SYDE 556/750

Simulating Neurobiological Systems
Lecture 8: Learning

Andreas Stöckel

February 25 & 27, 2020

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Supervised Learning

$\mathbf{x}_1 =$  $\mathbf{t}_1 = (1, 0, 0)$

$\mathbf{x}_2 =$  $\mathbf{t}_2 = (1, 0, 0)$

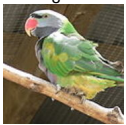
$\mathbf{x}_3 =$  $\mathbf{t}_3 = (1, 0, 0)$

$\mathbf{x}_4 =$  $\mathbf{t}_4 = (0, 0, 1)$

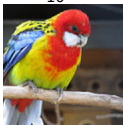
$\mathbf{x}_5 =$  $\mathbf{t}_5 = (1, 0, 0)$

$\mathbf{x}_6 =$  $\mathbf{t}_6 = (0, 0, 1)$

$\mathbf{x}_7 =$  $\mathbf{t}_7 = (1, 0, 0)$

$\mathbf{x}_8 =$  $\mathbf{t}_8 = (0, 0, 1)$

$\mathbf{x}_9 =$  $\mathbf{t}_9 = (0, 1, 0)$

$\mathbf{x}_{10} =$  $\mathbf{t}_{10} = (0, 0, 1)$

$\mathbf{x}_{11} =$  $\mathbf{t}_{11} = (1, 0, 0)$

$\mathbf{x}_{12} =$  $\mathbf{t}_{12} = (1, 0, 0)$

$\mathbf{x}_{13} =$  $\mathbf{t}_{13} = (0, 1, 0)$

$\mathbf{x}_{14} =$  $\mathbf{t}_{14} = (0, 1, 0)$

$\mathbf{x}_{15} =$  $\mathbf{t}_{15} = (0, 1, 0)$

$\mathbf{x}_{16} =$  $\mathbf{t}_{16} = (0, 1, 0)$

$\mathbf{x}_{17} =$  $\mathbf{t}_{17} = (0, 0, 1)$

$\mathbf{x}_{18} =$  $\mathbf{t}_{18} = (0, 1, 0)$

$\mathbf{x}_{20} =$  $\mathbf{t}_{20} = (0, 1, 0)$

$\mathbf{x}_{21} =$  $\mathbf{t}_{21} = (0, 0, 1)$

$\mathbf{x}_{22} =$  $\mathbf{t}_{22} = (0, 0, 1)$

$\mathbf{x}_{23} =$  $\mathbf{t}_{23} = (1, 0, 0)$

$\mathbf{x}_{24} =$  $\mathbf{t}_{24} = (0, 1, 0)$

Supervised Learning – Training

$\mathbf{x}_{10} =$

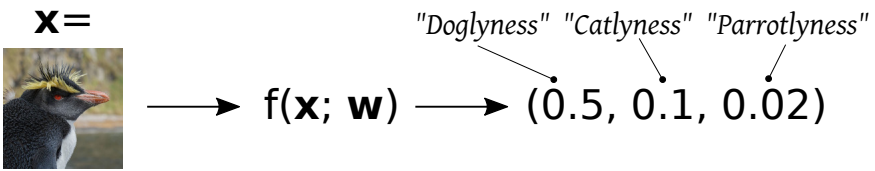
$\rightarrow$ $f(\mathbf{x}; \mathbf{w})$ $\rightarrow$ (0.1, 0.9, 0.25)

