

# Machine Learning I

## Probabilistic Linear Classification

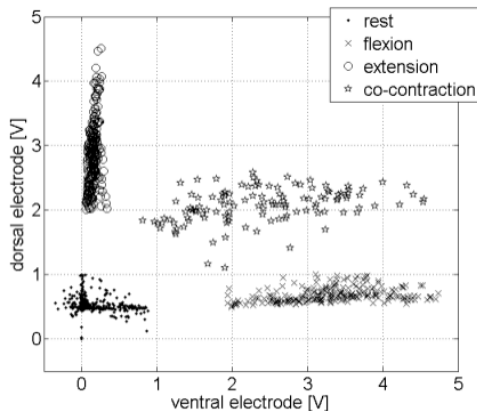
Justin Bayer

Technische Universität München

# Classification Problems

**Goal:** Assign unknown input vector  $\mathbf{x}$  to one of  $K$  classes, denoted  $\mathcal{C}_k$   
 $k = 1, \dots, K$ .

**Example:** sEMG data.



Input space  $X$ : typically  $\mathbb{R}^n$ .

Class space  $Z$ : set containing classes, e.g.

`{rest, flexion, extension, co – contraction}`

These are hard to work with, since there are no mathematical operations defined on them.

How do we encode the set of classes to work with our models?

# 1-of-K encoding

One binary per class. This seems arbitrary, but it turns out to be useful!

$$\text{rest} \equiv (1, 0, 0, 0)$$

$$\text{flexion} \equiv (0, 1, 0, 0)$$

$$\text{extension} \equiv (0, 0, 1, 0)$$

$$\text{co - contraction} \equiv (0, 0, 0, 1)$$

$\{0, 1\}^k \subset \mathbb{R}^k \Rightarrow$  all mathematical operations well defined  $\Rightarrow$  works with our models.

Note that each class vector has equal Euclidean distance to each other class vector.

# Least squares for classification

**Idea:** Good model for linear regression, use it for classification?  
How do we find parameters  $\mathbf{W}$ ?

# Least squares for classification

**Idea:** Good model for linear regression, use it for classification?  
How do we find parameters  $\mathbf{W}$ ?

**Recap:** For regression, we minimized the quadratic error function

$$E_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2} \sum^n (\mathbf{z}_n - \mathbf{W}\mathbf{x})^T (\mathbf{z}_n - \mathbf{W}\mathbf{x}),$$

that resulted from log likelihood on Gaussians.

$\implies$  Can find exact closed form solution for  $\mathbf{W}$  using pseudo-inverse, as before.

# Least squares for classification

**Idea:** Good model for linear regression, use it for classification?  
How do we find parameters  $\mathbf{W}$ ?

**Recap:** For regression, we minimized the quadratic error function

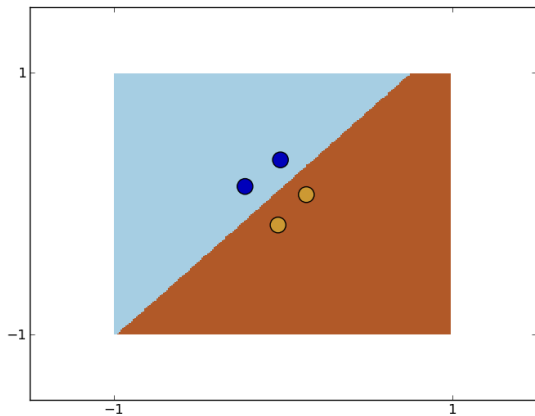
$$E_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2} \sum^n (\mathbf{z}_n - \mathbf{W}\mathbf{x})^T (\mathbf{z}_n - \mathbf{W}\mathbf{x}),$$

that resulted from log likelihood on Gaussians.

$\implies$  Can find exact closed form solution for  $\mathbf{W}$  using pseudo-inverse, as before.

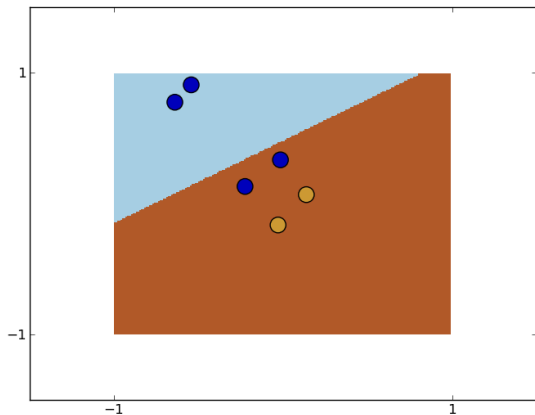
**Decision rule:** Train on the 1-of-K targets, then use class with highest value as prediction.

