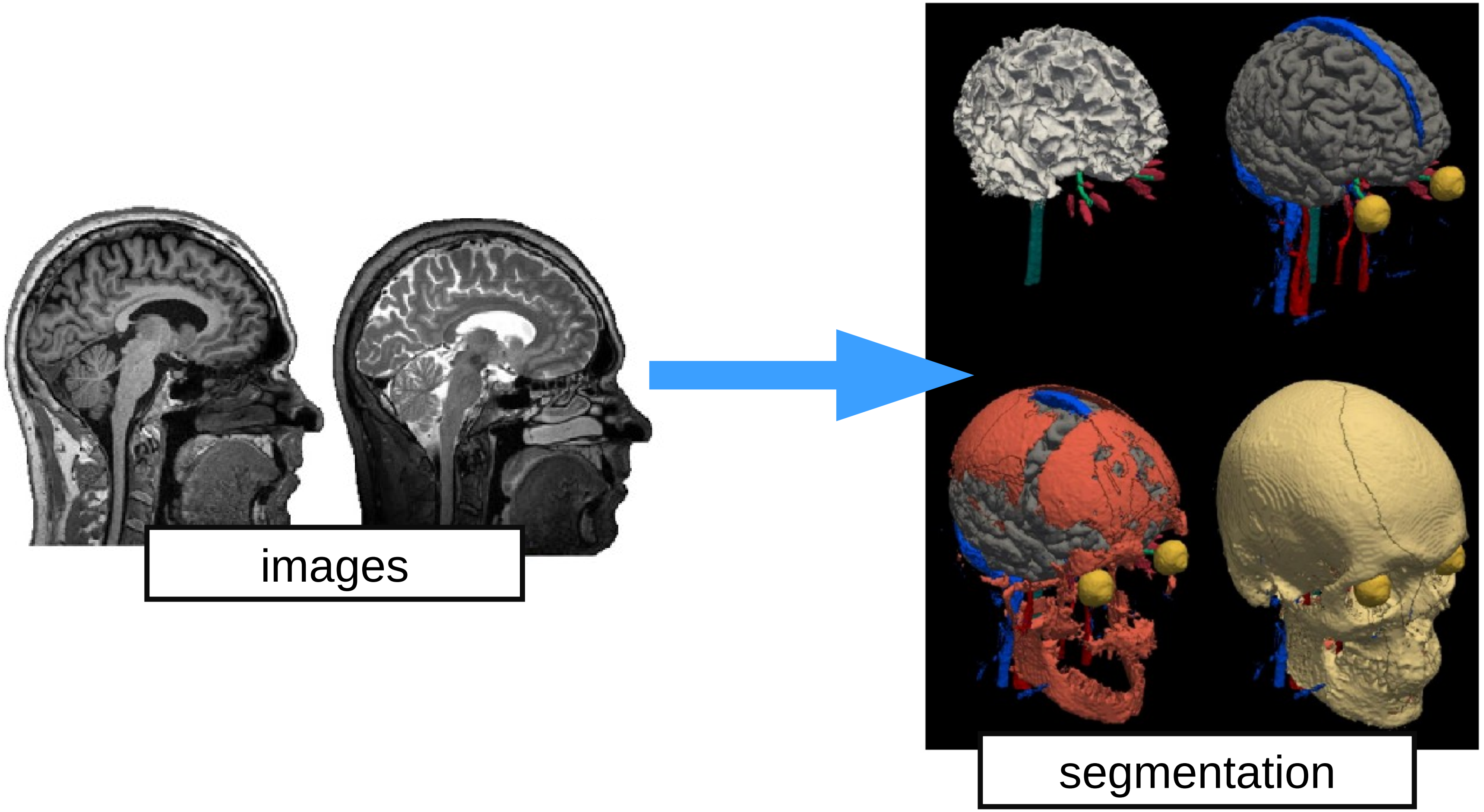
Neural Networks

Aalto-yliopisto
Aalto-universitetet
Aalto University

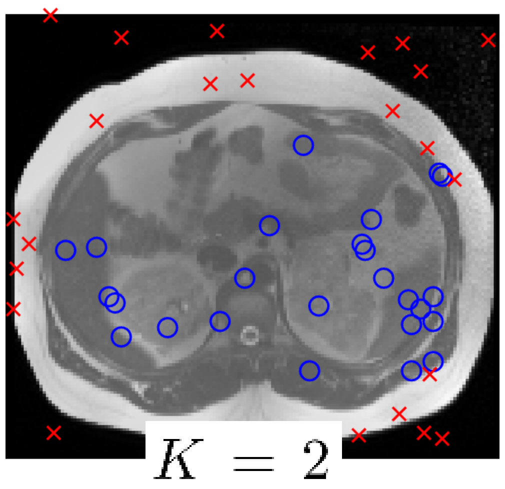
Medical Image Analysis
Koen Van Leemput
Fall 2024

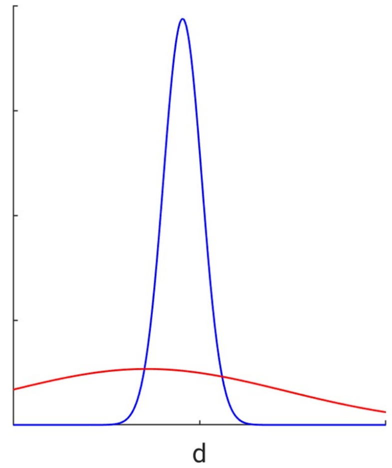
Focus: automatic segmentation

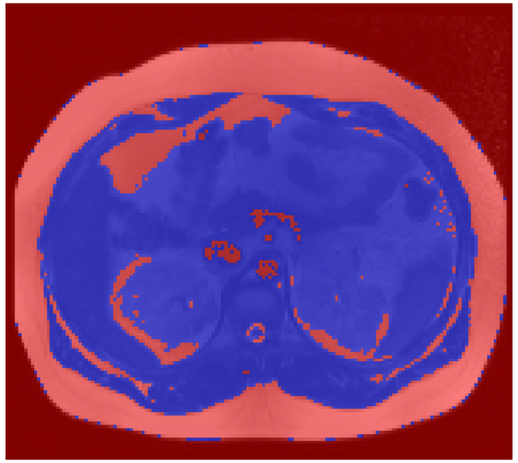


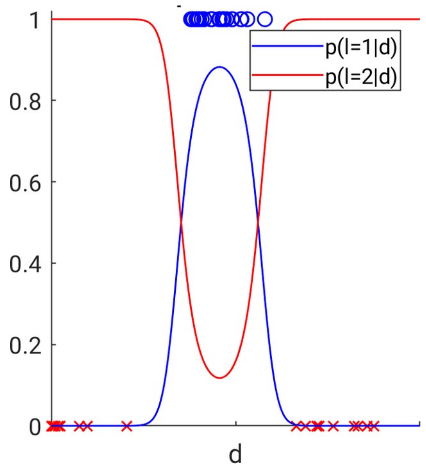
Aalto-yliopisto
Aalto-universitetet
Aalto University

Remember the Gaussian mixture model?





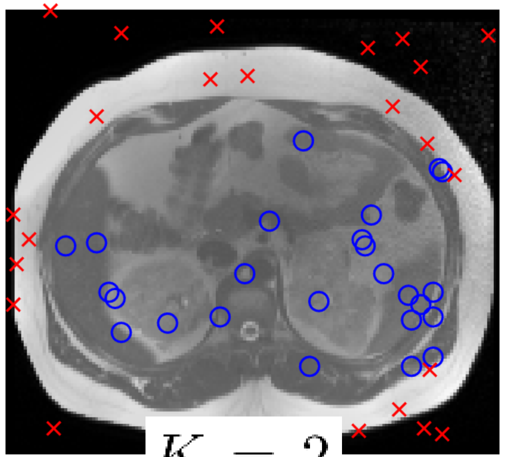


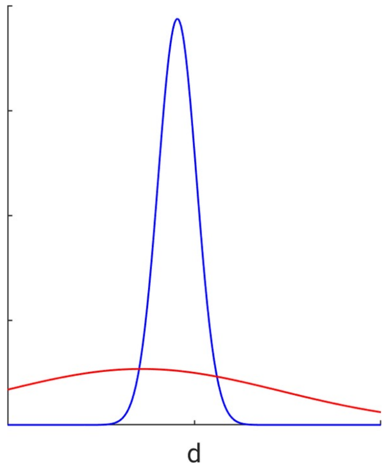


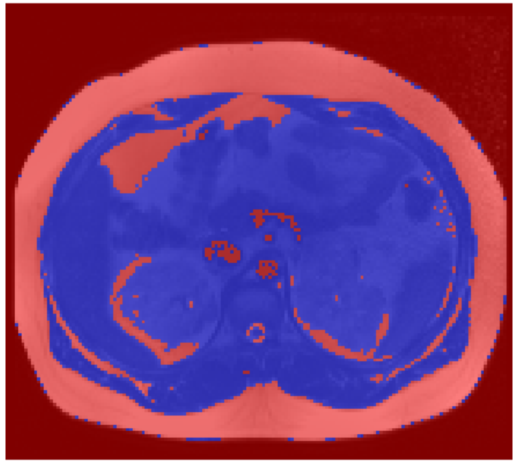
Posterior using Bayes' rule: $p(l = k|d, \boldsymbol{\theta}) = \dfrac{\mathcal{N}(d|\mu_k, \sigma_k^2)\pi_k}{\sum_{k'} \mathcal{N}(d|\mu_{k'}, \sigma_{k'}^2)\pi_{k'}}$

Aalto-yliopisto
Aalto-universitetet
Aalto University

Remember the Gaussian mixture model?





