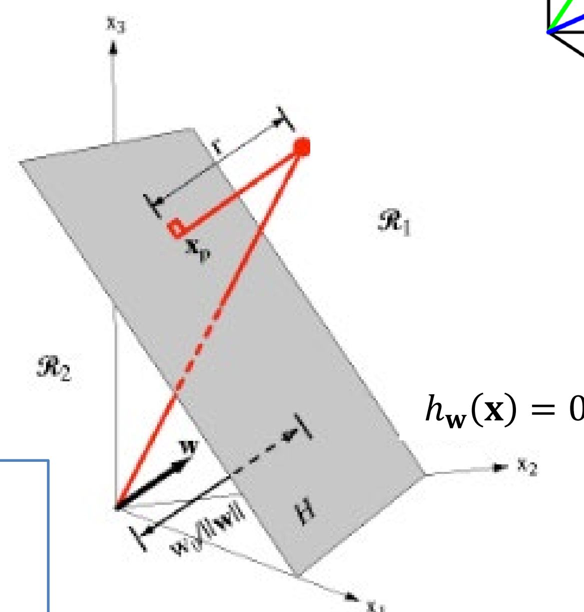


As in linear regression, we introduce *dummy* input $x_0 = 1$ and define w_0 as an extra parameter:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

3D Example:

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2$$



Note: $h_{\mathbf{w}}(\mathbf{x})$ gives an ordered (and signed) value according to the distance from the point $\mathbf{x}$ to the hyperplane $h_{\mathbf{w}}(\mathbf{x}) = 0$

Linear Models for Classification

Can we do classification using least-squares regression?

Naïve (and wrong) choice:

- Suppose we have a binary problem $y \in \{-1,1\}$
- Use the **class output directly as target variable**
- Assuming the standard model used for regression:

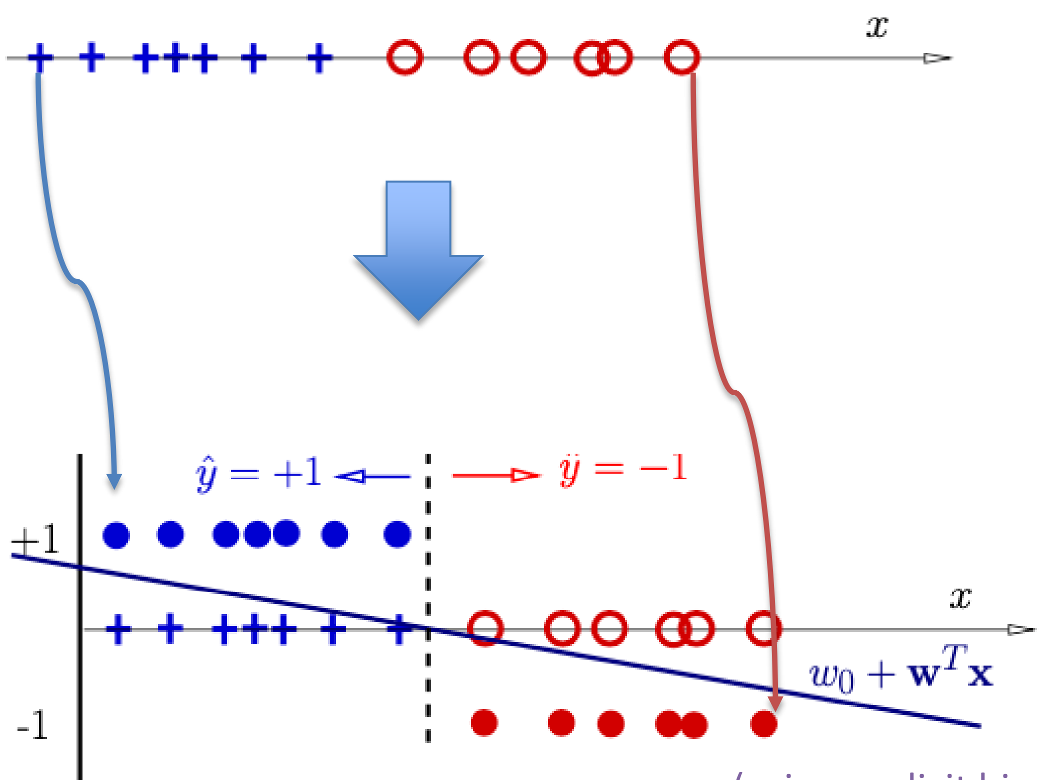
$$h_w(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

- We obtain $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-\mathbf{1}} \mathbf{X}^\top \mathbf{y}$
- The loss to minimize:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left(h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) - y^{(n)} \right)^2$$

- How do I compute a label for a new example?

