Machine Learning

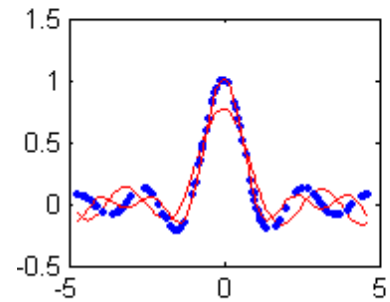
4771

Instructor: Tony Jebara

Topic 3

- Additive Models and Linear Regression
- Sinusoids and Radial Basis Functions
- Neural Networks and Nonlinear Regression
- Linear Neuron
- Logistic Neuron
- Gradient Descent

Sinusoidal Basis Functions



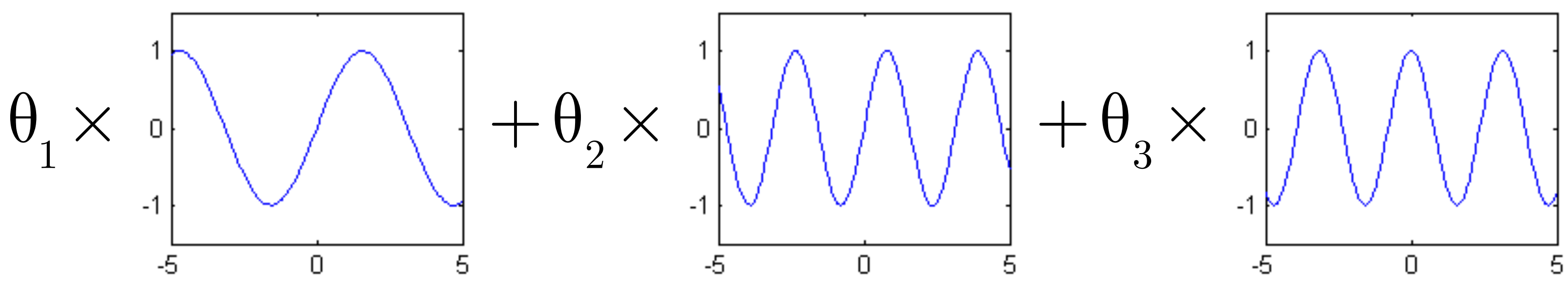
- More generally, we don't just have to deal with polynomials, use any set of basis fn's:

$$f\left(x;\theta\right) = \sum\nolimits_{p=1}^{P} \theta_p \phi_p\left(x\right) + \theta_0$$

- These are generally called Additive Models
- Regression adds linear combinations of the basis fn's
- For example: Fourier (sinusoidal) basis

$$\phi_{2k}\left(x_i\right) = \sin\left(kx_i\right) \qquad \phi_{2k+1}\left(x_i\right) = \cos\left(kx_i\right)$$

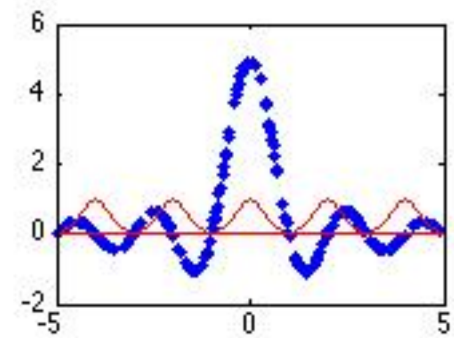
- Note, don't have to be a basis per se, usually subset



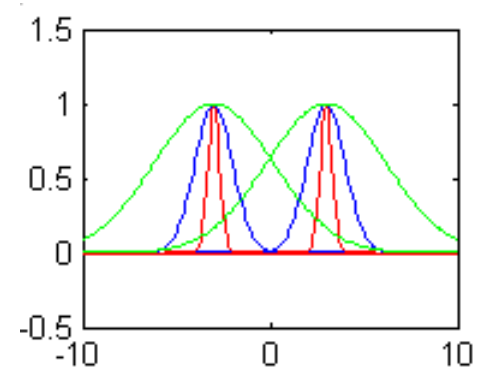
Radial Basis Functions

- Can act as prototypes of the data itself

$$f\left(\mathbf{x};\theta\right) = \sum_{k=1}^{N} \theta_k \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{x}_k\right\|^2\right) + \theta_0$$