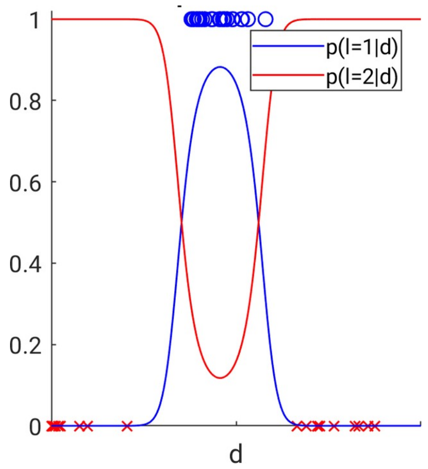


This lecture: can we get a "classifier"

$$p(l|d,\boldsymbol{\theta})$$

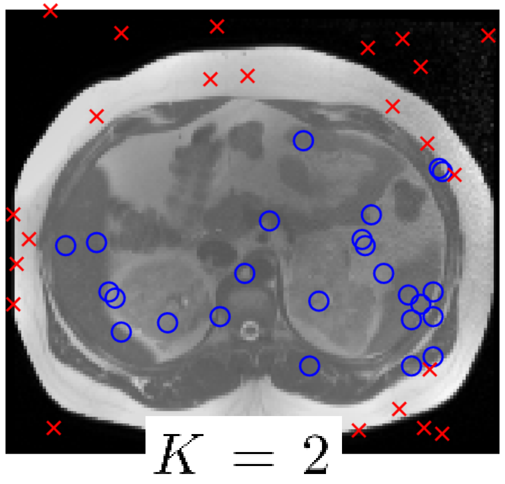
directly

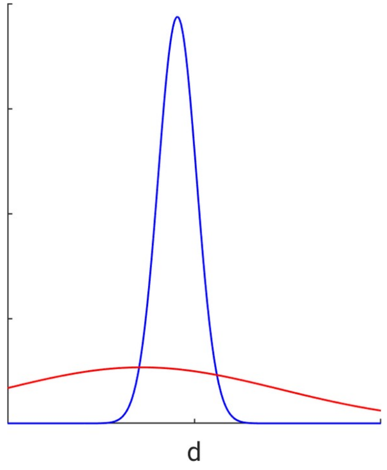
<u>without a model</u>?

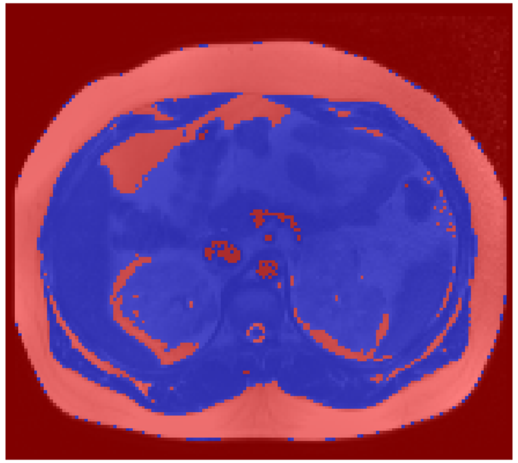


Posterior using Bayes' rule: $p(l=k|d,\boldsymbol{\theta}) = \dfrac{\mathcal{N}(d|\mu_k,\sigma_k^2)\pi_k}{\sum_{k'}\mathcal{N}(d|\mu_{k'},\sigma_{k'}^2)\pi_{k'}}$

Remember the Gaussian mixture model?





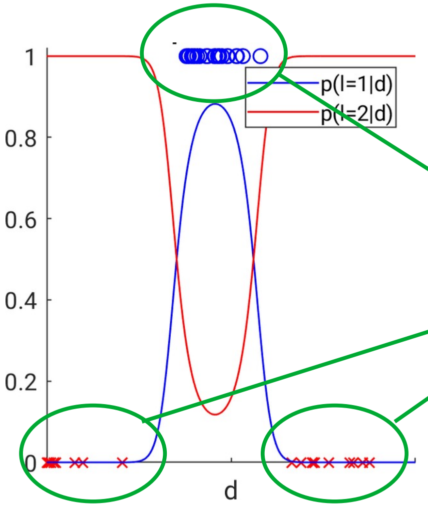


This lecture: can we get a "classifier"

$$p(l|d, \boldsymbol{\theta})$$

directly

without a model?



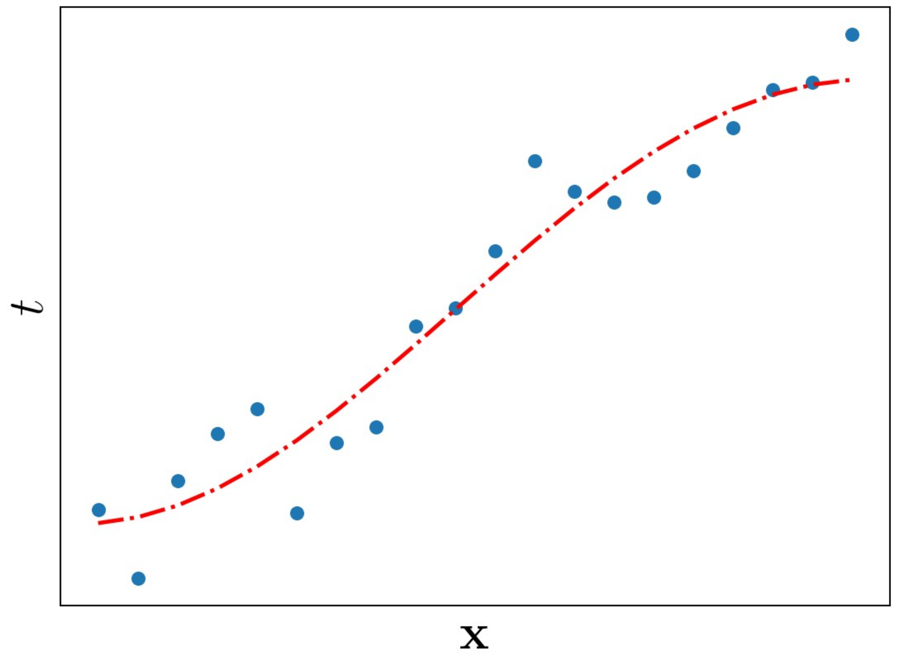
new notation/terminology:

- "training samples"
- "t=1" if l=1
- "t=0" if l=2

Posterior using Bayes' rule: $p(l = k|d, \boldsymbol{\theta}) = \dfrac{\mathcal{N}(d|\mu_k, \sigma_k^2)\pi_k}{\sum_{k'} \mathcal{N}(d|\mu_{k'}, \sigma_{k'}^2)\pi_{k'}}$

Remember linear regression?

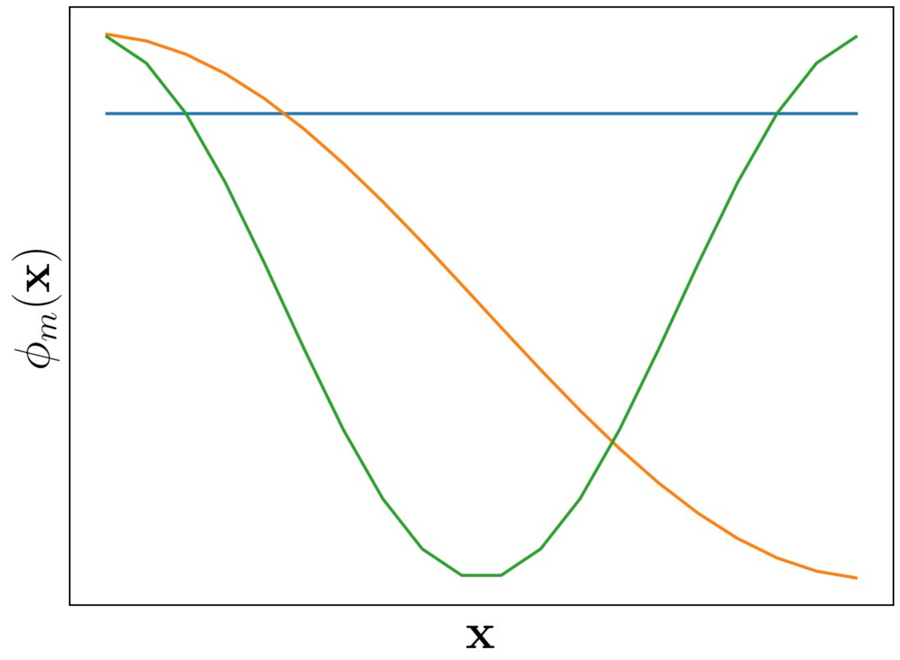
- Let $\mathbf{x} = (x_1, \ldots, x_D)^{\mathrm{T}}$ denote an input vector in a D-dimensional space
- Given N measurements $\{t_n\}_{n=1}^{N}$ at inputs $\{\mathbf{x}_n\}_{n=1}^{N}$, what is t at a new input $\mathbf{x}$?