Linear Models for Classification

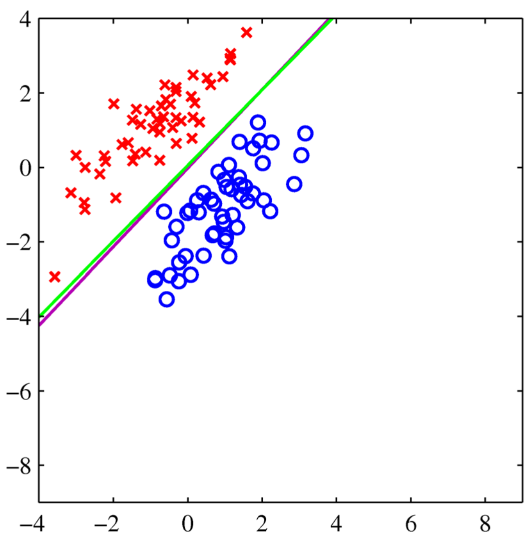


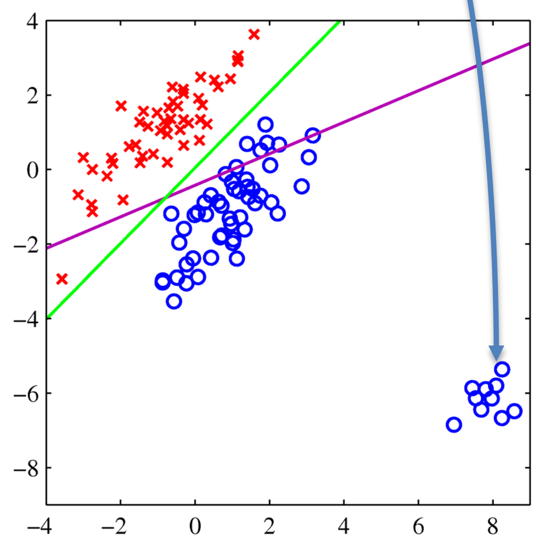
(using explicit bias term w_0 notation)

Our linear classifier can be the **sign** of $h_{\mathbf{w}}(\mathbf{x})$
The *linear* boundary: $w_0 + \mathbf{w}^\top \mathbf{x} = 0$ separates the space into two "half-spaces": in 1D is a value, in 2D is a line, in 3D a plane, etc

Linear Models for Classification

- Naïve (and wrong) choice: simply do a linear regression from $\mathbf{x}$ to y by minimizing the least squares error:
  - Poor results in the presence of outliers
  - Penalizes predictions that are too correct





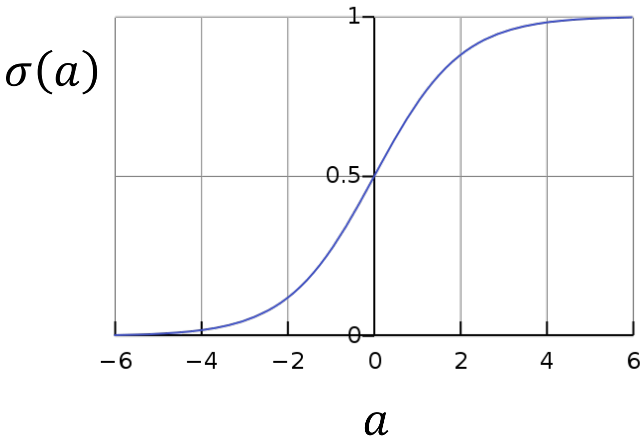
  - Least-squares solution decision (wrong choice)
  - Logistic regression decision (explained later)

Linear Models for Classification

Logistic regression

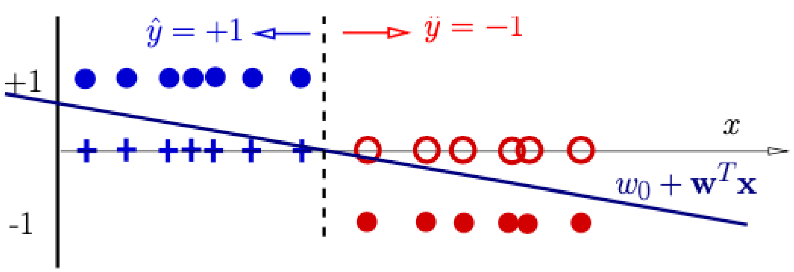
Instead of sign we take the **Logistic function:**