"Doglyness" *"Catlyness"* *"Parrotlyness"*

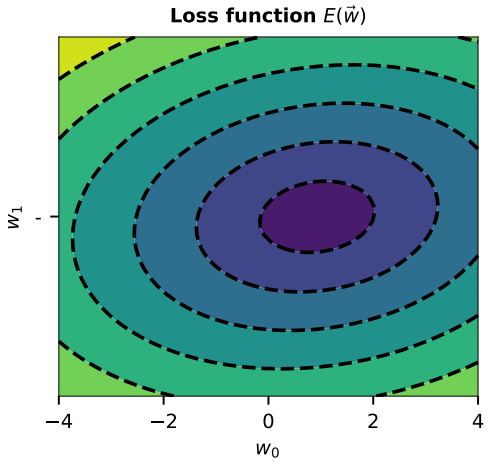
weight adaptation

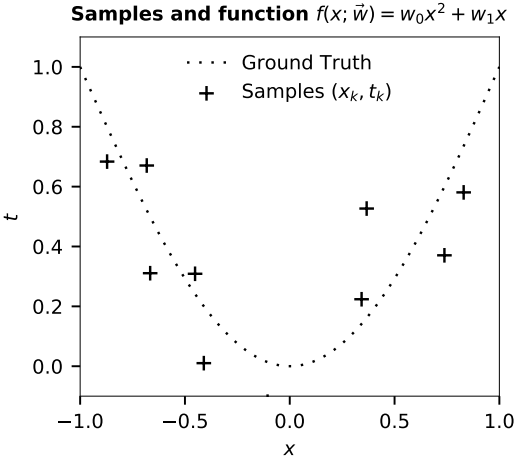
$\mathbf{t}_{10} = (0.0, 0.0, 1.0)$

Supervised Learning – Inference

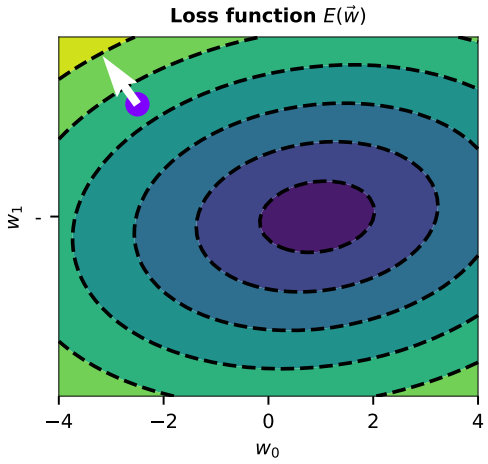


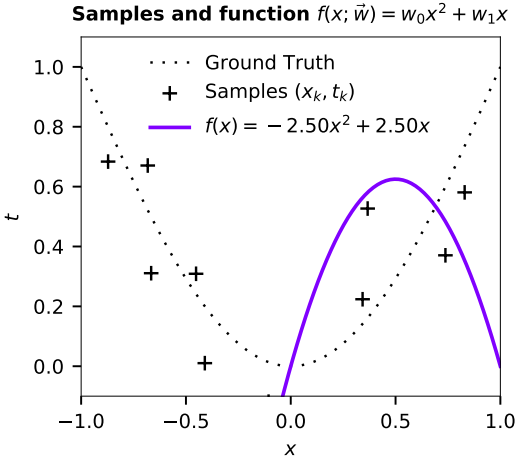
Gradient Descent – Example



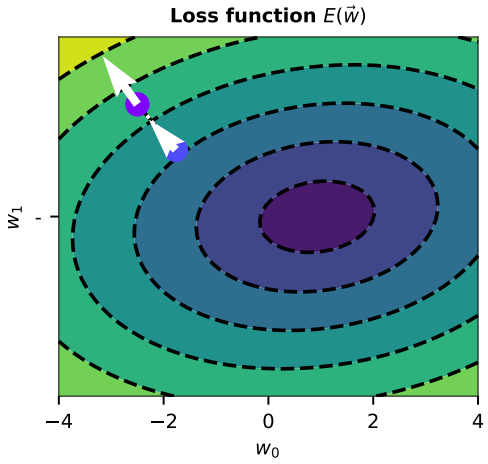


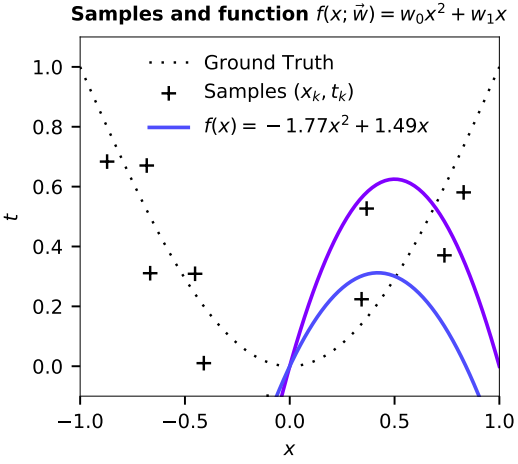
Gradient Descent – Example



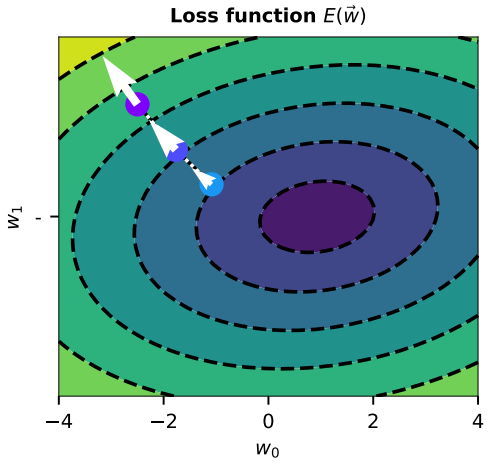