$$y(\mathbf{x}, \mathbf{w}) = \sum_{m=0}^{M-1} w_m \phi_m(\mathbf{x})$$

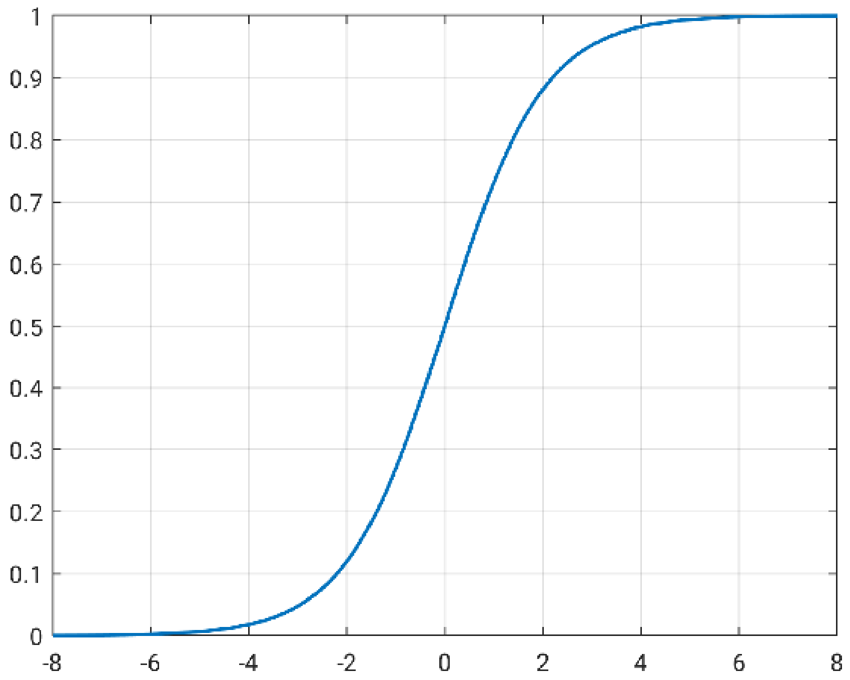
tunable weights

nonlinear basis functions



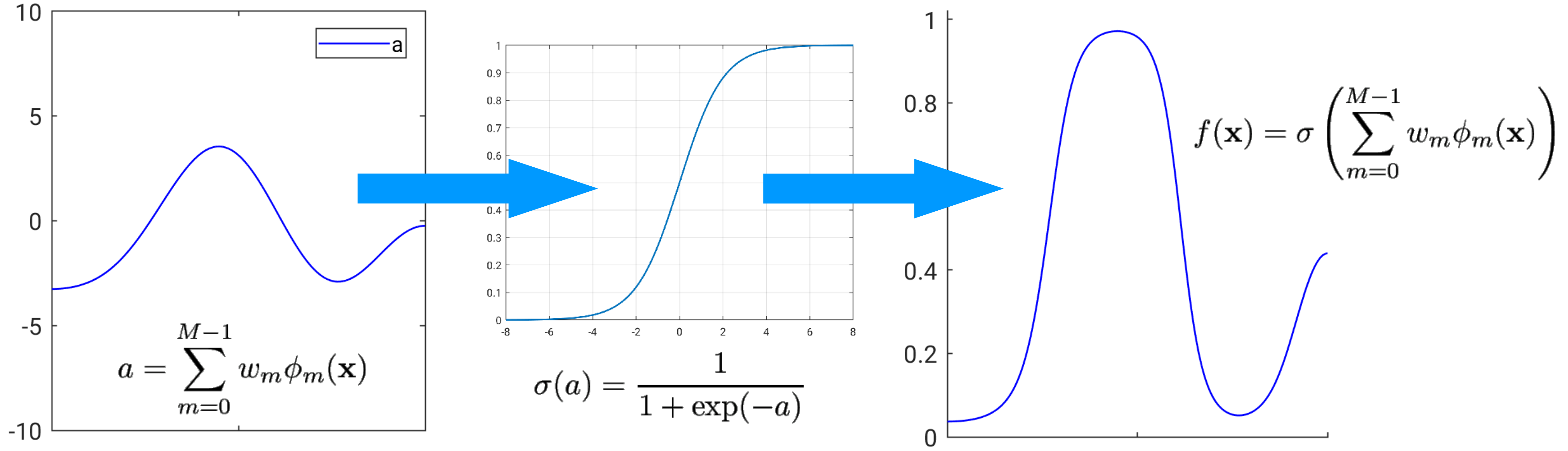
Logistic regression

– Logistic function as a “squashing” function



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

Logistic regression



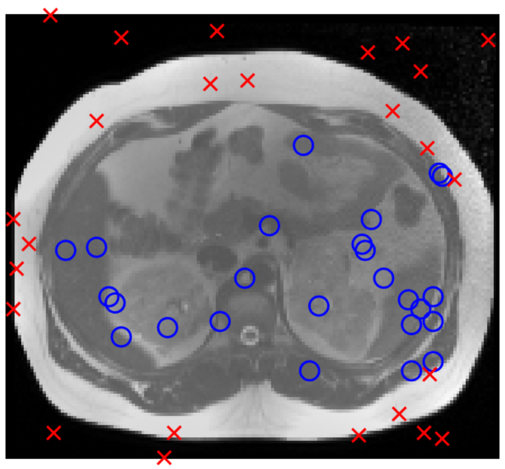
regression for binary outcomes:

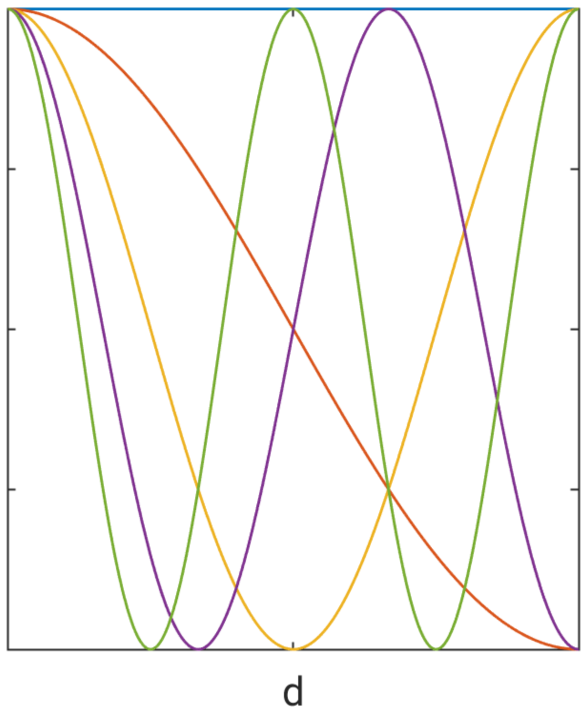
$t \in \{0, 1\}$

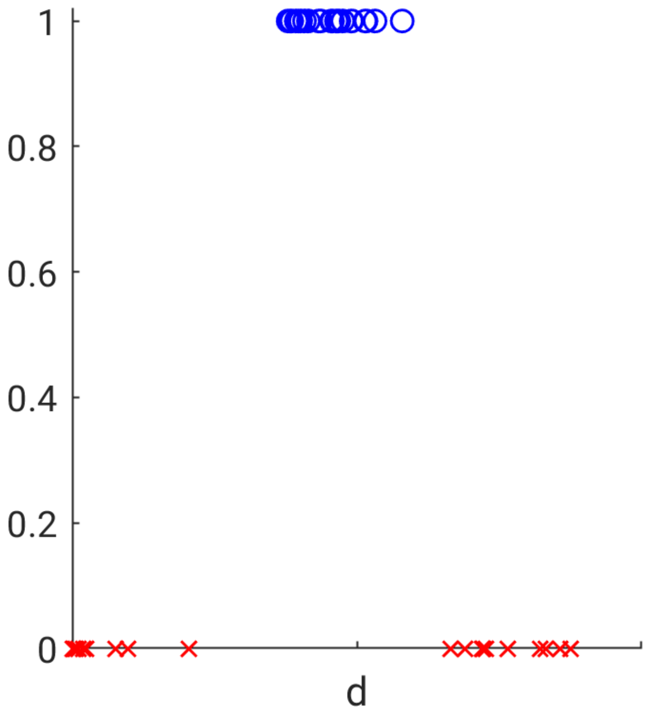
- $p(t=1|\mathbf{x},\boldsymbol{\theta}) = f(\mathbf{x})$ where $f(\mathbf{x}) = \sigma\left(\sum_{m=0}^{M-1} w_m \phi_m(\mathbf{x})\right)$
- Of course: $p(t=0|\mathbf{x},\boldsymbol{\theta}) = 1 - p(t=1|\mathbf{x},\boldsymbol{\theta}) = 1 - f(\mathbf{x})$

Voxel-based classifier

- Training data $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$ with $\mathbf{x}_n = d_n$ (i.e., $D = 1$) and $t_n \in \{0, 1\}$
- Estimate parameters $\boldsymbol{\theta} = (w_0, \ldots, w_{M-1})^{\mathrm{T}}$ by maximizing the likelihood $\prod_{n=1}^{N} p(t_n | \mathbf{x}_n, \boldsymbol{\theta})$