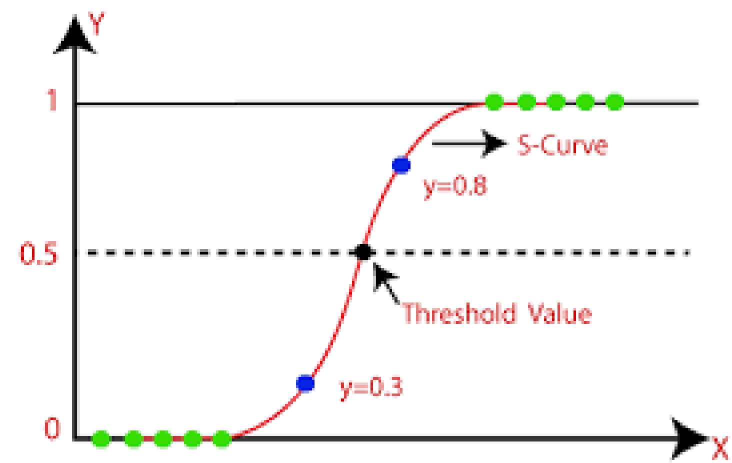
$$\sigma(a) = \frac{1}{1+\exp(-a)} = \frac{\exp(a)}{\exp(a)+1}$$



- Bounded between zero and one
- Symmetric $\sigma(-a) = 1 - \sigma(a)$
- Continuous and differentiable
- Its derivative is simple

$$\sigma'(a) = \sigma(a)\big(1 - \sigma(a)\big)$$





Linear Models for Classification

Logistic regression

- Our classifier adds a "new layer". It "squashes" the result of the linear mapping using the logistic function

$$h_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$$

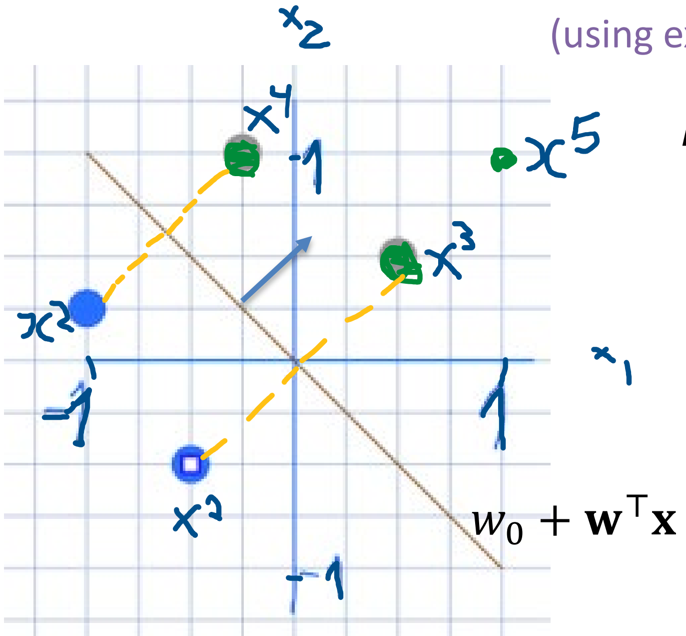
with $\sigma(a) = \frac{1}{1+\exp(-a)}$

We are not increasing the number of parameters!

- The output $h_{\mathbf{w}}$ is a **smooth** function of the inputs $\mathbf{x}$ (each coordinate a feature) and the weights $\mathbf{w}$ (a parameter for each feature)