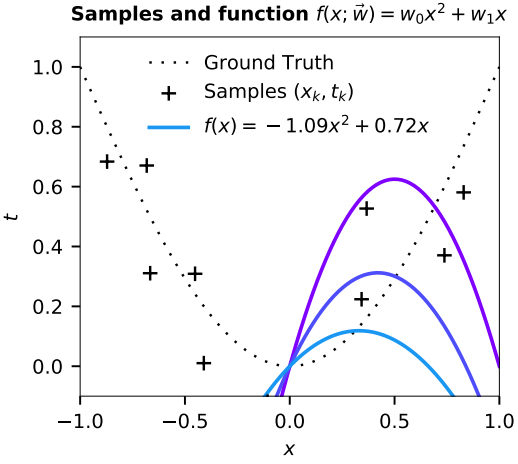
Gradient Descent – Example



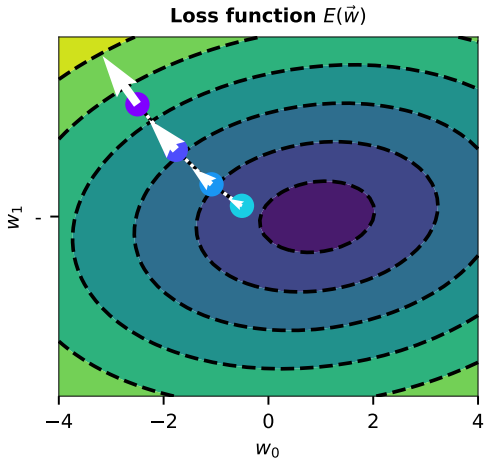


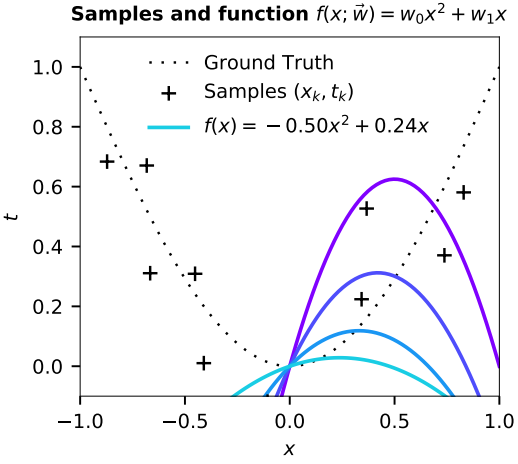
Gradient Descent – Example



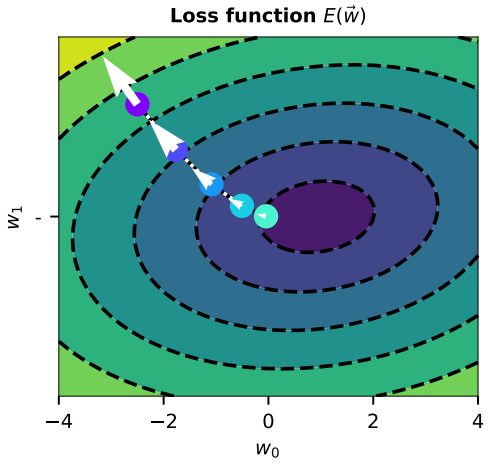


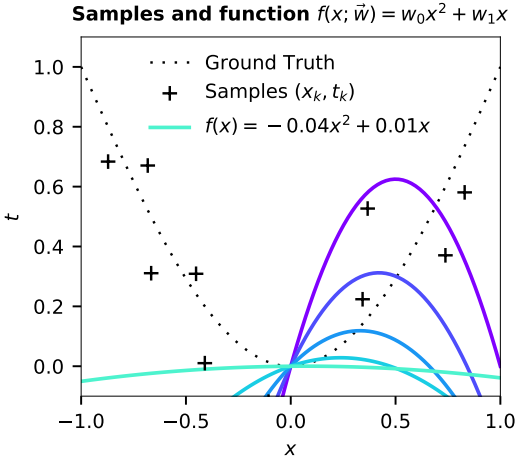
Gradient Descent – Example



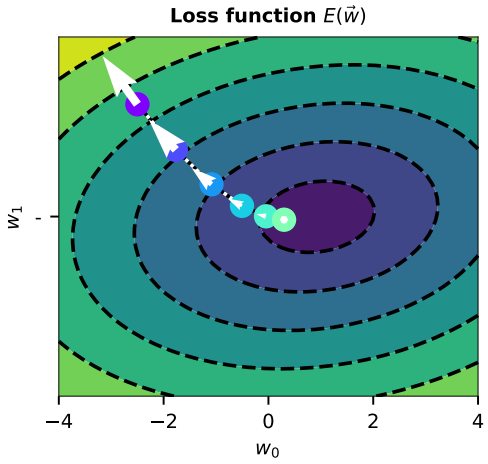


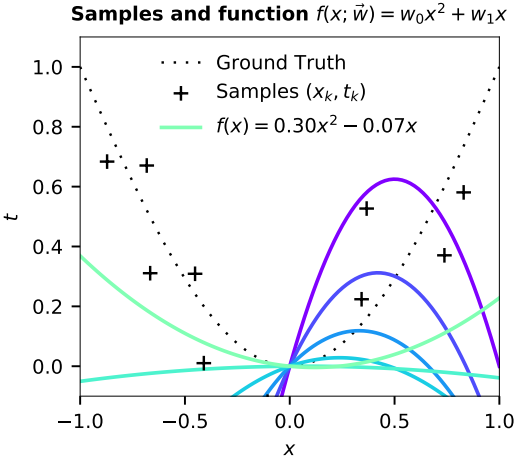
Gradient Descent – Example



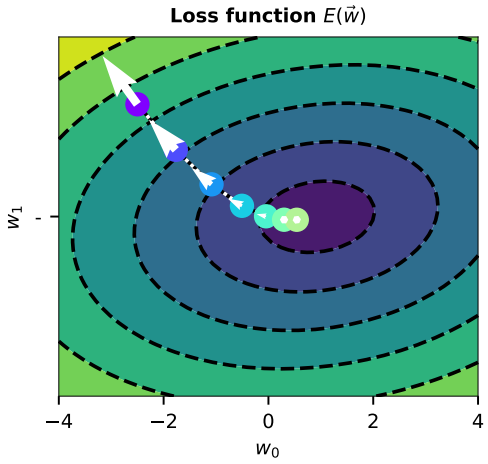


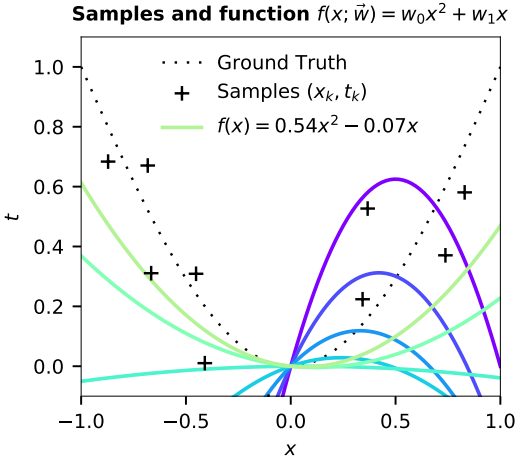