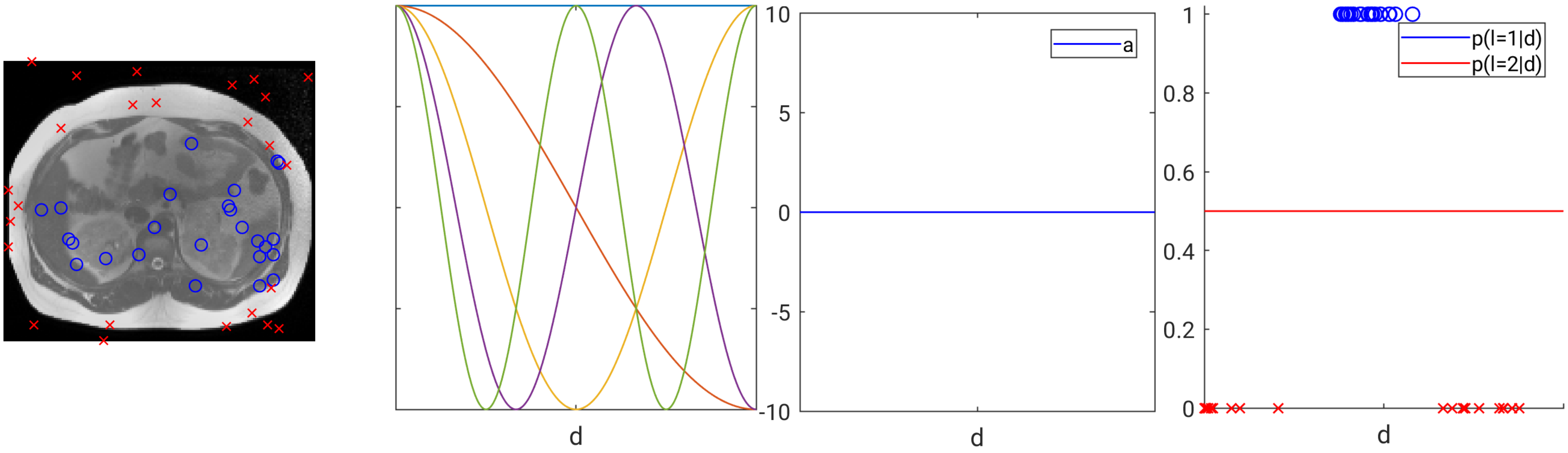






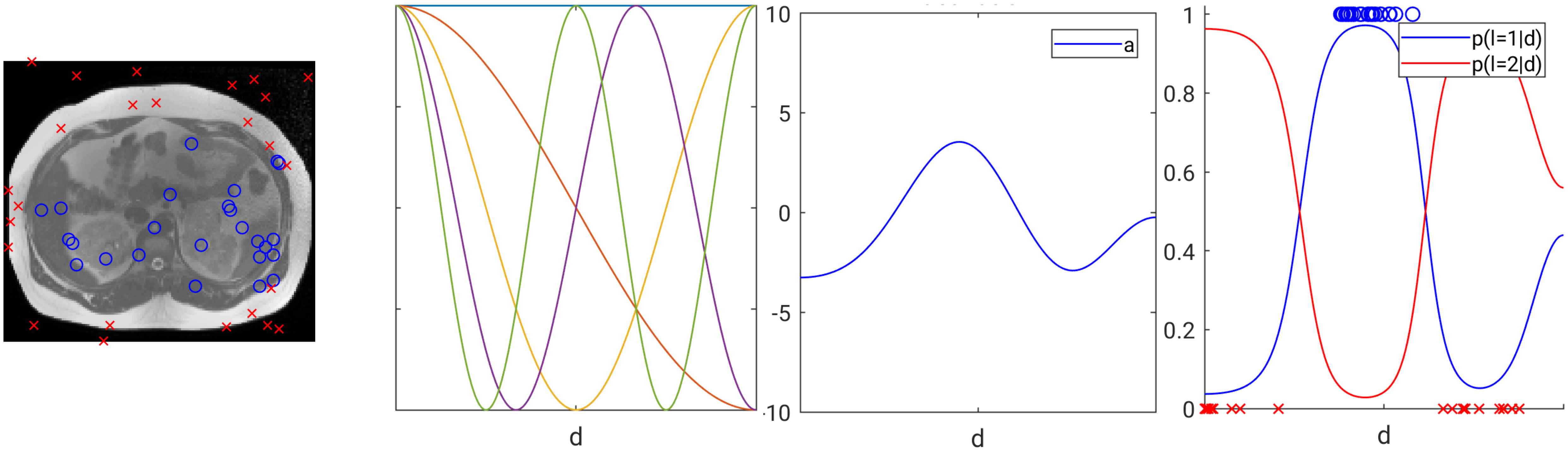
Voxel-based classifier

- Training data $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$ with $\mathbf{x}_n = d_n$ (i.e., $D = 1$) and $t_n \in \{0, 1\}$
- Estimate parameters $\boldsymbol{\theta} = (w_0, \ldots, w_{M-1})^{\mathrm{T}}$ by maximizing the likelihood $\prod_{n=1}^{N} p(t_n | \mathbf{x}_n, \boldsymbol{\theta})$



Voxel-based classifier

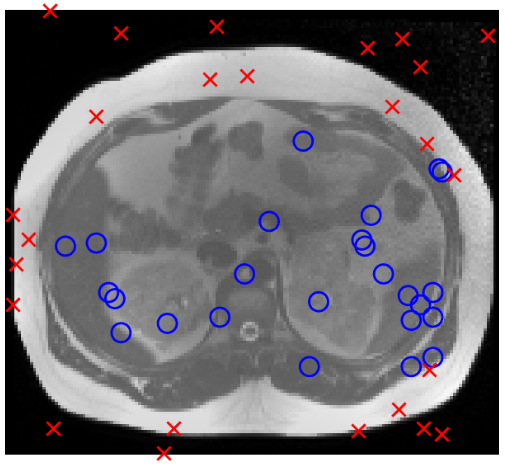
- Training data $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$ with $\mathbf{x}_n = d_n$ (i.e., $D = 1$) and $t_n \in \{0,1\}$
- Estimate parameters $\boldsymbol{\theta} = (w_0, \ldots, w_{M-1})^{\mathrm{T}}$ by maximizing the likelihood $\prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \boldsymbol{\theta})$

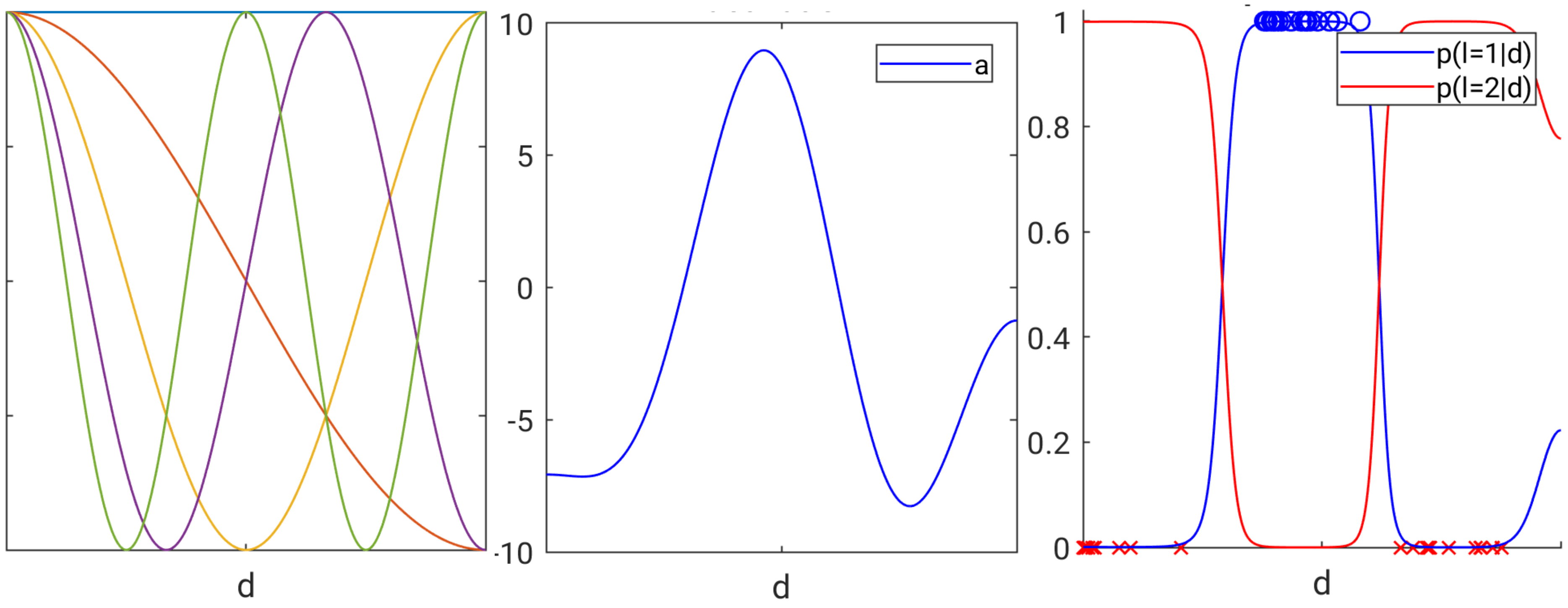


Voxel-based classifier

– Training data $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$ with $\mathbf{x}_n = d_n$ (i.e., $D = 1$) and $t_n \in \{0, 1\}$

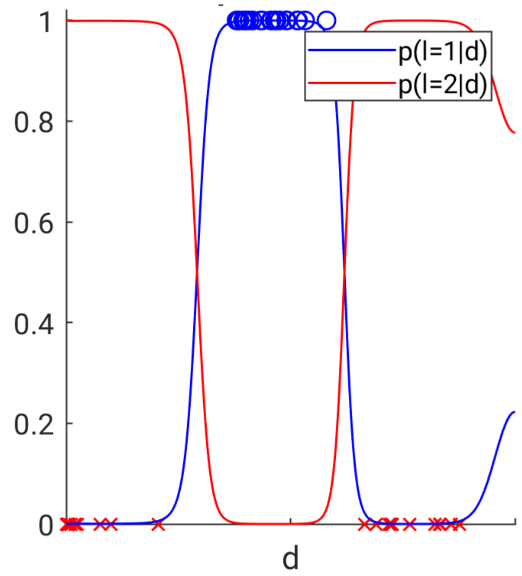
– Estimate parameters $\boldsymbol{\theta} = (w_0, \ldots, w_{M-1})^{\mathrm{T}}$ by maximizing the likelihood $\prod_{n=1}^{N} p(t_n | \mathbf{x}_n, \boldsymbol{\theta})$



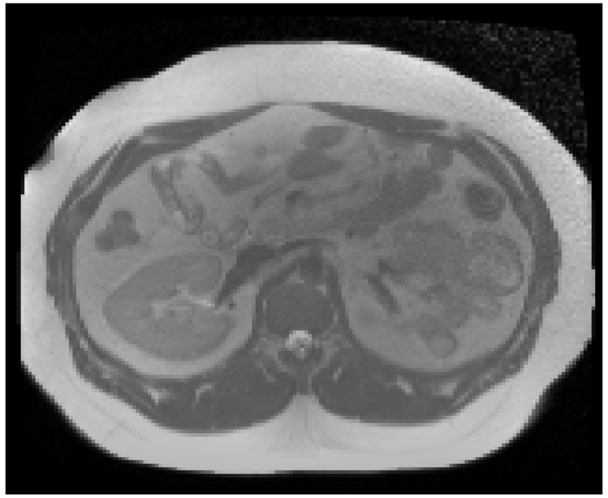


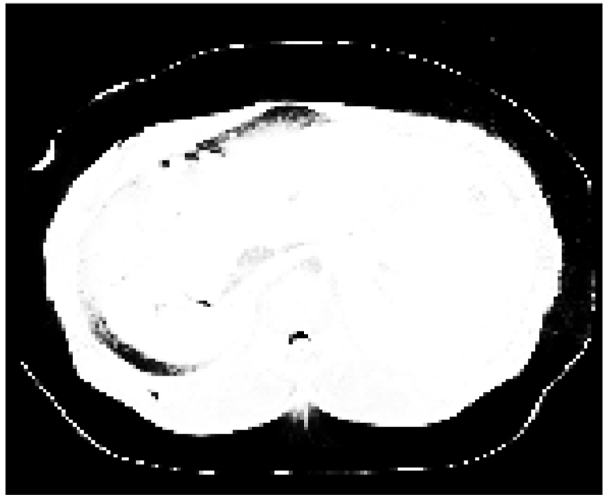
Voxel-based classifier

– Once trained keep the classifier: $p(l = 1|d, \hat{\boldsymbol{\theta}})$

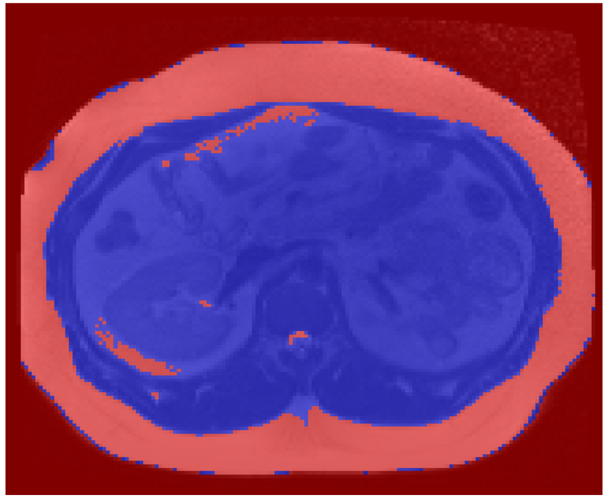


– Simply apply it to new data:





$p(l = 1|d, \hat{\boldsymbol{\theta}})$



$p(l = 1|d, \hat{\boldsymbol{\theta}}) > 0.5$

Aalto-yliopisto
Aalto-universitetet
Aalto University

Optimization algorithm for training

– Maximizing the likelihood function $\prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \boldsymbol{\theta})$ is equivalent to minimizing

$$E_N(\boldsymbol{\theta}) = -\log \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \boldsymbol{\theta}) = -\sum_{n=1}^{N} \{t_n \log f(\mathbf{x}_n) + (1-t_n)\log[1-f(\mathbf{x}_n)]\}$$

step size (user-specified)

– Gradient descent: $\boldsymbol{\theta}^{(\tau+1)} = \boldsymbol{\theta}^{(\tau)} - \nu \nabla E_N(\boldsymbol{\theta}^{(\tau)})$ with gradient $\nabla E_N(\boldsymbol{\theta}) = \frac{\partial E_N}{\partial \boldsymbol{\theta}}$

– Stochastic gradient descent: use only $N' \ll N$ randomly sampled training points, and approximate:

$$\nabla E_N(\boldsymbol{\theta}) \simeq \frac{N}{N'} \nabla E_{N'}(\boldsymbol{\theta})$$

More fun: patch-based classifier

- Classify 3x3 image "patches": intensity of the pixel to be classified + intensities of 8 neighboring pixels

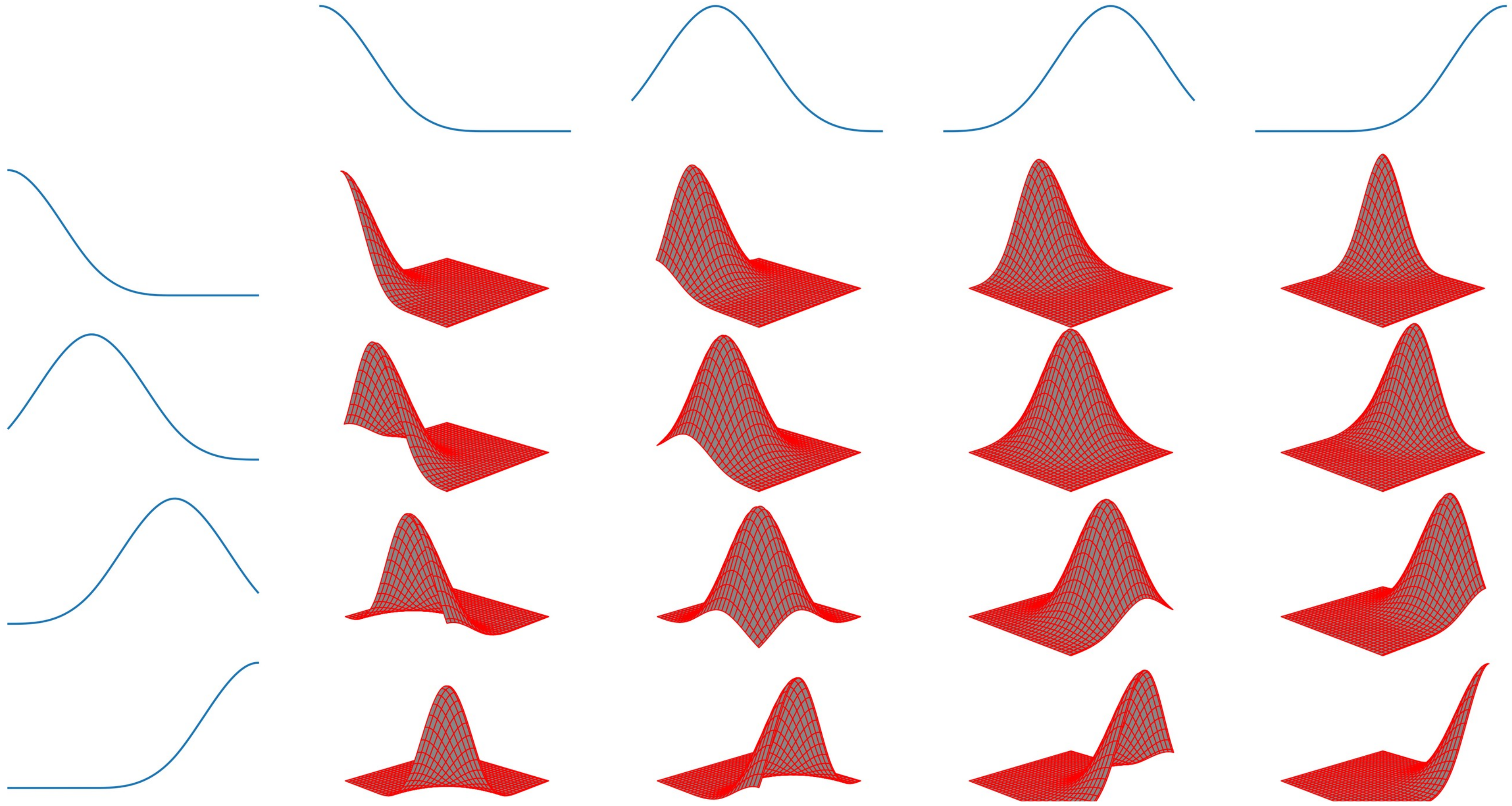
- $\mathbf{x}$ is now a 9-dimensional vector ($D = 9$), but otherwise everything is the same:

$$p(t=1|\mathbf{x}, \hat{\boldsymbol{\theta}}) = \sigma\left(\sum_{m=0}^{M-1} \hat{w}_m \phi_m(\mathbf{x})\right)$$

- But how to choose basis functions $\phi_m(\mathbf{x})$ in a 9-dimensional space?

Basis functions in high dimensions?

– Idea: remember the “separable basis functions” trick?



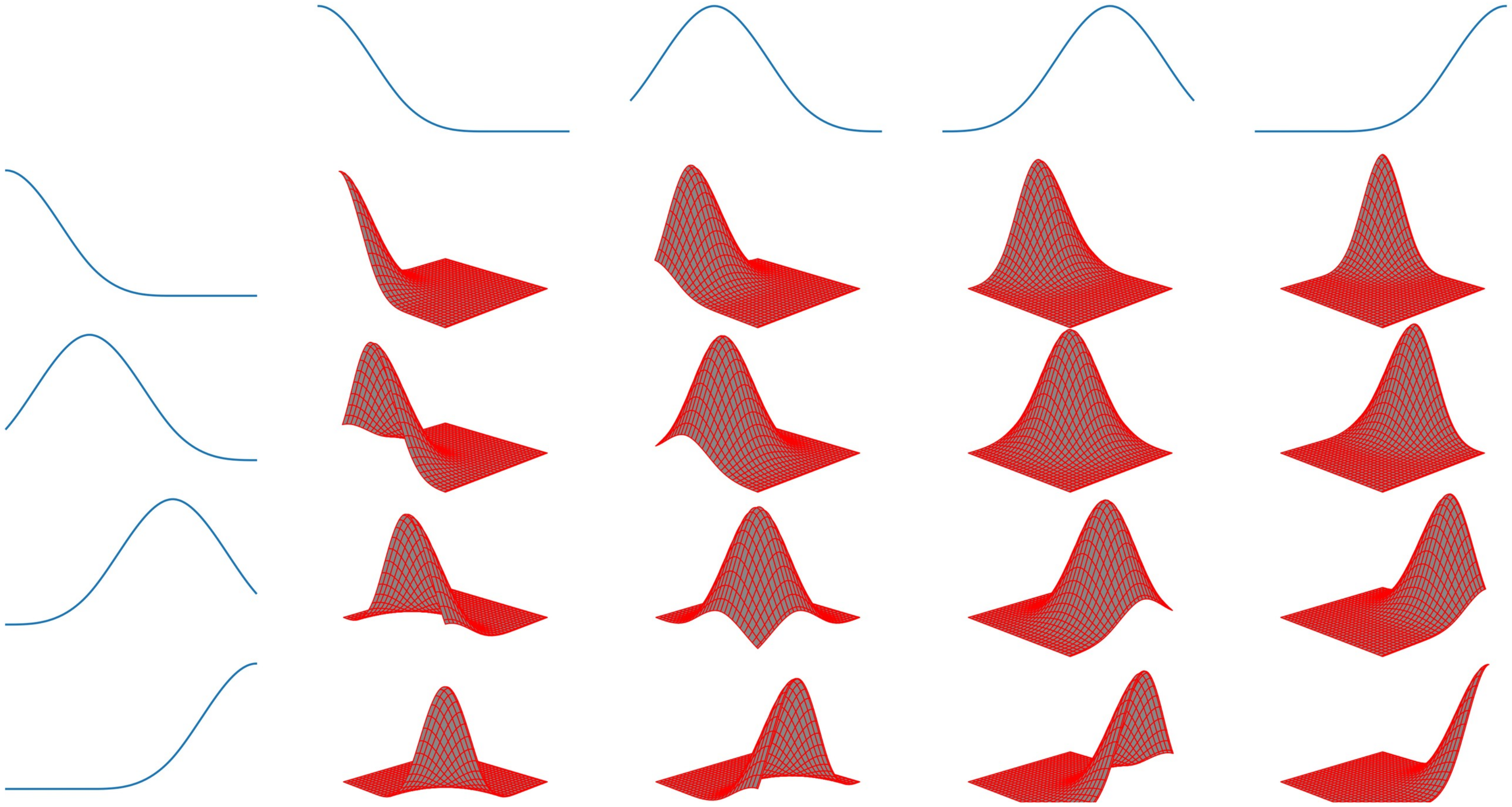
<u>Question</u>:

does this work in 9D?