## Least squares: Two classes



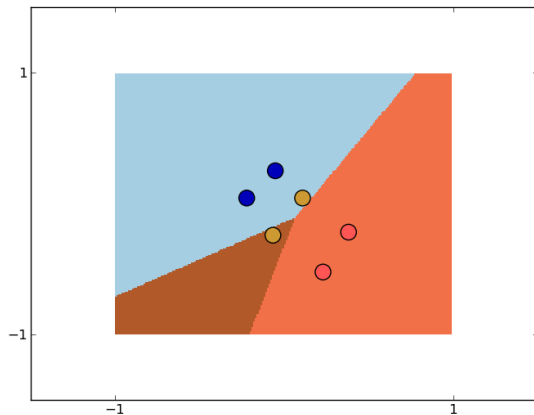
Looks good.

## Least squares: Two classes



Outliers screw it up. There clearly is a line that would be better!

## Least squares: Three classes



This could be separated by two lines – but only some star like thing is used.

# What went wrong?

Predictions not in the class space

- ▶ don't sum to 1,
- ▶ negative values possible.

Assumption of Gaussian noise is wrong.

- ▶ class space binary,
- ▶ labels not noisy.

Let's start over properly: Write down the likelihood  $\mathcal{L}$  of the data  $\mathcal{D}$  given a model's parameters  $\mathbf{W}$  ...

$$\begin{aligned}\mathcal{L}(\mathbf{W}) &= p(\mathcal{D}|\mathbf{W}) \\ &= \prod_i p(x_i, C_i|\mathbf{W}) \\ &= \prod_i p(C_i|x_i, \mathbf{W})p(x_i)\end{aligned}$$

Let's start over properly: Write down the likelihood  $\mathcal{L}$  of the data  $\mathcal{D}$  given a model's parameters  $\mathbf{W}$  ...

$$\begin{aligned}\mathcal{L}(\mathbf{W}) &= p(\mathcal{D}|\mathbf{W}) \\ &= \prod_i p(x_i, C_i|\mathbf{W}) \\ &= \prod_i p(C_i|x_i, \mathbf{W})p(x_i)\end{aligned}$$

... and then state the learning problem.

$$\operatorname{argmax}_{\mathbf{W}} \prod_i p(C_i|x_i, \mathbf{W})$$

We ignore  $p(x_i)$ , since we don't model it and it is just a constant factor then.

# Three Approaches to Classification

Discriminant function: Function maps input to class

$$f : X \rightarrow \{C_k\}.$$

# Three Approaches to Classification

Discriminant function: Function maps input to class

$$f : X \rightarrow \{C_k\}.$$

Discriminative model: Probability of class given input:

$$p(C_k|\mathbf{x}).$$

# Three Approaches to Classification

**Discriminant function:** Function maps input to class

$$f : \mathcal{X} \rightarrow \{\mathcal{C}_k\}.$$

**Discriminative model:** Probability of class given input:

$$p(\mathcal{C}_k|\mathbf{x}).$$

**Generative model:** Combine conditional densities with class prior probabilities via Bayes':

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

# Probabilistic Generative Models

Idea: Use  $p(C_k)$  and  $p(\mathbf{x}|C_k)$  and Bayes to get  $p(C_k|\mathbf{x})$ .

$K = 2$  classes:

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

# Probabilistic Generative Models

Idea: Use  $p(C_k)$  and  $p(\mathbf{x}|C_k)$  and Bayes to get  $p(C_k|\mathbf{x})$ .

$K = 2$  classes:

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

$K > 2$  classes:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)}.$$

## Model for the class priors

How do we model

$$p(C_k)?$$

Use categorical distribution. For each of the classes  $1, \dots, k$  there is one parameter  $0 \leq \theta_i \leq 1$ . Furthermore  $\sum_i \theta_i = 1$ . So

$$p(C_k = i | \boldsymbol{\theta}) = \theta_i.$$

## Model for the class priors

How do we model

$$p(C_k)?$$

Use categorical distribution. For each of the classes  $1, \dots, k$  there is one parameter  $0 \leq \theta_i \leq 1$ . Furthermore  $\sum_i \theta_i = 1$ . So

$$p(C_k = i | \boldsymbol{\theta}) = \theta_i.$$

Or use a uniform distribution: each class has equal probability. What does this mean for our learning problem?