Gradient Descent – Example



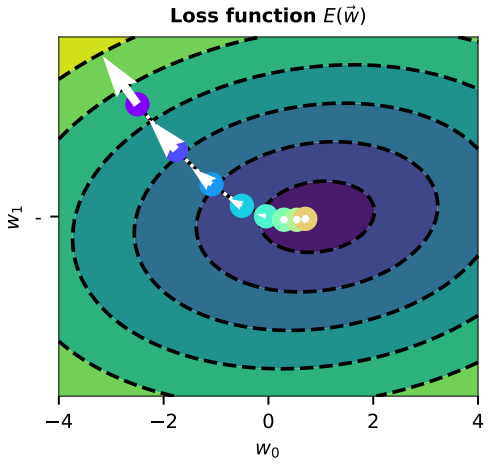


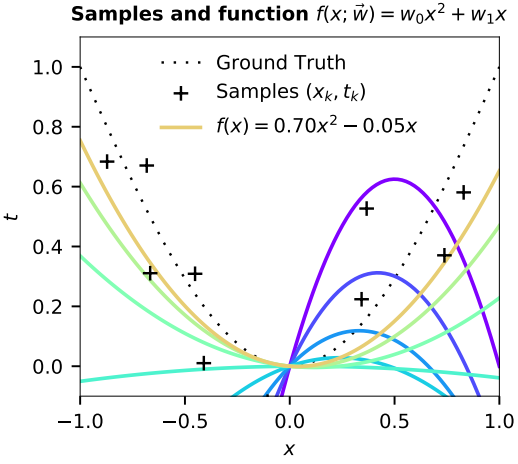
Gradient Descent – Example



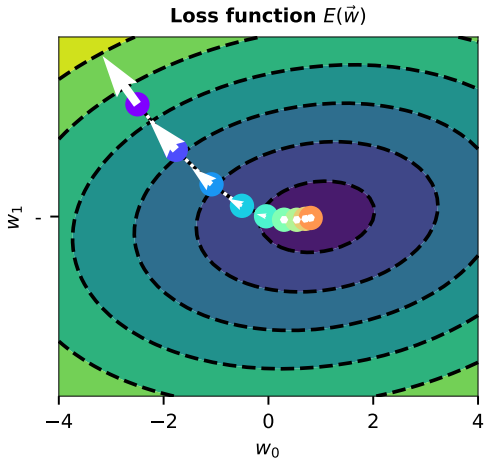


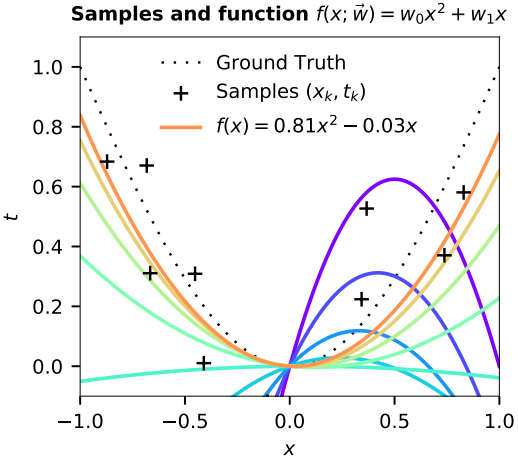
Gradient Descent – Example



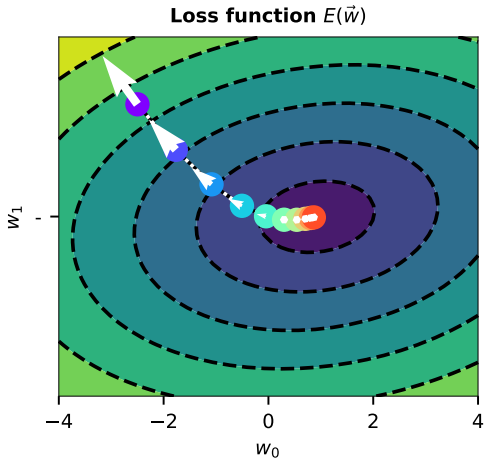


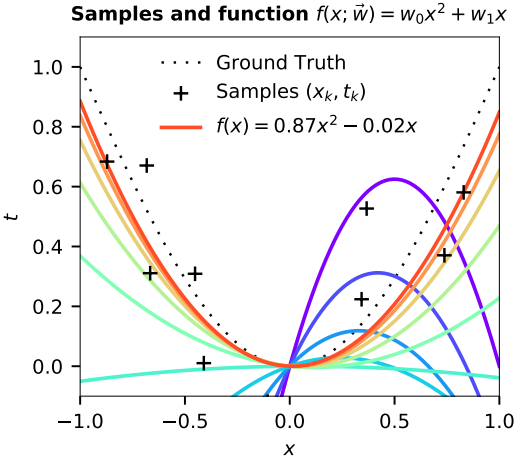
Gradient Descent – Example



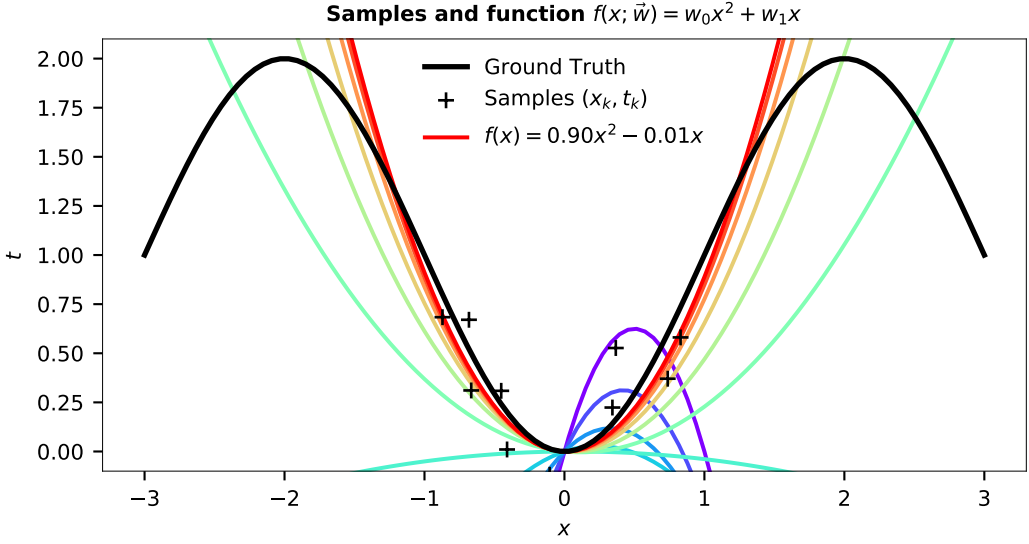


Gradient Descent – Example



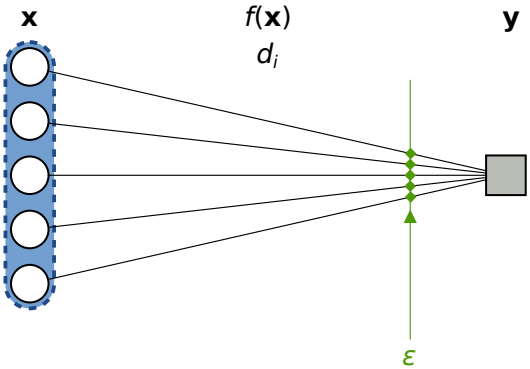