“making” sixteen 2D basis functions
out of two sets of four 1D basis functions

Basis functions in high dimensions?

– Idea: remember the “separable basis functions” trick?



“making” sixteen 2D basis functions
out of two sets of four 1D basis functions

Question:

does this work in 9D?

No!
4^9 = 262144
basis functions!

Adaptive basis functions

- Introduce extra parameters that alter the ***form*** of a limited set of basis functions

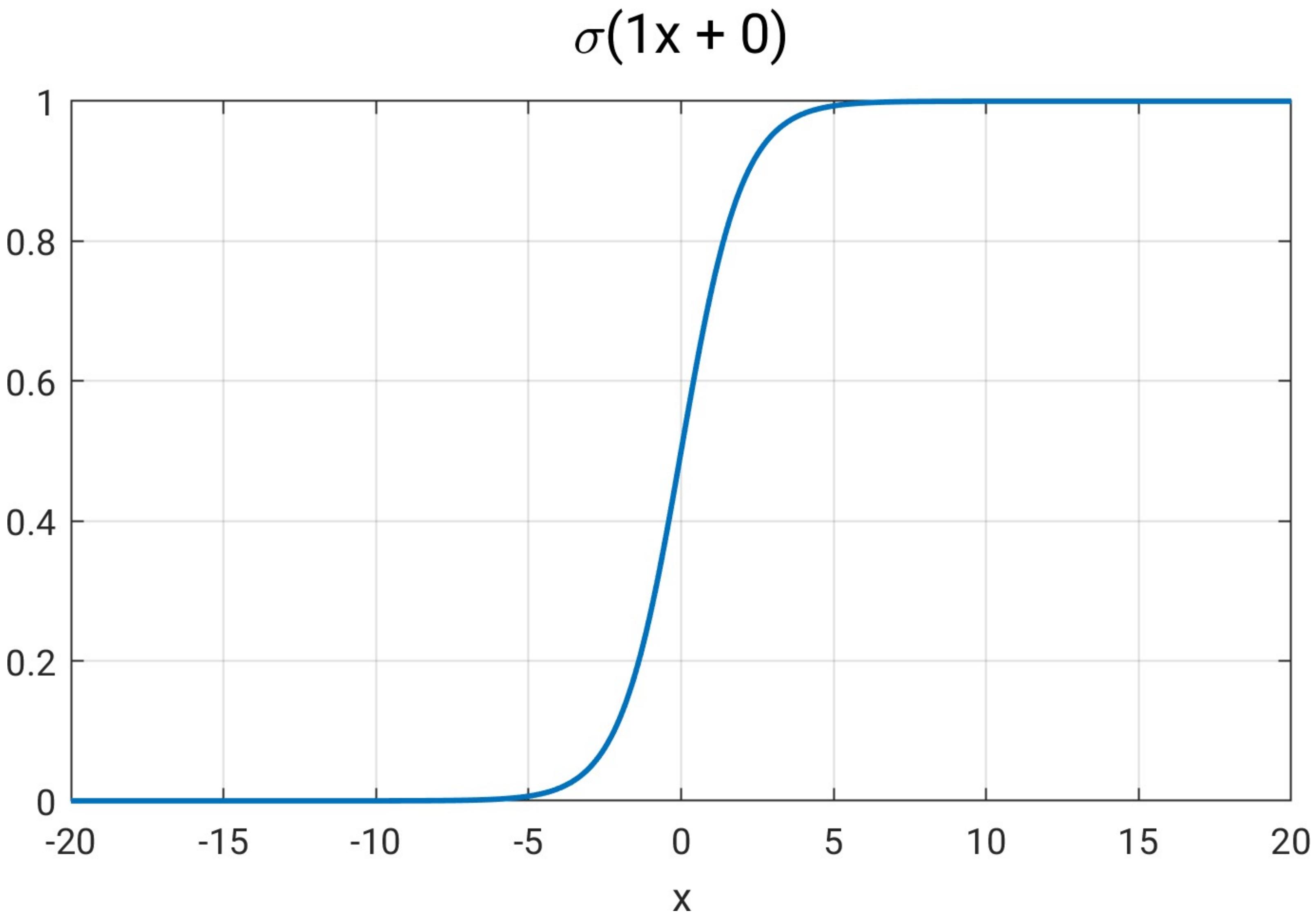
- Prototypical example:

$$\phi_m(\mathbf{x}) = \begin{cases} 1 & \text{if } m = 0, \\ \sigma\left(\sum_{d=1}^{D} \beta_{m,d} x_d + \beta_{m,0}\right) & \text{otherwise} \end{cases}$$

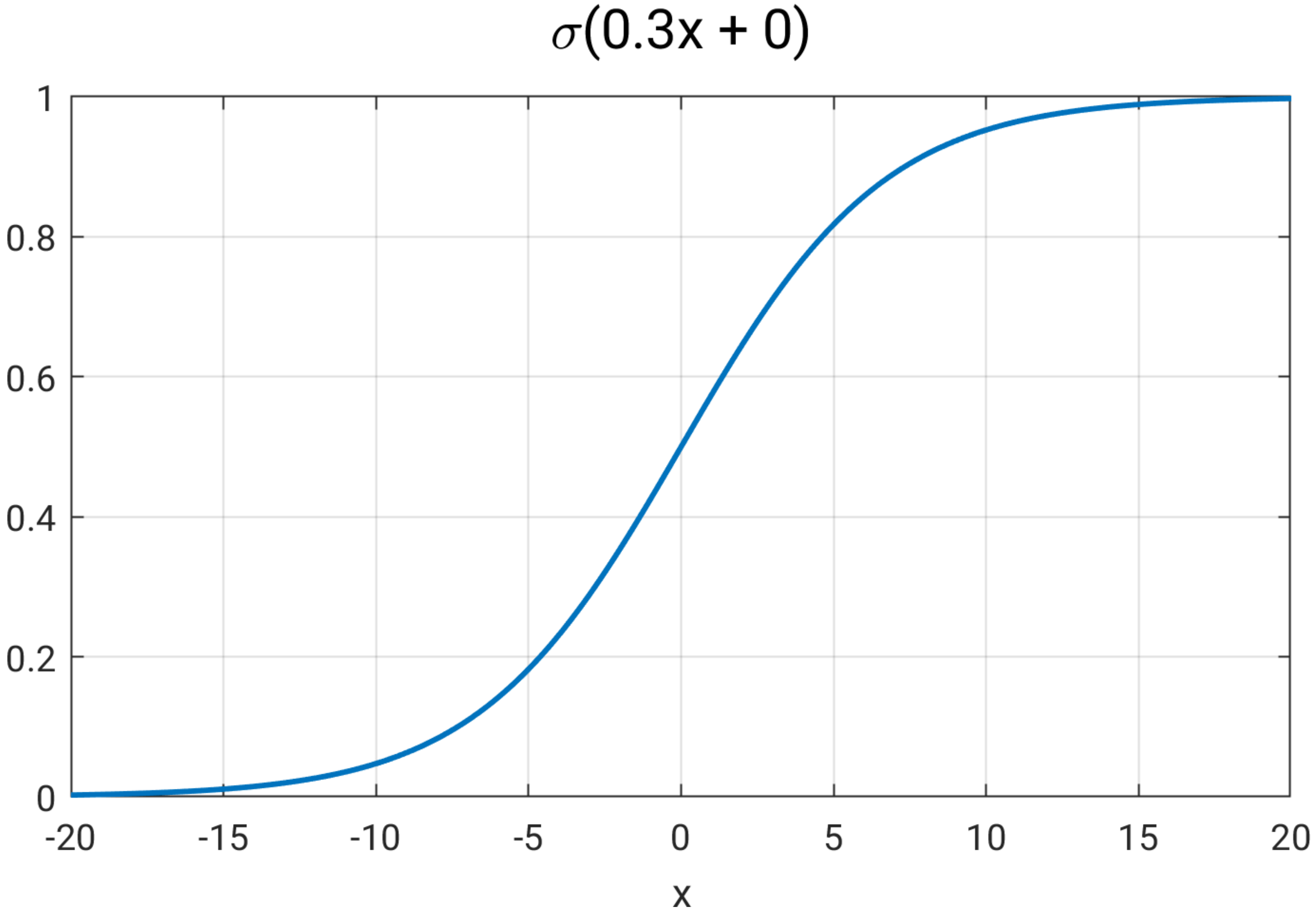
extra parameters

- All parameters ($\{\beta_{m,d}\}$ and $\{w_m\}$) are optimized together during training (stochastic gradient descent)