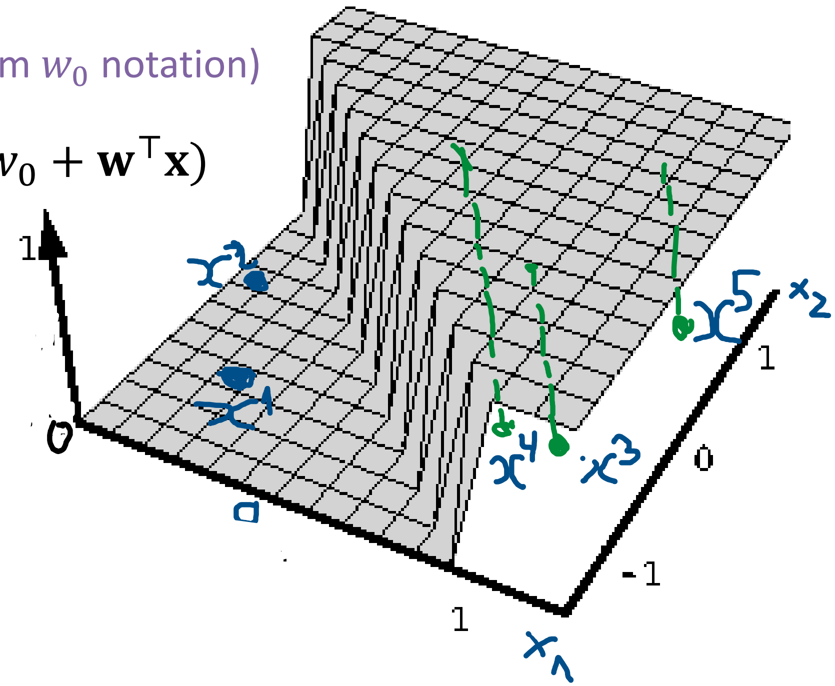
Linear Models for Classification

Logistic regression: intuitions



(using explicit bias term w_0 notation)

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



point			class	wTx=x1+x2+0	h_w(x)
x1	-0,50	-0,50	0	-1	0,2689
x2	-1,00	0,25	0	-0,75	0,3208
x3	0,50	0,50	1	1	0,7311
x4	-0,25	1,00	1	0,75	0,6792
x5	1,00	1,00	1	2	0,8808

NOTE: points far from the decision surface are close to zero or close to one. Vector $\mathbf{w}$ points to the class ‘1’

Linear Models for Classification

- Logistic regression: Probabilistic interpretation

  - For a binary class problem, the logistic sigmoid can be interpreted as posterior probabilities

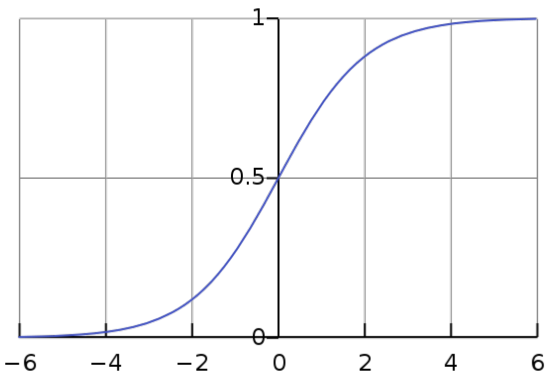
  - Remember Bayes' Theorem

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)} = \frac{1}{1 + \frac{p(\mathbf{x}|C_2)p(C_2)}{p(\mathbf{x}|C_1)p(C_1)}}$$

$$= \frac{1}{1 + \exp(-a)}$$

for $a = \ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)} = \ln \frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})} + \ln \frac{p(C_1)}{p(C_2)}$

$$\boxed{h_{\mathbf{w}}(\mathbf{x}) = p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})}$$



Linear Models for Classification

- Logistic regression learning (parameter estimation)

For a dataset $\{\mathbf{x}^{(n)}, y^{(n)}\}$, where $y^{(n)} \in \{0,1\}$, $\quad n = 1, \dots, N$

- Probabilistic interpretation: Outputs $y^{(n)}$ are Bernoulli variables
  - value 1 with probability p
  - value 0 with probability $1 - p$

$$\text{Ber}(Y): \; P(Y = y) = p^y (1-p)^{1-y}, \qquad y = 0,1$$

- The probabilities are approximated using the logistic model

$$p(y|\mathbf{x}, \mathbf{w}) = h_{\mathbf{w}}(\mathbf{x})^y \left(1 - h_{\mathbf{w}}(\mathbf{x})\right)^{1-y}$$

where $h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) \; = \sigma(\mathbf{w}^\top \mathbf{x})$

Linear Models for Classification

- Logistic regression learning (parameter estimation)

For a dataset $\{\mathbf{x}^{(n)}, y^{(n)}\}$, where $y^{(n)} \in \{0,1\}, \quad n = 1, \dots, N$

- The likelihood function (to be maximized) of the parameters $\mathbf{w}$ for the entire dataset can be written (for i.i.d. examples)

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right)^{y^{(n)}} \left(1 - h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right)\right)^{1-y^{(n)}}$$

where $h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) = \sigma\left(\mathbf{w}^{\top}\mathbf{x}^{(n)}\right)$

Linear Models for Classification

- Logistic regression learning (parameter estimation)

For a dataset $\{\mathbf{x}^{(n)}, y^{(n)}\}$, where $y^{(n)} \in \{0,1\}$, $\quad n = 1, \ldots, N$

- The likelihood function (to be maximized) of the parameters $\mathbf{w}$ for the entire dataset can be written (for i.i.d. examples)

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right)^{y^{(n)}} \left(1 - h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right)\right)^{1-y^{(n)}}$$

where $h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) \ = \sigma\left(\mathbf{w}^{\top}\mathbf{x}^{(n)}\right)$