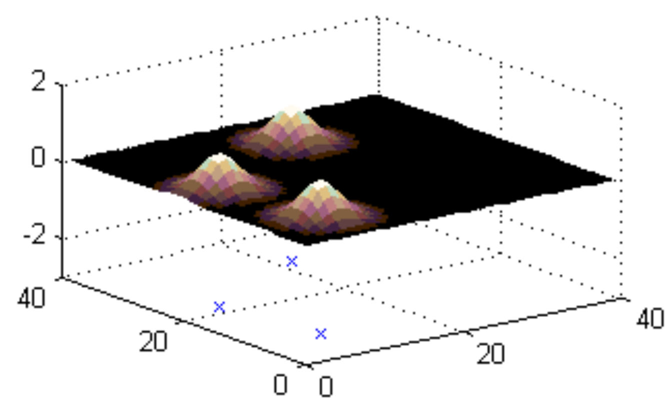


- Parameter σ = standard deviation
  σ^2 = covariance



controls how wide bumps are
what happens if too big/small?

- Also works in multi-dimensions



Radial Basis Functions

- Each training point leads to a bump function

$$f\left(\mathbf{x};\theta\right) = \sum_{k=1}^{N} \theta_k \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{x}_k\right\|^2\right) + \theta_0$$

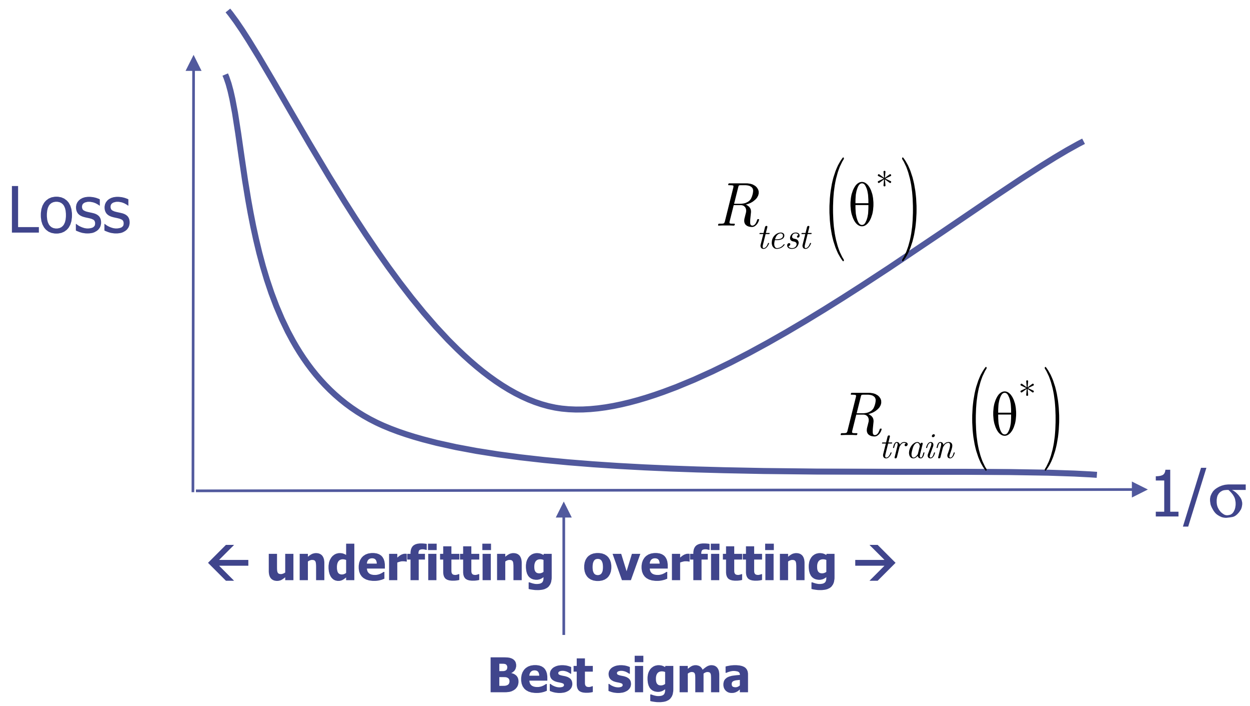
- Reuse solution from linear regression: $\theta^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$
- Can view the data instead as Q, a big matrix of size N x N