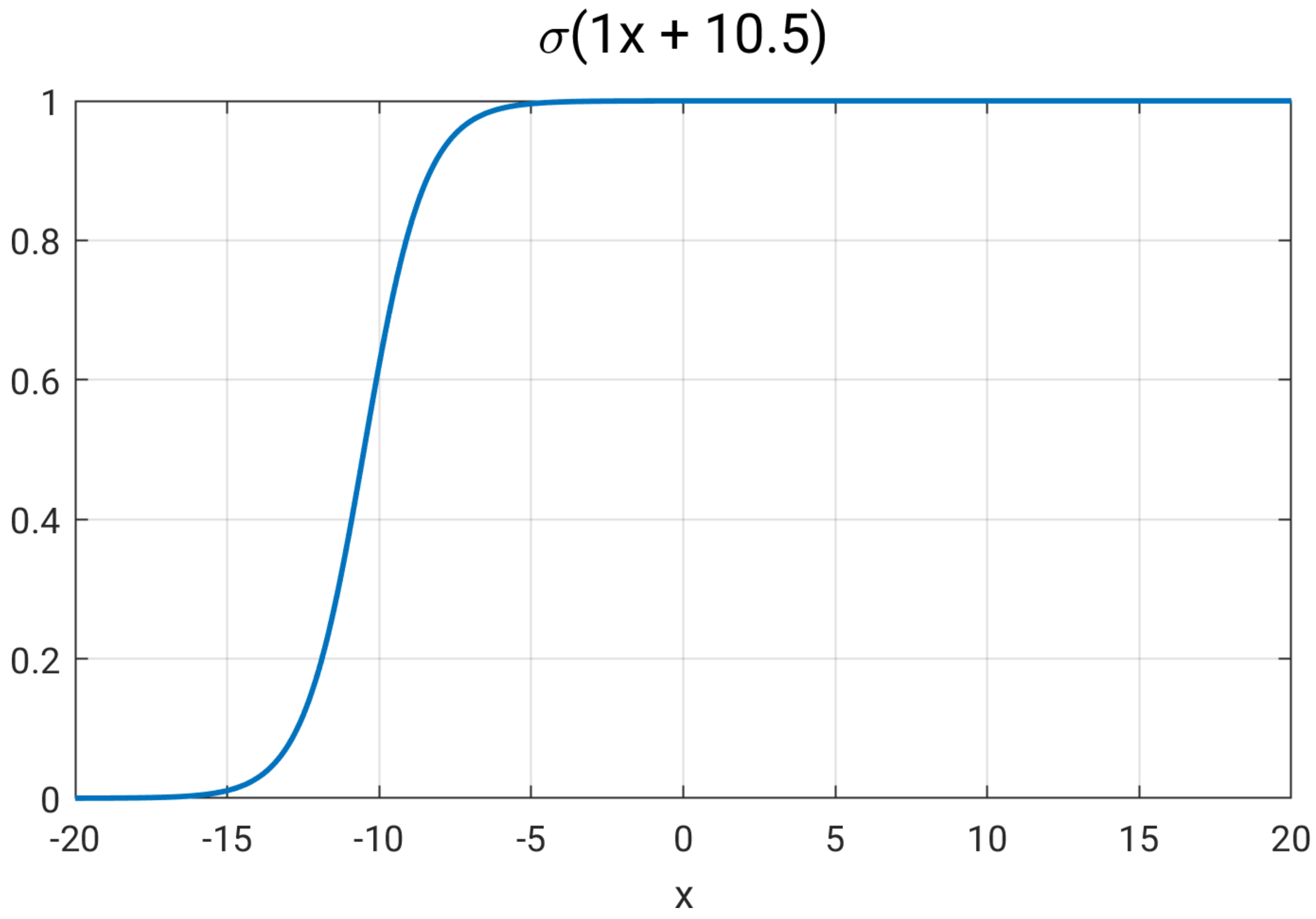
Adaptive basis functions (D=1)



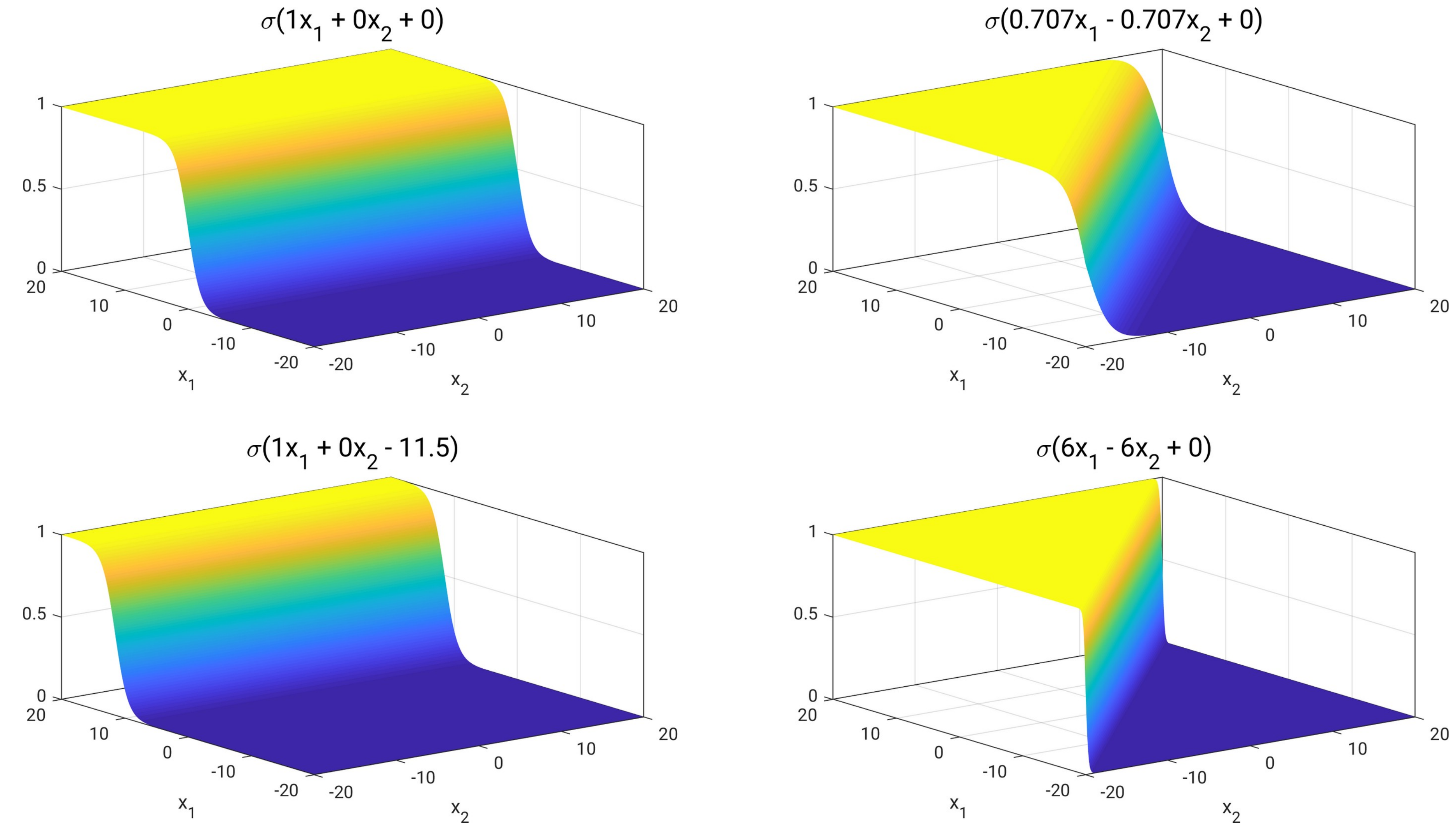
Adaptive basis functions (D=1)



Adaptive basis functions (D=1)



Adaptive basis functions (D=2)



Feed-forward neural network

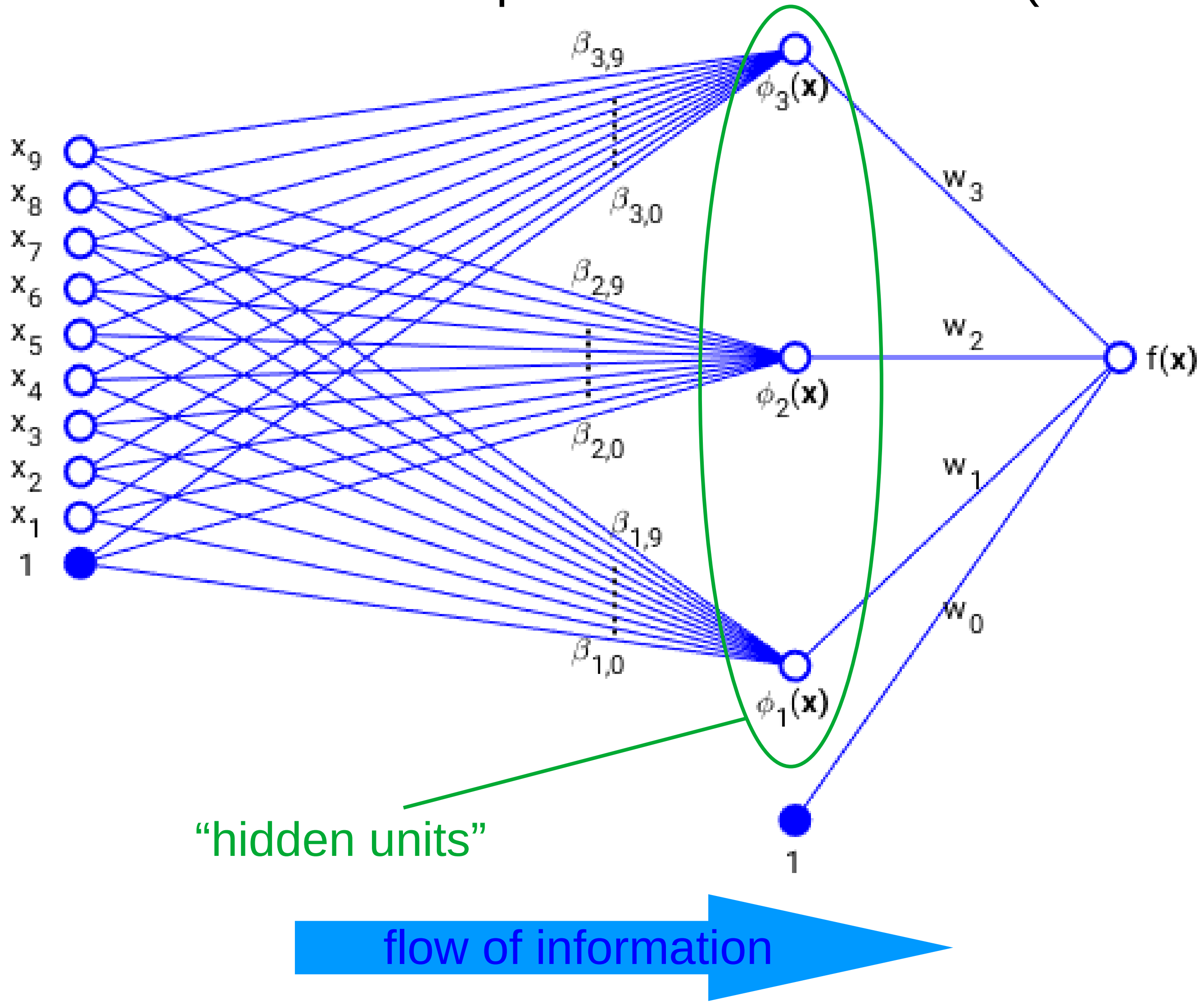
So the model is: $$p(t=1|\mathbf{x},\boldsymbol{\theta}) = \sigma\left(\sum_{m=0}^{M-1} w_m \phi_m(\mathbf{x})\right)$$

parameters

with basis functions $$\phi_m(\mathbf{x}) = \begin{cases} 1 & \text{if } m=0, \\ \sigma\left(\sum_{d=1}^{D} \beta_{m,d} x_d + \beta_{m,0}\right) & \text{otherwise} \end{cases}$$