This will prove convenient later on:

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

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

## Class conditionals

We have to make assumptions about the class conditional densities. Let's choose a multivariate Gaussian:

$$p(\mathbf{x}|C_k) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

Note: each class has a different  $\boldsymbol{\mu}_k$ , but all share the same covariance matrix!

## Class conditionals

We have to make assumptions about the class conditional densities. Let's choose a multivariate Gaussian:

$$p(\mathbf{x}|C_k) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

Note: each class has a different  $\boldsymbol{\mu}_k$ , but all share the same covariance matrix!

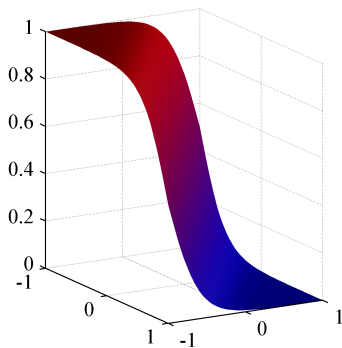
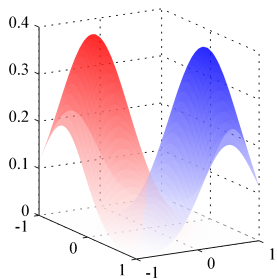
For the case with  $K = 2$  we have

$$\begin{aligned} p(C_1|\mathbf{x}) &= (1 + \exp(\mathbf{w}^T \mathbf{x} + w_0))^{-1} \\ &= \sigma(\mathbf{w}^T \mathbf{x} + w_0) \end{aligned}$$

with

$$\begin{aligned} \mathbf{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)} \end{aligned}$$

Note: we only have a single output for two classes and the bias  $w_0$  is not part of the parameter vector  $\mathbf{w}$  here.



$K > 2$  classes:  $a_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_0$  with

$$\begin{aligned}\mathbf{w}_k &= \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k \\ w_{0k} &= -\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \ln p(C_k)\end{aligned}$$

Plug it into:

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

Can you see a quick decision rule? In what situation is this not convenient?

## Getting the parameters right – maximum likelihood

Wanted:

$$\begin{aligned} \operatorname{argmax}_{\theta, \mathbf{w}} \mathcal{L}(\theta, \mathbf{W}) \\ = \operatorname{argmax}_{\theta, \mathbf{w}} \prod_i p(\mathbf{x} | C_k, \mathbf{W}) p(C_k | \theta) \end{aligned}$$

We can do this separately since  $p(\mathbf{x} | C_k, \mathbf{W})$  and  $p(C_k | \theta)$  depend on different parameters.

## Getting the parameters right – maximum likelihood

Wanted:

$$\begin{aligned} \operatorname{argmax}_{\theta, \mathbf{W}} \mathcal{L}(\theta, \mathbf{W}) \\ = \operatorname{argmax}_{\theta, \mathbf{W}} \prod_i p(\mathbf{x} | \mathcal{C}_k, \mathbf{W}) p(\mathcal{C}_k | \theta) \end{aligned}$$

We can do this separately since  $p(\mathbf{x} | \mathcal{C}_k, \mathbf{W})$  and  $p(\mathcal{C}_k | \theta)$  depend on different parameters.

Use maximum likelihood estimation for the individual components:

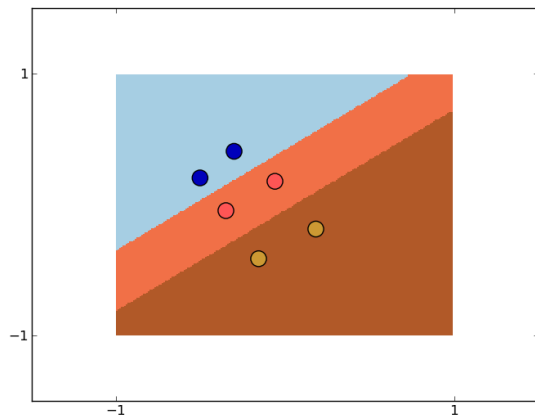
- ▶  $p(\mathcal{C}_k) \approx \frac{1}{|\mathcal{D}|} \sum_i \mathbb{I}(z_i = \mathcal{C}_k)$
- ▶  $p(\mathbf{x} | \mathcal{C}_k)$  – previous lecture!

where  $\mathbb{I}(a)$  is the indicator function and evaluates to 0 if  $a$  is false, else 1.