$$\mathbf{Q} = \begin{bmatrix} \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_1-\mathbf{x}_1\right\|^2\right) & \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_1-\mathbf{x}_2\right\|^2\right) & \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_1-\mathbf{x}_3\right\|^2\right) \\ \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_2-\mathbf{x}_1\right\|^2\right) & \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_2-\mathbf{x}_2\right\|^2\right) & \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_2-\mathbf{x}_3\right\|^2\right) \\ \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_3-\mathbf{x}_1\right\|^2\right) & \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_3-\mathbf{x}_2\right\|^2\right) & \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{x}_3-\mathbf{x}_3\right\|^2\right) \end{bmatrix}$$

- In this setting, X is invertible ,solution is just $\theta^* = \mathbf{Q}^{-1}\mathbf{y}$

Crossvalidation

- Try fitting with different sigma radial basis function widths
- Select sigma which gives lowest $R_{test}(\theta^*)$



- Think of sigma as a measure of the simplicity of the model
- Thinner RBFs are more flexible and complex

Regularized Risk Minimization

- Empirical Risk Minimization gave overfitting & underfitting
- We want to add a penalty for using too many theta values
- This gives us the Regularized Risk

$$R_{empirical} = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \theta^T x_i\right)$$

$$R_{regularized} = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \theta^T x_i\right) + \frac{\lambda}{2}\left\|\theta\right\|^2$$

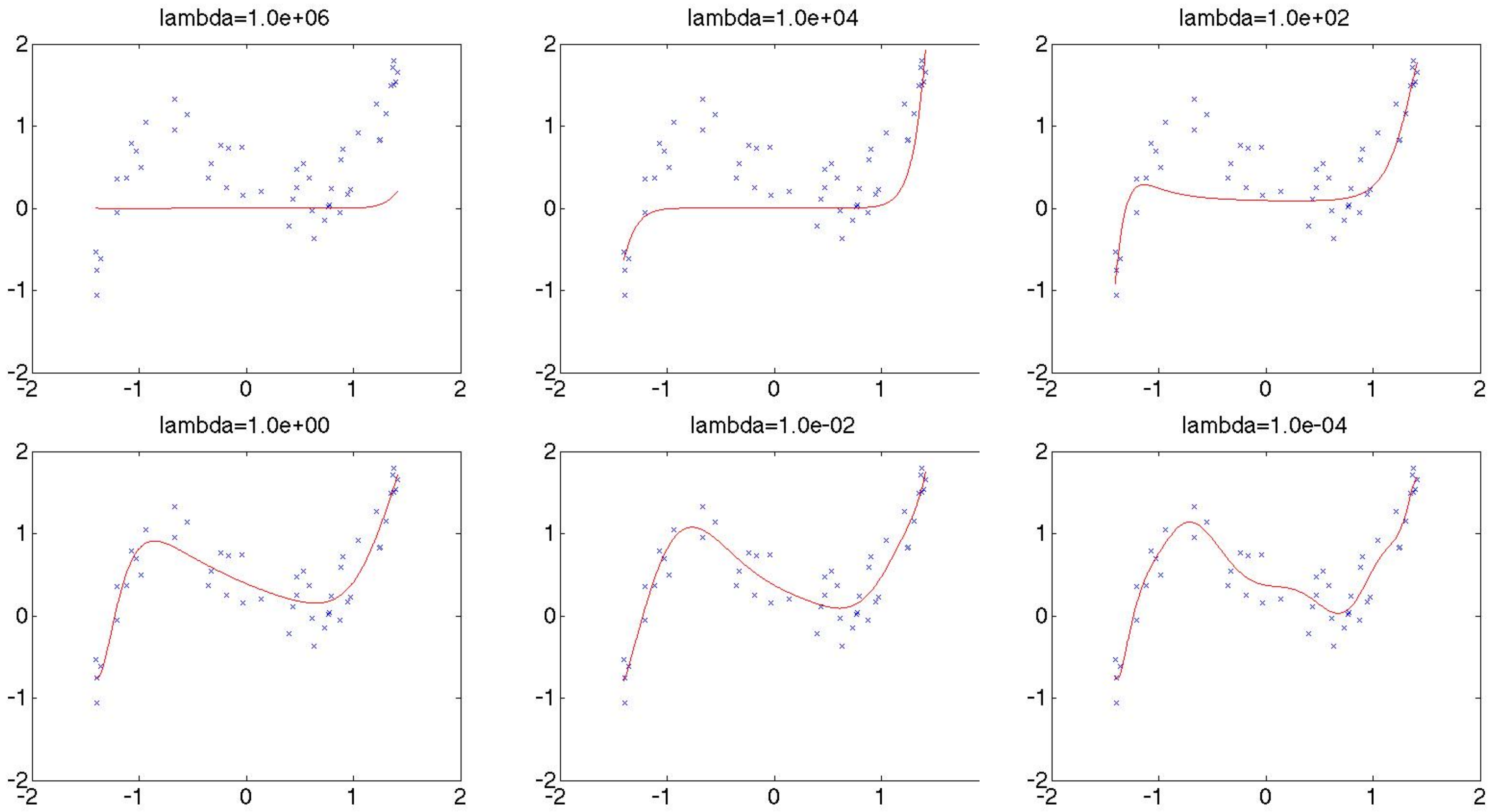
- Solution for Regularized Risk with Least Squares Loss:

$$\nabla_\theta R_{regularized} = 0 \quad \Rightarrow \quad \nabla_\theta \left(\frac{1}{2N}\left\|\mathbf{y} - \mathbf{X}\theta\right\|^2 + \frac{\lambda}{2}\left\|\theta\right\|^2\right) = 0$$

$$\theta^* = \left(\mathbf{X}^T\mathbf{X} + \lambda I\right)^{-1}\mathbf{X}^T\mathbf{y}$$

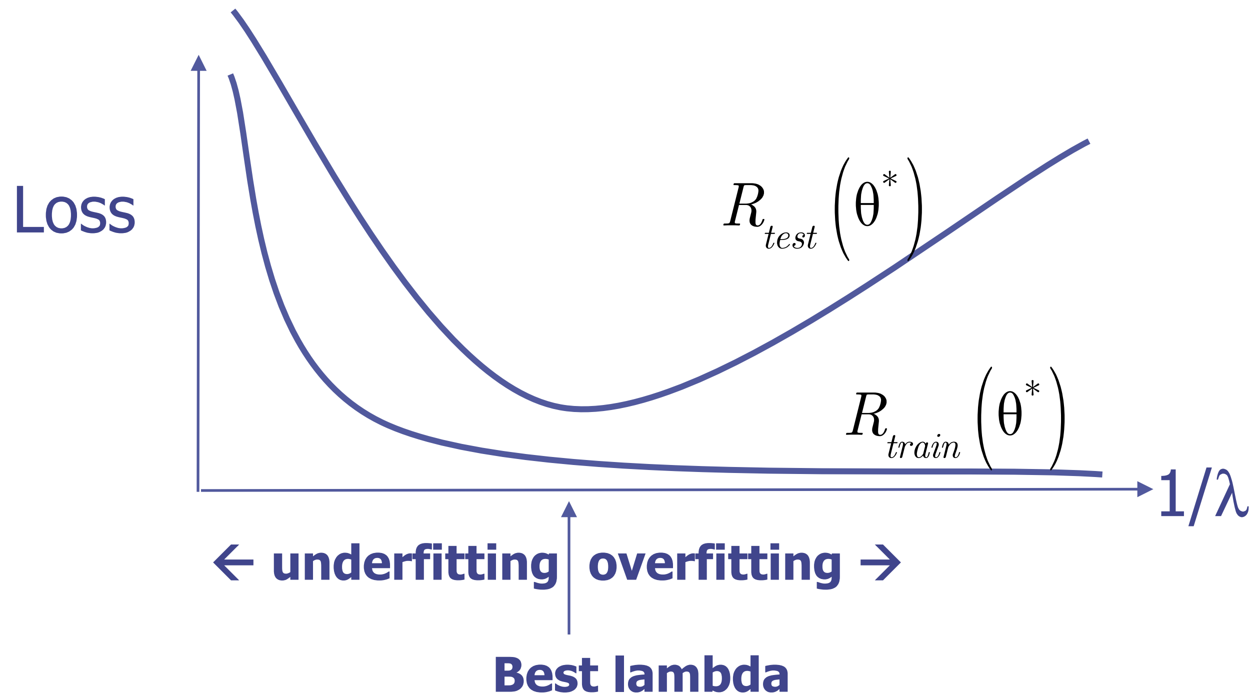
Regularized Risk Minimization

- Set P to 15 throughout. Try varying λ instead.
- Minimize $R_{regularized}(\theta)$ to get θ^*, observe $R_{empirical}(\theta^*)$



Crossvalidation

- Try fitting with different lambda regularization levels