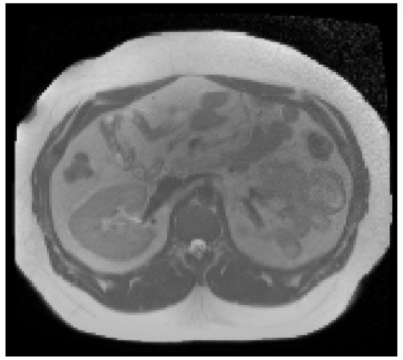
Feed-forward neural network

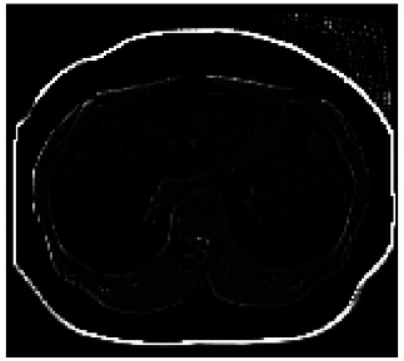
Graphical representation of our 3x3 patch-based classifier (D=9 and M=4):



Can insert more than one “hidden” layer (“deep learning”)

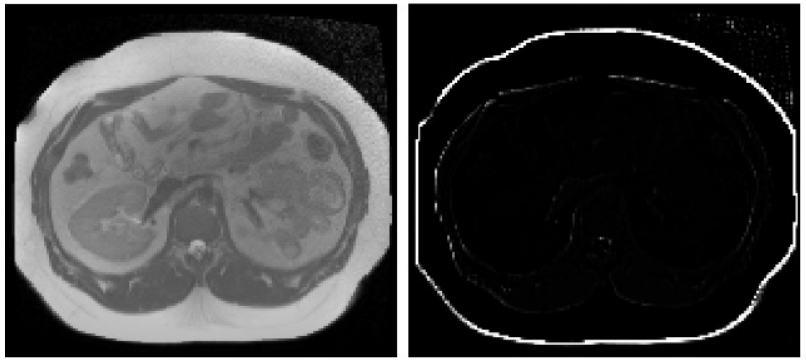
Applying the trained classifier on new data:





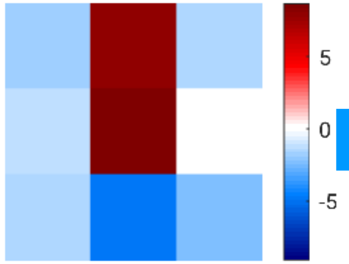
$p(y = 1|\mathbf{x}, \hat{\boldsymbol{\theta}})$

Applying the trained classifier on new data:



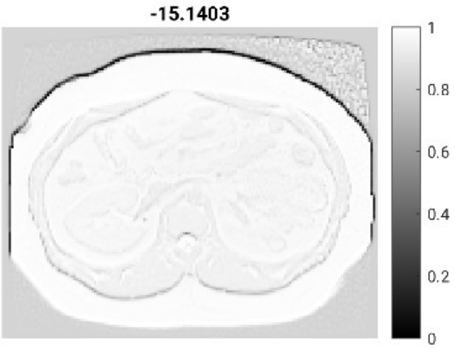
$p(y = 1 | \mathbf{x}, \hat{\boldsymbol{\theta}})$

$\{\beta_{1,1}, \ldots, \beta_{1,9}\}$

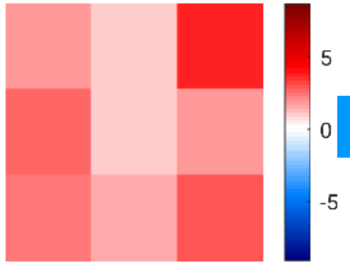


filter

$\phi_1(\mathbf{x})$

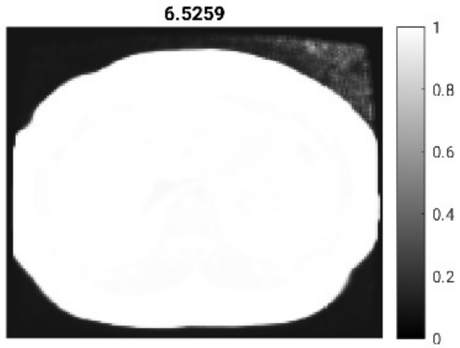


$\{\beta_{2,1}, \ldots, \beta_{2,9}\}$

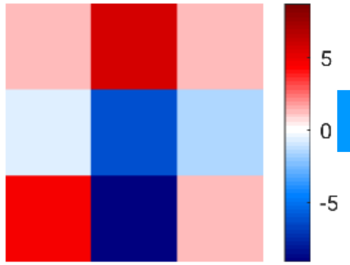


filter

$\phi_2(\mathbf{x})$

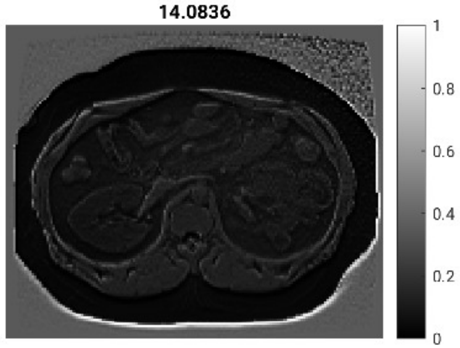


$\{\beta_{3,1}, \ldots, \beta_{3,9}\}$

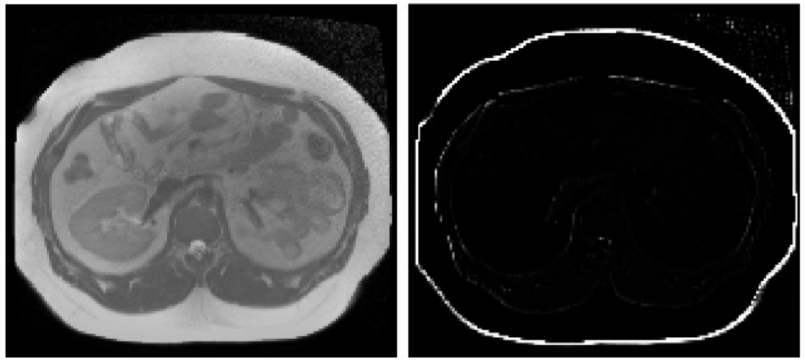


filter

$\phi_3(\mathbf{x})$



Applying the trained classifier on new data:

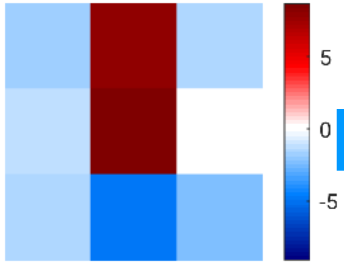


$p(y = 1|\mathbf{x}, \hat{\boldsymbol{\theta}})$

Filtering operations can be implemented using convolutions

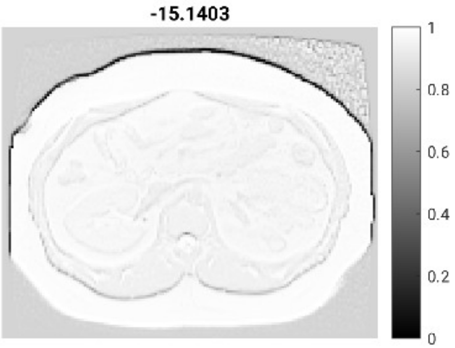
=> “convolutional neural network”

$\{\beta_{1,1}, \ldots, \beta_{1,9}\}$

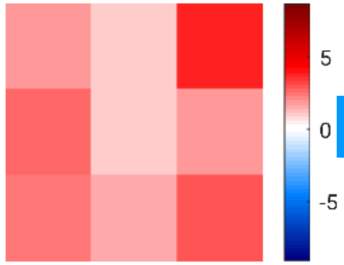


filter

$\phi_1(\mathbf{x})$

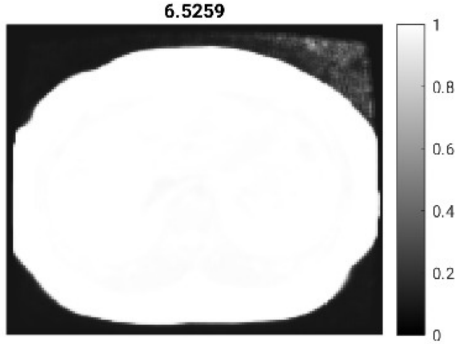


$\{\beta_{2,1}, \ldots, \beta_{2,9}\}$

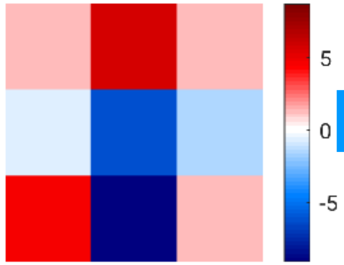


filter

$\phi_2(\mathbf{x})$

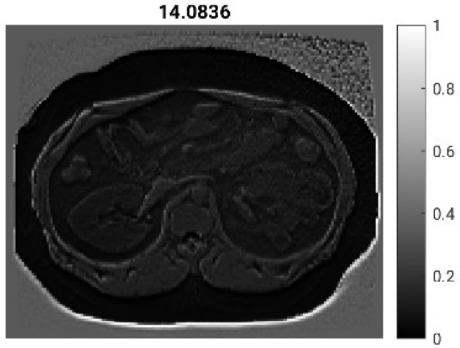


$\{\beta_{3,1}, \ldots, \beta_{3,9}\}$



filter

$\phi_3(\mathbf{x})$



Neural networks = ultimate solution?

No model, only training data:

- No domain expertise needed
- Very easy to train and deploy
- Super fast (GPUs)

- Training data often very hard to get in medical imaging!
- Scanning hardware/software/protocol changes routinely!