# Estimate Parameters of Gaussian Class Conditionals with Global Covariance

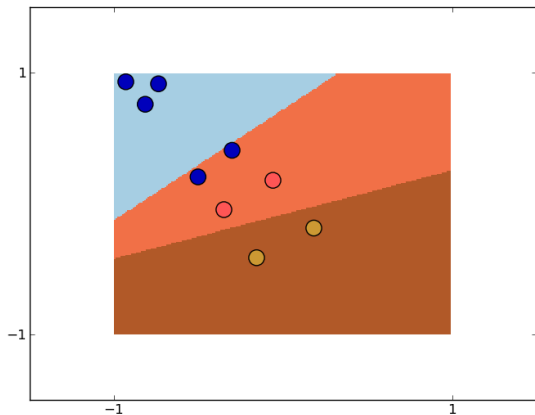
1. Estimate  $\mu_k$  for each class  $k$ ,
2. Calculate set  $C = \{x_i - \mu_{C_i}\}$  where each from data point the mean of the corresponding class is subtracted,
3. Estimate global  $\Sigma$  from  $C$ .

## Generative: Three classes



Separation better.

## Generative: Three classes



Still prone to data with classes covering differently sized areas.

	LSQ	Generative
Model	$\mathbf{W}\mathbf{x}$	$p(\mathcal{C}_k) = \text{uniform},$ $p(x \mathcal{C}_k) = \mathcal{N}(x \boldsymbol{\mu}, \boldsymbol{\Sigma})$
Decision	$\text{argmax}_i(\mathbf{w}_i^T \mathbf{x})$	$\text{argmax}_i(\mathbf{w}_i^T \mathbf{x})$
Assumptions	$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathbf{x} k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}),$ $k \sim \text{uniform}$

Different approaches can lead to the same decision rule but with different parameters.

**Idea:** The outputs of our generative model are in  $[0, 1]$  and sum to 1. Use the same model to model categorical variables.

**Idea:** The outputs of our generative model are in  $[0, 1]$  and sum to 1. Use the same model to model categorical variables.

$$p(C_k|\mathbf{x}) = \text{Cat}(C_k|\mathbf{x}_i, \mathbf{W})$$

**Idea:** The outputs of our generative model are in  $[0, 1]$  and sum to 1. Use the same model to model categorical variables.

$$\begin{aligned} p(C_k|\mathbf{x}) &= \text{Cat}(C_k|\mathbf{x}_i, \mathbf{W}) \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

**Idea:** The outputs of our generative model are in  $[0, 1]$  and sum to 1. Use the same model to model categorical variables.

$$\begin{aligned} p(C_k|\mathbf{x}) &= \text{Cat}(C_k|\mathbf{x}_i, \mathbf{W}) \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \\ &= \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T \mathbf{x})} \end{aligned}$$

Same model as before, but different assumptions  $\rightarrow$  discriminative model, not generative.

For 2 classes, this formula simplifies to  $\frac{1}{1+\exp(\mathbf{w}^T \mathbf{x})} =: \sigma(\mathbf{w}^T \mathbf{x})$  which is also known as the *logistic sigmoid function*.

# Likelihood of Logistic Regression

This model is called **logistic regression**. Although it's not regression.

# Likelihood of Logistic Regression

This model is called **logistic regression**. Although it's not regression. Let's write down the likelihood, as always.

$$\begin{aligned}\mathcal{L}(\mathbf{W}) &= p(\mathcal{D}|\mathbf{W}) = \prod_i p(\mathbf{x}_i, C_i|\mathbf{W}) \\ &= \prod_i p(C_i|\mathbf{x}_i, \mathbf{W})p(\mathbf{x}_i) \\ &= \prod_i \prod_k p(\mathbf{y}_i = C_k|\mathbf{x}_i, \mathbf{W})^{\mathbb{I}(\mathbf{y}_i=C_k)} p(\mathbf{x}_i)\end{aligned}$$

The exponent with the indicator function makes all factors where this is not met go to 1. Why can we do that?