- Taking logs and changing sign we obtain the error function (to be minimized). Also known as (binary) cross-entropy

$$E(\mathbf{w}) = -\sum_{n=1}^{N} y^{(n)} \ln h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) + \left(1 - y^{(n)}\right) \ln\left(1 - h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right)\right)$$

Linear Models for Classification

- Logistic regression learning (parameter estimation)

For a dataset $\{\mathbf{x}^{(n)}, y^{(n)}\}$, where $y^{(n)} \in \{0,1\}$, $\;n = 1, \dots, N$

- Taking logs and changing sign we obtain the error function (to be minimized). Also known as (binary) cross-entropy

$$E(\mathbf{w}) = -\sum_{n=1}^{N} y^{(n)} \ln h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) + \left(1 - y^{(n)}\right) \ln\left(1 - h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right)\right)$$

where $h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) = \sigma\left(\mathbf{w}^{\top}\mathbf{x}^{(n)}\right)$

- No closed-form solution for $\mathbf{w}$
- The gradient of the error function with respect to $\mathbf{w}$

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_{n=1}^{N} \left(h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) - y^{(n)}\right) \cdot \mathbf{x}^{(n)}$$

similar form as the gradient of the least-squares linear regression

Application of Gradient Descent for Logistic regression

[D. Mackay book (pag. 477)](#)

Notation differences:

$\alpha \to \eta$

$E(\mathbf{w}) \to G(\mathbf{w})$

$|\mathbf{w}| \to E_W(\mathrm{w})$

$y \to a$

$h_{\mathbf{w}} \to y$

The magnitude of the weights **increases** with the number of iterations (remember **overfitting**)

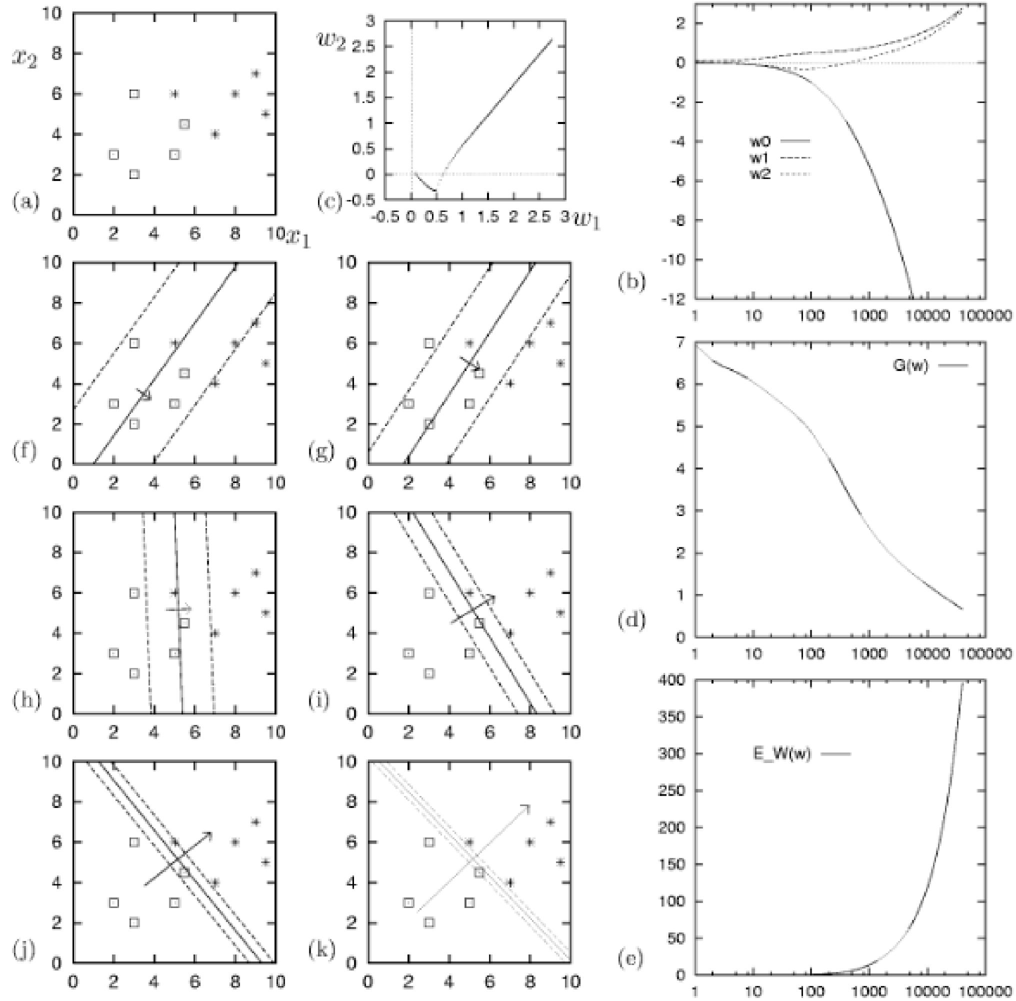
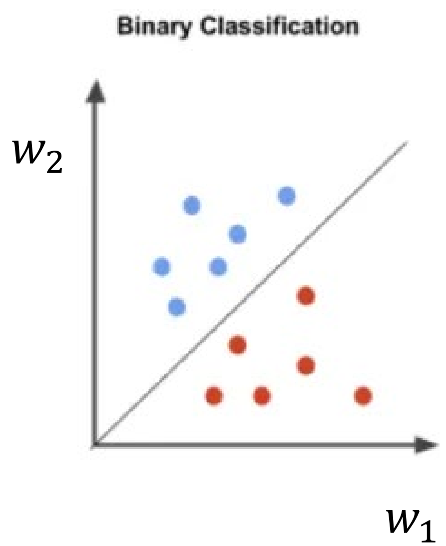
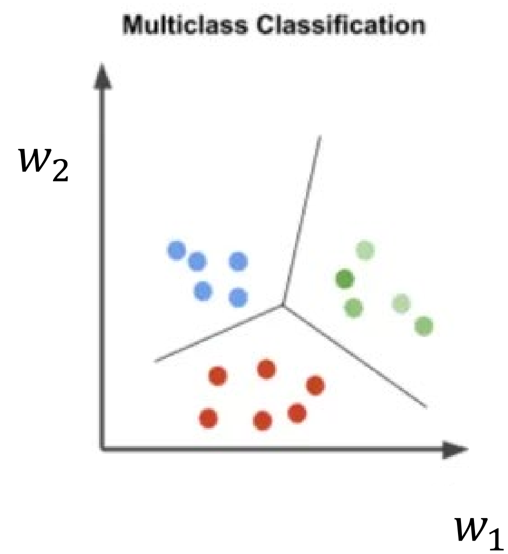