Supervised Learning – Generalisation



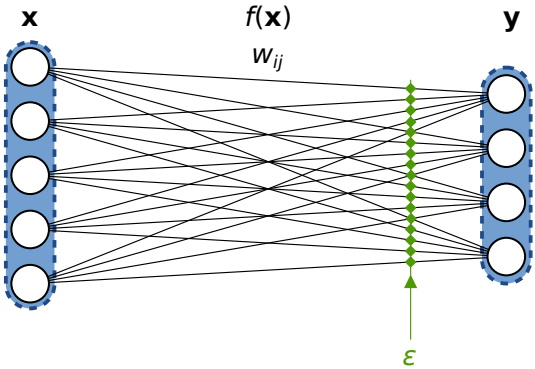
Learning Decoders and Learning Weights

Learning Decoders (Delta Rule)



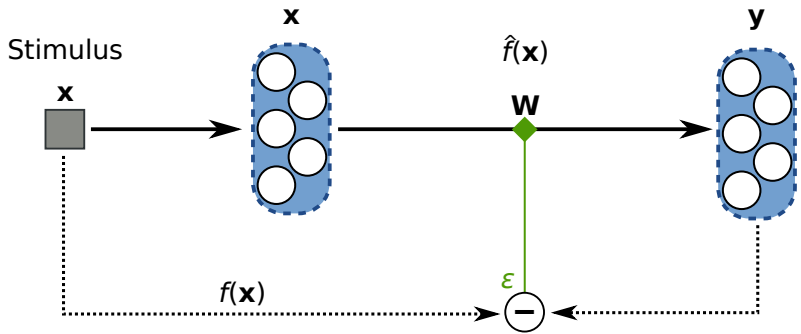
$$\Delta d_i = -\eta a_i(\mathbf{x}) \underbrace{\left(y(t) - y^{\mathrm{d}}(t)\right)}_{\varepsilon(t)}$$

Learning Weights (PES Rule)

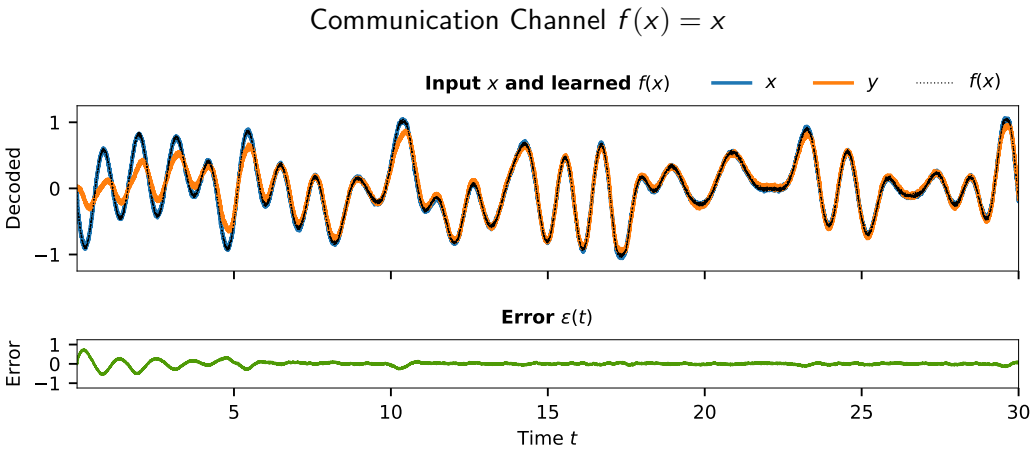


$$\Delta w_{ij} = -\eta a_i(\mathbf{x}) \Big(\alpha_j \langle \mathbf{e}_j, \varepsilon(t) \rangle \Big)$$

Example: Learning Functions (I)