# The learning problem

We use the logarithm again to get sums instead of products. Also, we can drop  $p(x_i)$  since it is not important for optimization with respect to  $\mathbf{W}$ .

$$\begin{aligned}\log \mathcal{L}(\mathbf{W}) &= \sum_i \sum_{j,k} \mathbf{z}_{ij} \log p(y = \mathcal{C}_k | \mathbf{x}_i, \mathbf{W}) \\ &= \sum_i \sum_{j,k} \mathbf{z}_{ij} \log \frac{\exp(\mathbf{w}_j^T \mathbf{x}_i)}{\sum_l \exp(\mathbf{w}_l^T \mathbf{x}_i)} \\ &= \sum_i \sum_{j,k} \mathbf{z}_{ij} [\mathbf{w}_j^T \mathbf{x}_i - \log \sum_l \exp(\mathbf{w}_l^T \mathbf{x}_i)]\end{aligned}$$

→ Minimize the negative log likelihood with respect to the model parameters. This is also known as the **cross-entropy error function**.  
Btw: it is an upper bound to the actual misclassification rate.

No closed-form as with the pseudo inverse for linear regression.

**Instead:** Gradient descent ( $\rightarrow$  online linear regression).

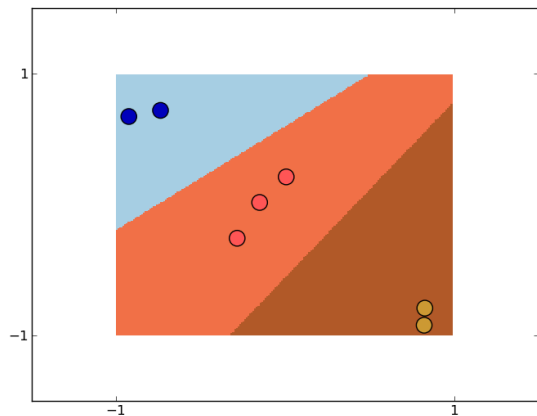
But the cross-entropy is **convex**, which means that there is a unique minimum that is easy to find.

## Gradient for logistic regression

$$\frac{\partial \mathcal{L}(\mathbf{W})}{\partial \mathbf{W}} = \sum_{i=1}^{|\mathcal{D}|} (\mathbf{y}_i - \mathbf{z}_i) \mathbf{x}_i$$

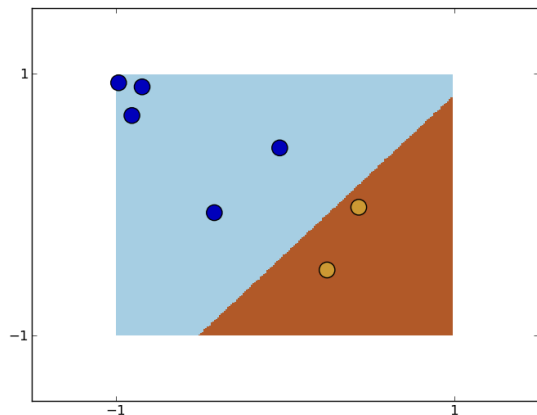
where the  $\mathbf{z}_i$  are 1-of-K vectors and  $\mathbf{y}_i = \frac{\exp(\mathbf{w}_j^T \mathbf{x}_i)}{\sum_l \exp(\mathbf{w}_l^T \mathbf{x}_i)}$ .

## Logistic Regression: Three classes



Logistic regression can handle this gracefully.

## Logistic Regression: points far away



Handles this far better.

	LSQ	Generative	Logistic Regression
Like Model	$\mathbf{W}^T \mathbf{x}$	$p(C_k) = \text{uniform},$ $p(x C_k) = \mathcal{N}(x \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$p(C_k \mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T \mathbf{x})}$
Decision	$\text{argmax}_i(\mathbf{w}_i^T \mathbf{x})$	$\text{argmax}_i(\mathbf{w}_i^T \mathbf{x})$	$\text{argmax}_i(\mathbf{w}_i^T \mathbf{x})$
Assumptions	$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathbf{x} k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}),$ $k \sim \text{uniform}$	$k \sim \text{Cat}(\boldsymbol{\theta})$

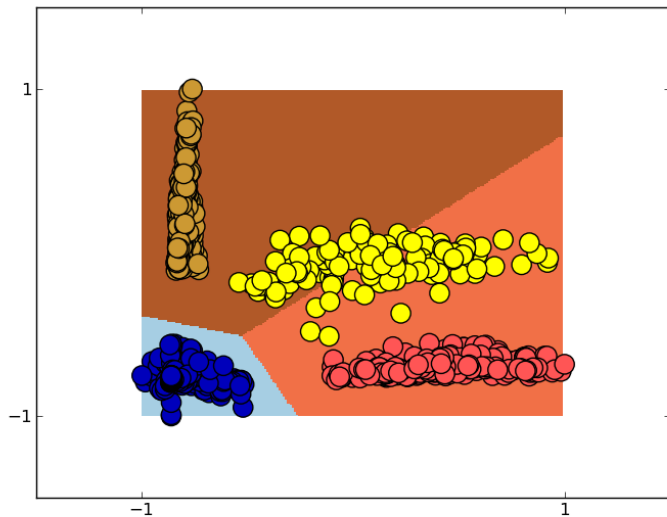
Different approaches can lead to the same decision rule but with different parameters and different quality.