Figure 39.4. A single neuron learning to classify by gradient descent. The neuron has two weights w_1 and w_2 and a bias w_0. The learning rate was set to $\eta = 0.01$ and batch-mode gradient descent was performed using the code displayed in algorithm 39.5. (a) The training data. (b) Evolution of weights w_0, w_1 and w_2 as a function of number of iterations (on log scale). (c) Evolution of weights w_1 and w_2 in weight space. (d) The objective function $G(\mathbf{w})$ as a function of number of iterations. (e) The magnitude of the weights $E_W(\mathbf{w})$ as a function of time. (f–k) The function performed by the neuron (shown by three of its contours) after 30, 80, 500, 3000, 10 000 and 40 000 iterations. The contours shown are those corresponding to $a = 0, \pm 1$, namely $y = 0.5, 0.27$ and 0.73. Also shown is a vector proportional to (w_1, w_2). The larger the weights are, the bigger this vector becomes, and the closer together are the contours.

Linear Models for Classification

Multi-Class Classification
Classifying more than 2 classes





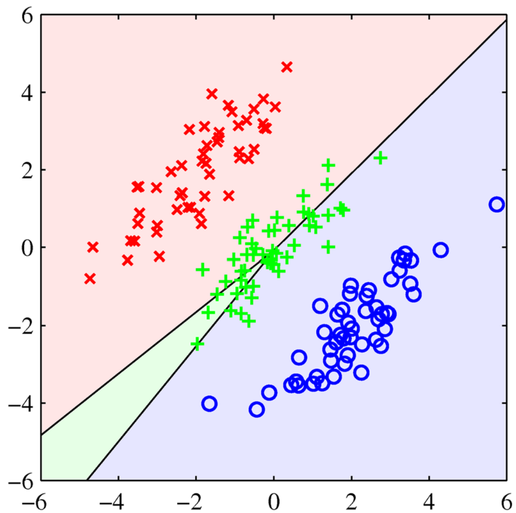
Output representation (one-hot encoding)

vectors $\mathbf{y}^{(n)}$ (1-of-K) with $y_k^{(n)} = 1$ if $\mathbf{x}^{(n)}$ belongs to class k, and $y_k^{(n)} = 0$ otherwise