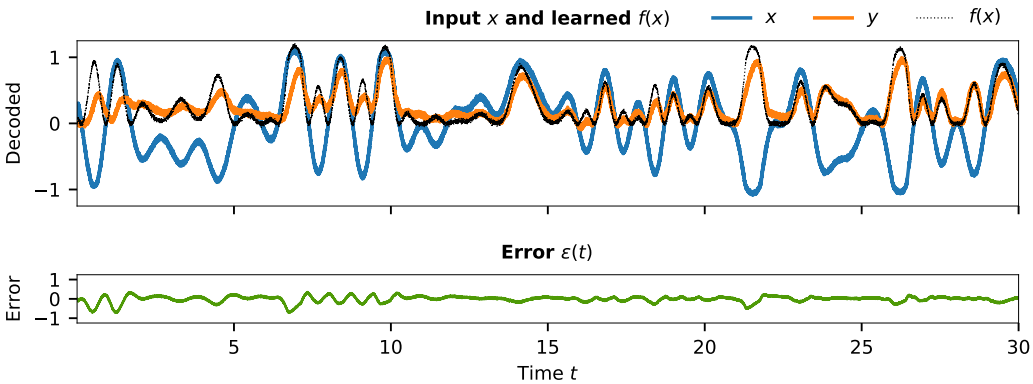


Example: Learning Functions (II)



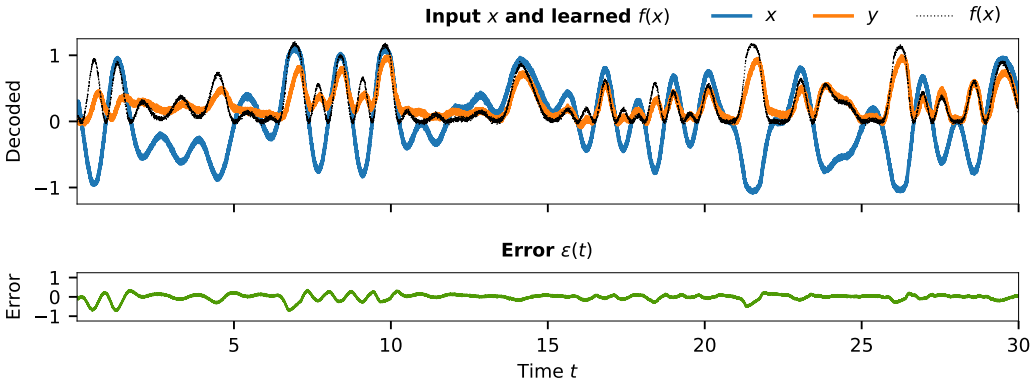
Example: Learning Functions (III)

Square $f(x) = x^2$



Example: Learning Functions (III)

Square $f(x) = x^2$

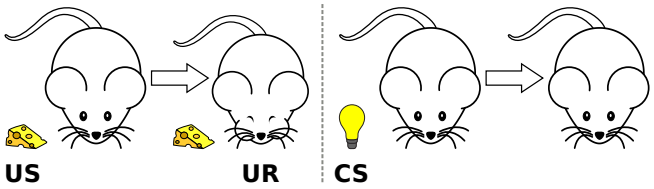


Works, but learns more slowly!

Where is the error signal $\varepsilon(t)$ coming from?

Example: Classical Conditioning (I)

Before conditioning:



After conditioning:

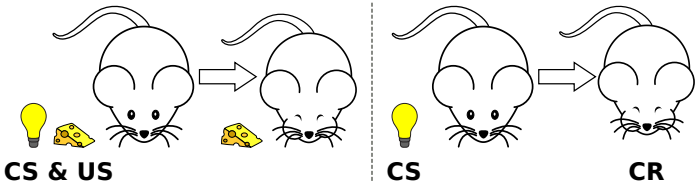
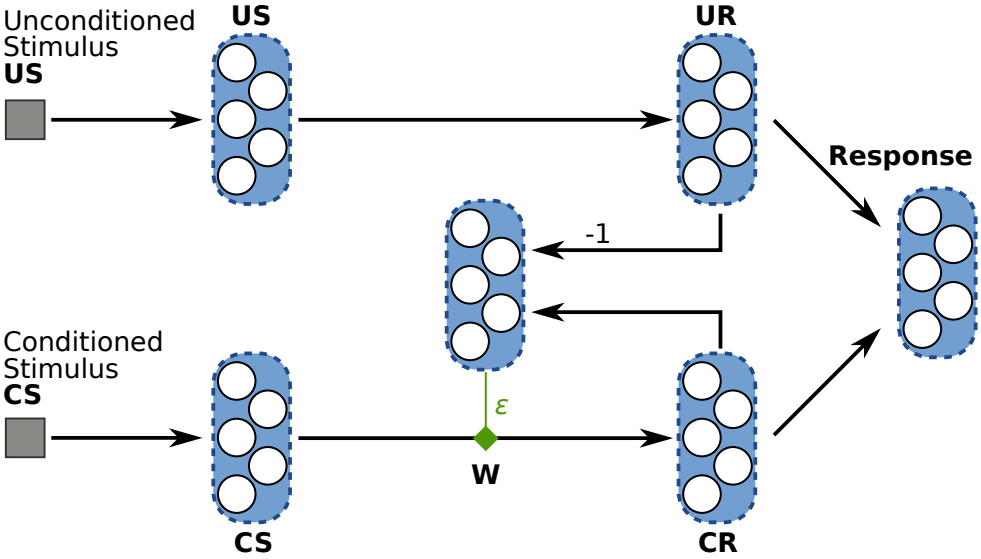
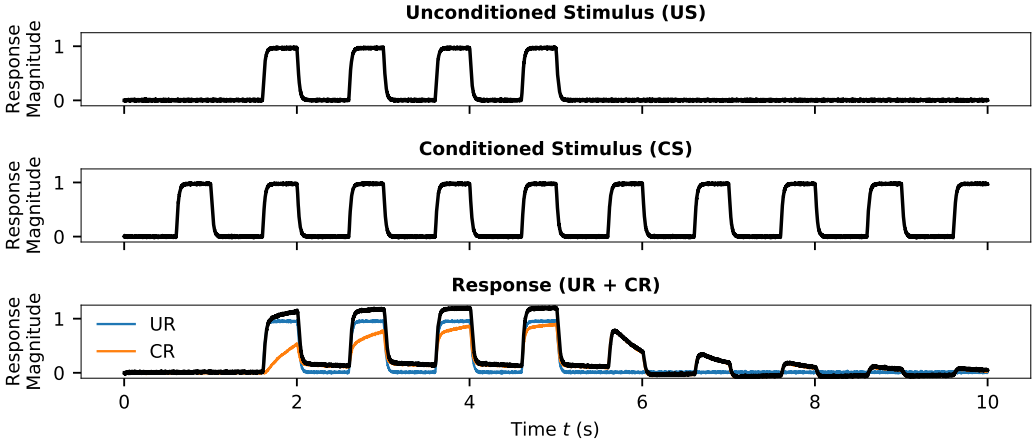