- Select lambda which gives lowest $R_{test}(\theta^*)$

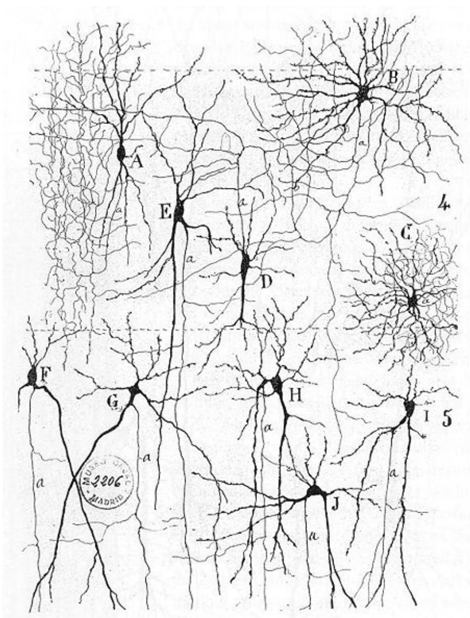


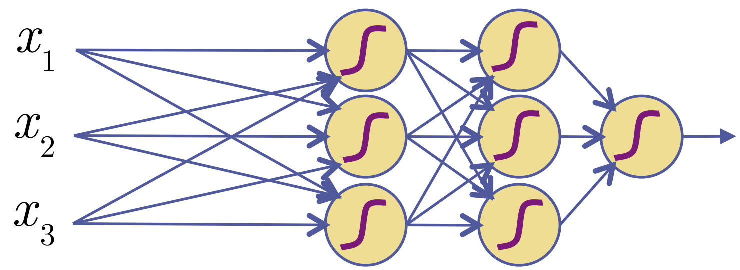
- Think of lambda as a measure of the simplicity of the model
- Models with low lambda are more flexible and complex

Beyond Linear (in θ) Regression

- Simple linear regression case $f\left(\mathbf{x};\theta\right) = \sum_{p=1}^{P} \theta_p x_p + \theta_0$

- What is a more complicated function f(x) we could try?
- Inspired by the brain, a neural network
- Can be seen as a function from inputs to outputs