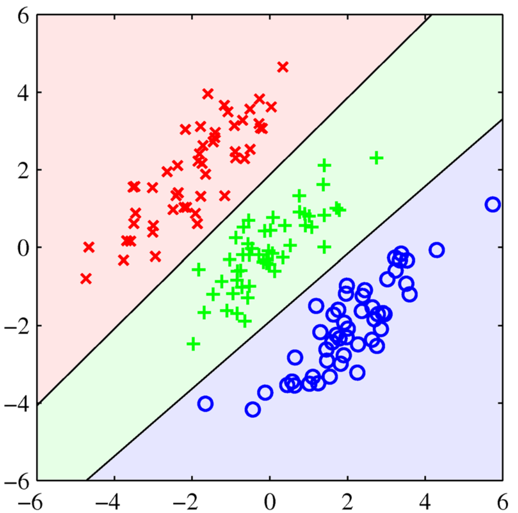
Example: $K = 3$ classes $\rightarrow \mathbf{y} = [0,0,1]^\top$ for an example of class 3

Linear Models for Classification

- Naïve (and again wrong) choice: simply do a linear regression from $\mathbf{x}$ to $\mathbf{y}$ by minimizing the least squares error:
  - Assumes that the target data is generated according to a Gaussian
  - Conditioning on $\mathbf{x}$, the targets $\mathbf{y}$ are far from being Gaussian
  - 3- class example



Least-Squares solution (wrong)



Multi-Class Logistic Regression (explained later)

Linear Models for Classification

- Logistic regression for multiple classes (soft-max)

- Generalization to $K > 2$ classes

- **Soft-max** function generalizes logistic function.
  It "squashes" K numbers (scores) exponential function
  Provides K normalized (between 0 and 1) outputs

$$\mathrm{softmax}(\mathbf{a}) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}$$

The soft-max function is a "soft" version of the **argmax** function

Linear Models for Classification

- Logistic regression for multiple classes (soft-max)

- Generalization to $K > 2$ classes

- **Soft-max** function generalizes logistic function.
  Using a linear model

$$h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) = p(y_k = 1|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

- Requires k parameter vectors $\mathbf{w}_1, \dots, \mathbf{w}_K$ (including respective biases $\mathrm{w}_{01}, \dots w_{0k}$)

  We represent them as a matrix $\mathbf{W}$ of size $(D + 1) \times K$

Linear Models for Classification

- Logistic regression for multiple classes (soft-max)

- Generalization to $K > 2$ classes

- **Soft-max** function generalizes logistic function.
  As before, outputs can also be interpreted as posterior probabilities

$$\boxed{\begin{aligned} p(C_k|\mathbf{x}) &= \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_{j=1}^{K} p(\mathbf{x}|C_j)p(C_j)} \\ &= \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)} \end{aligned}}$$

for $a_k = \ln p(\mathbf{x}|C_k)p(C_k)$

Linear Models for Classification

- Logistic regression for multiple classes (soft-max)

Cross-Entropy Error for parameters $\mathbf{W} = [\mathbf{w}_1, .., \mathbf{w}_K]$:

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_k^{(n)} \ln h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right)$$

$$\text{where } h_{\mathbf{w}}\left(\mathbf{x}^{(n)}\right) = \operatorname{softmax}\left(\mathbf{w}^{\top}\mathbf{x}^{(n)}\right)$$

Again, no closed-form solution for $\mathbf{w}$

Optimization is performed by Gradient Descent

$$\nabla_{\mathbf{w}_k} E(\mathbf{W}) = \sum_{n=1}^{N} \left(h_k^{(n)} - y_k^{(n)}\right) \cdot \mathbf{x}^{(n)}$$

https://cs231n.github.io/linear-classify/

Linear Models for Classification

- **Multiple classes:**

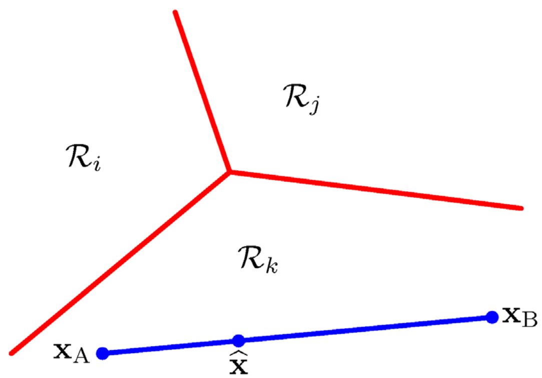
Multi-class Logistic Regression: K linear functions followed by the soft-max

(using explicit bias term w_{k0} notation)

$$h_k(\mathbf{x}) = \mathrm{softmax}\left(\mathbf{w}_k^\top \mathbf{x} + w_{k0}\right)$$

assign $\mathbf{x}$ to class C_k if $h_k(\mathbf{x}) > h_l(\mathbf{x})$ for all $l \neq k$