Image Sources. Images adapted from Wikimedia.

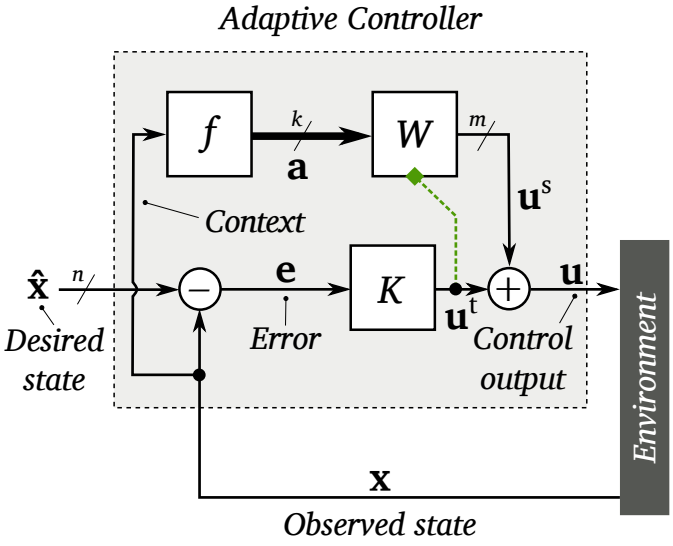
Example: Classical Conditioning (II)



Example: Classical Conditioning (III)