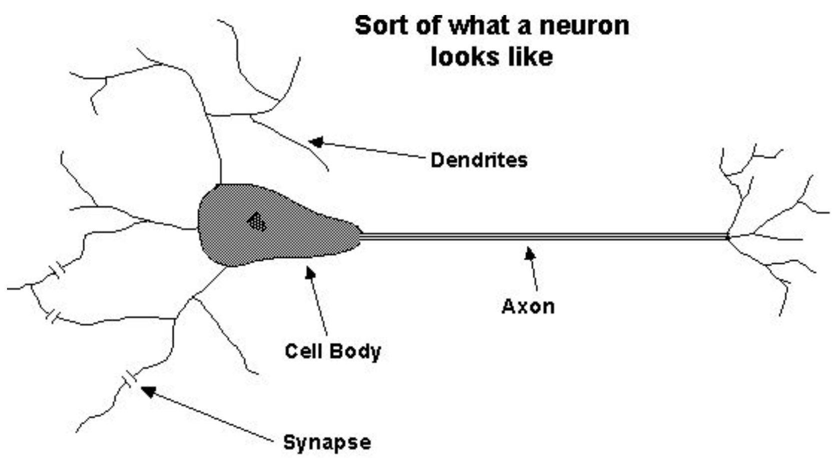


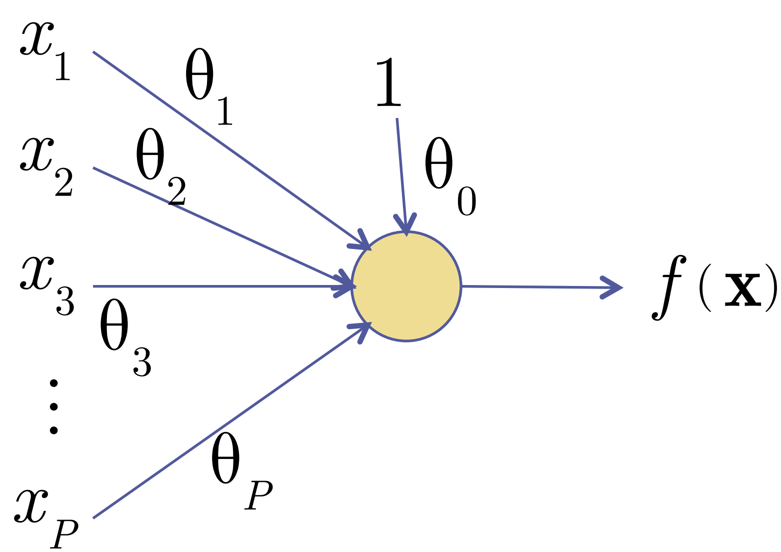


- Smallest piece is a Neuron, a node in the network…

The Neuron as Regression

- The McCullough-Pitts Neuron is a graphical representation of linear regression $f\left(\mathbf{x};\theta\right) = \sum_{p=1}^{P} \theta_p x_p + \theta_0$
- Edges multiply signal by scalar weight
- Nodes just sum inputs here
- Parameters: $\theta_1 \ldots \theta_P$ = weights θ_0 = bias