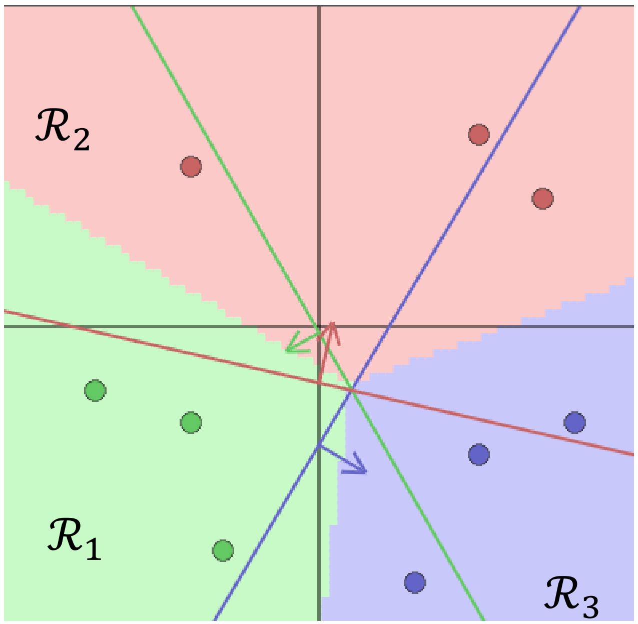
- Decision regions are hyperplanes $(\mathbf{w}_k - \mathbf{w}_l)^\top \mathbf{x} + (w_{k0} - w_{l0}) = 0$
- Always *singly connected* and *convex*



$y_k(\mathbf{x}) > y_l(\mathbf{x})$, for all $l \neq k$

Linear Models for Classification

Example: Green Points → C_1, Red → C_2, Blue → C_3
What is the discriminant function of each class? And the decision surface?



Green line is the **discriminator** of class 1:

Take a green point, always:

$$h_1(\text{green}) \geq h_i(\text{green}), i = 2{,}3$$

The same for red and blue

Decision boundaries (or D. Regions):

$\mathrm{DB}_{12}\{x \mid h_1(\mathbf{x}) = h_2(\mathbf{x})\}$: $(\mathbf{w}_1 - \mathbf{w}_2)^\top \mathbf{x} + (w_{10} - w_{20}) = 0$ is a Hyperplane

(using explicit bias term w_{k0} notation)

- Similarly, DB_{13} and DB_{23}
- The regions are **convex**

Summary

- Least-Squares Linear Classifier (wrong):

Model	$h_{\mathbf{w}}(\mathbf{x})$	$= \mathbf{w}^\top \mathbf{x}$ (or *sign* of)
Least-Squares Error	$E(\mathbf{w})$	$= \frac{1}{2} \sum_{n=1}^{N} \left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right)^2$
Gradient	$\frac{\partial \mathbb{E}(\mathbf{w})}{\partial w_j}$	$= \sum_{n=1}^{N} \left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right) x_j^{(n)}$

- Binary Logistic Regression (one output $\in [0, 1]$):

Model	$h_{\mathbf{w}}(\mathbf{x})$	$=P(\mathcal{C}_1 \vert \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$, where $\sigma(a) = \frac{1}{1+e^{-a}}$
Cross-entropy error	$E(\mathbf{w})$	$= -\sum_{n=1}^{N} \left(y^{(n)} \ln h_{\mathbf{w}}(\mathbf{x}^{(n)}) + (1 - y^{(n)}) \ln(1 - h_{\mathbf{w}}(\mathbf{x}^{(n)})\right)$
Gradient	$\frac{\partial \mathbb{E}(\mathbf{w})}{\partial w_j}$	$=\sum_{n=1}^{N} \left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right) x_j^{(n)}$

- Multi-Class Logistic Regression (K outputs, one-hot encoding):

Model	$h_{\mathbf{W}_k}(\mathbf{x})$	$=P(\mathcal{C}_k \vert \mathbf{x}) = \text{softmax}(\mathbf{w}_k^\top \mathbf{x}) = \frac{e^{\mathbf{w}_k^\top \mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}}}$
Cross-entropy error	$E(\mathbf{W})$	$= -\sum_{n=1}^{N} \sum_{k=1}^{K} y_k^{(n)} \ln h_{\mathbf{W}_k}(\mathbf{x}^{(n)})$
Gradient	$\frac{\partial \mathbb{E}(\mathbf{W})}{\partial \mathbf{w}_j}$	$=\sum_{n=1}^{N} \left(h_j^{(n)} - y_j^{(n)}\right) x_j^{(n)}$