## Take away messages

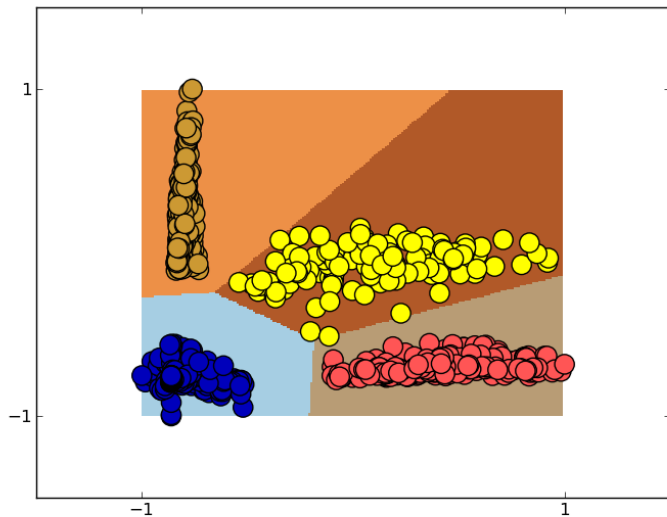
- ▶ Least squares for classification performs poorly – but we can solve it exactly very fast.
- ▶ Generative models allow you to include knowledge about the system you are modelling – if you have an idea on how the classes are distributed, you can choose  $p(\mathbf{x}|\mathcal{C})$  and  $p(\mathcal{C})$  accordingly.
- ▶ Logistic regression is robust to the pathological cases we illustrated and is one of the most popular classifiers out there – Google uses it on a big scale!

There are, however, a lot more methods for classification.

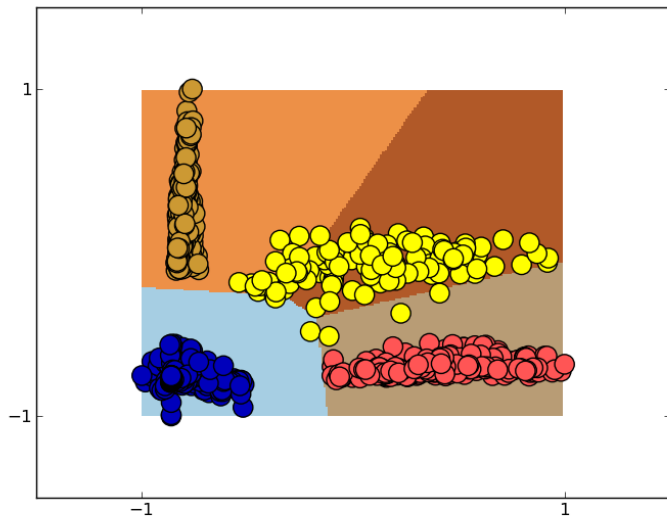
# sEMG: Least Squares



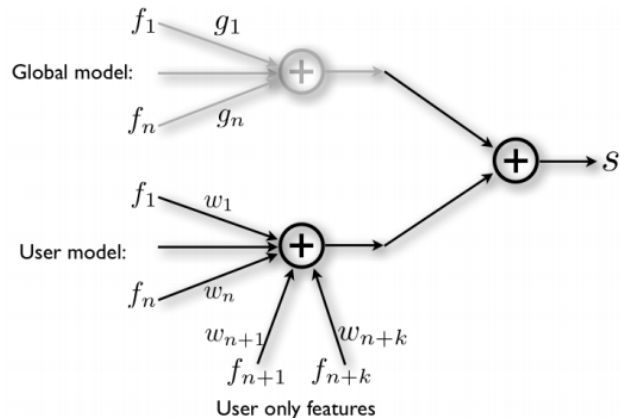
# sEMG: Generative Model



# sEMG: Logistic Regression



# How Google uses Logistic Regression



User data and global data are used to predict the importance of emails.