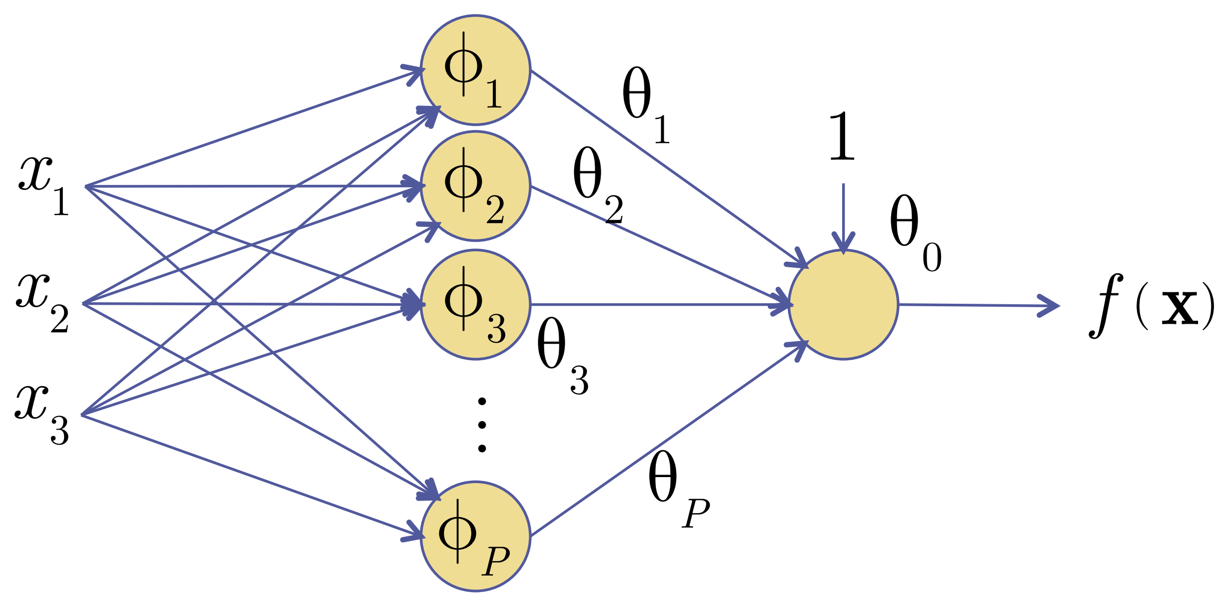




- If neuron is linear function → like usual linear regression

Neuron for Basis Regression

- Graphical representation of $f\left(\mathbf{x};\theta\right) = \sum_{p=1}^{P} \theta_p \phi_p\left(\mathbf{x}\right) + \theta_0$
- Edge multiply signal by scalar weight
- Nodes sum inputs or apply function to inputs
- Parameters: $\theta_1 \ldots \theta_P$ = weights θ_0 = bias

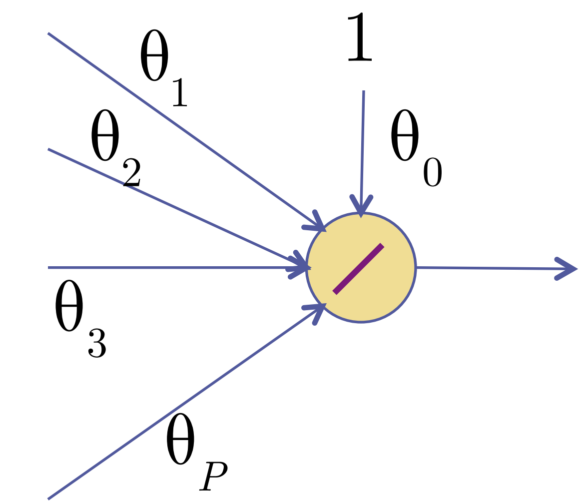


Logistic Neuron Output

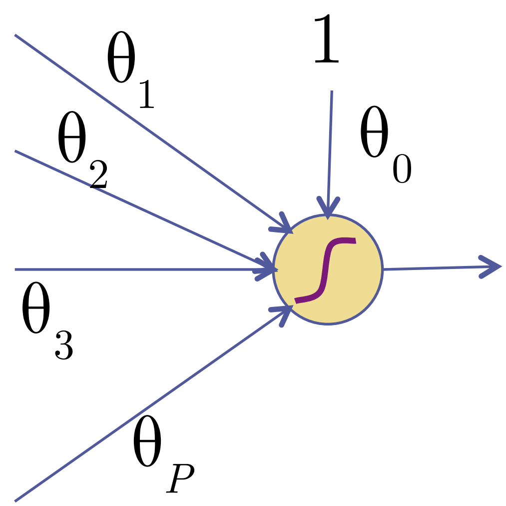
- Another choice of last node is squashing function g().

$$f(\mathbf{x};\theta) = \theta^T \mathbf{x}$$

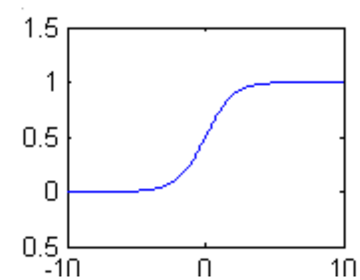
$$f(\mathbf{x};\theta) = g(\theta^T \mathbf{x})$$
$$g(z) = (1 + \exp(-z))^{-1}$$



Linear neuron



Logistic Neuron



- This squashing is called sigmoid or logistic function