Example: Adaptive Controller



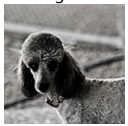
Unsupervised Learning

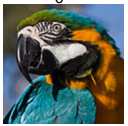
$\mathbf{x}_1 =$ 

$\mathbf{x}_2 =$ 

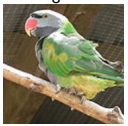
$\mathbf{x}_3 =$ 

$\mathbf{x}_4 =$ 

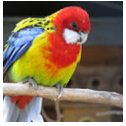
$\mathbf{x}_5 =$ 

$\mathbf{x}_6 =$ 

$\mathbf{x}_7 =$ 

$\mathbf{x}_8 =$ 

$\mathbf{x}_9 =$ 

$\mathbf{x}_{10} =$ 

$\mathbf{x}_{11} =$ 

$\mathbf{x}_{12} =$ 

$\mathbf{x}_{13} =$ 

$\mathbf{x}_{14} =$ 

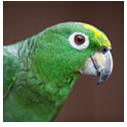
$\mathbf{x}_{15} =$ 

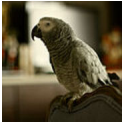
$\mathbf{x}_{16} =$ 

$\mathbf{x}_{17} =$ 

$\mathbf{x}_{18} =$ 

$\mathbf{x}_{20} =$ 

$\mathbf{x}_{21} =$ 

$\mathbf{x}_{22} =$ 

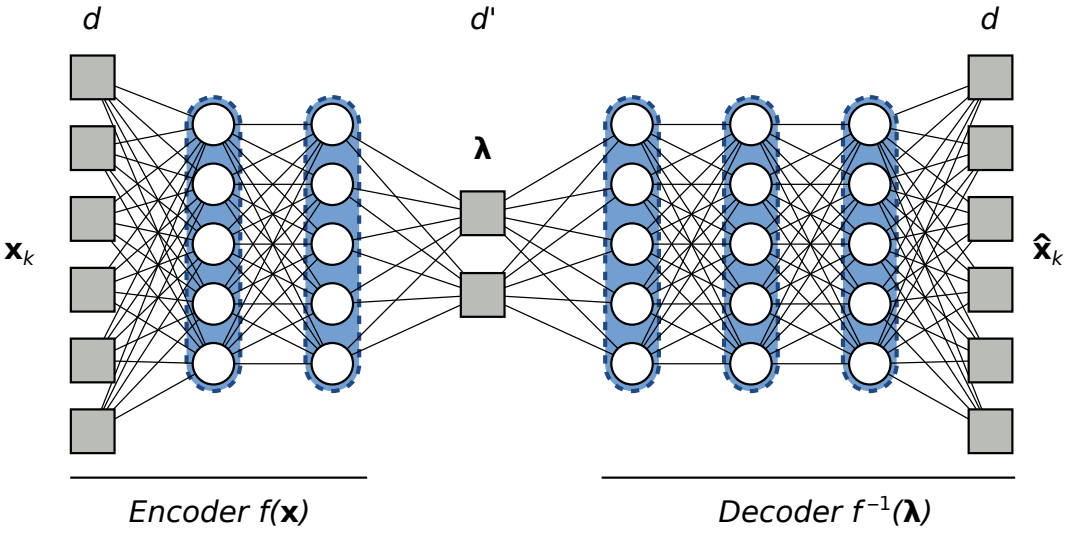
$\mathbf{x}_{23} =$ 

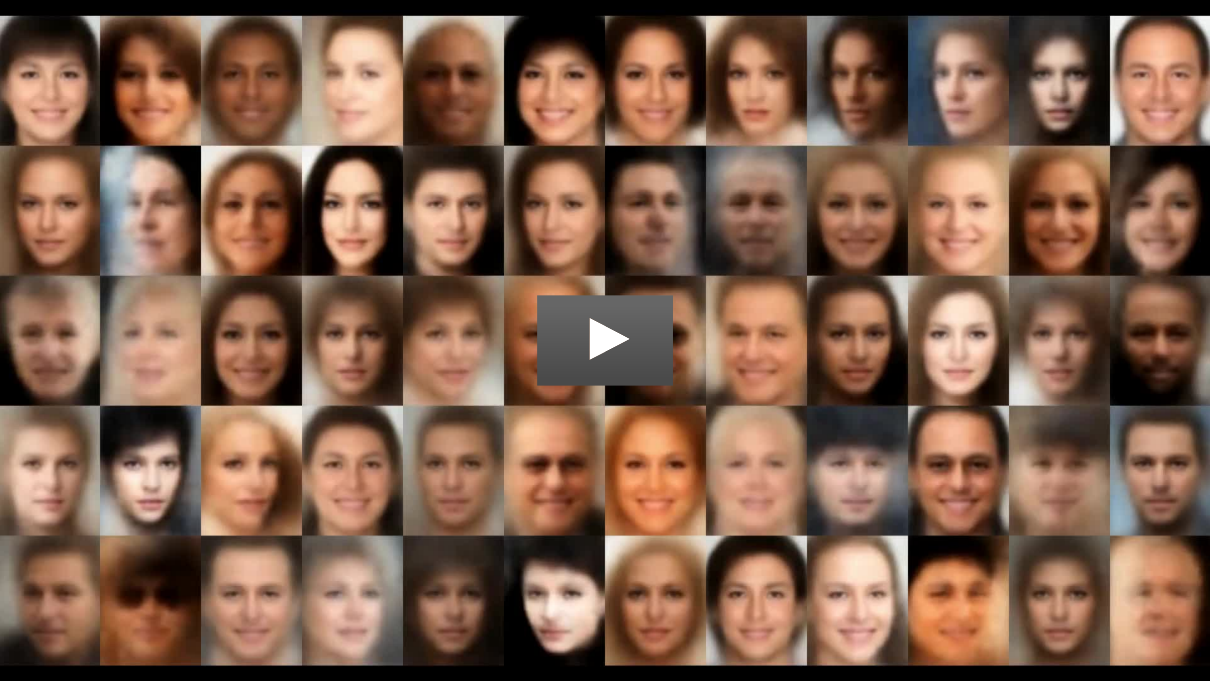
$\mathbf{x}_{24} =$ 

Unsupervised Learning – Training

Unsupervised Learning – Inference

Autoencoder





PCA in Python

```python
def PCA(X): # X: N x d matrix
    N, d = X.shape
    X_cen = X - np.mean(X, axis=0)
    C = (X_cen.T @ X_cen) / (N - 1)
    L, V = np.linalg.eigh(C) # "eigh" faster than "eig" for symmetric matrices
    return V.T[::-1, :] # d x d matrix
```

```python
def PCA_SVD(X): # X: N x d matrix
    return np.linalg.svd(X - np.mean(X, axis=0))[2]
```

PCA Example: Source Images

Face Database

- 84 images of 12 women with 7 different expressions
- Normalised eye location
- 45×60 pixels (2700 dimensions)
- Greyscale

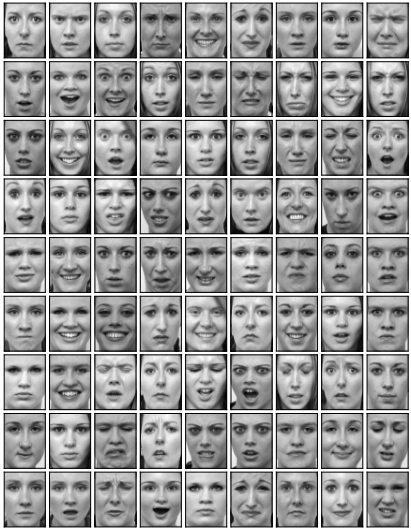
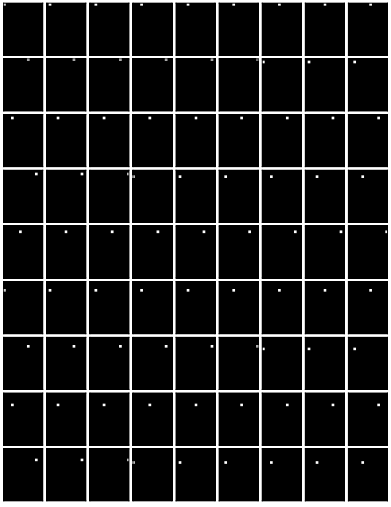


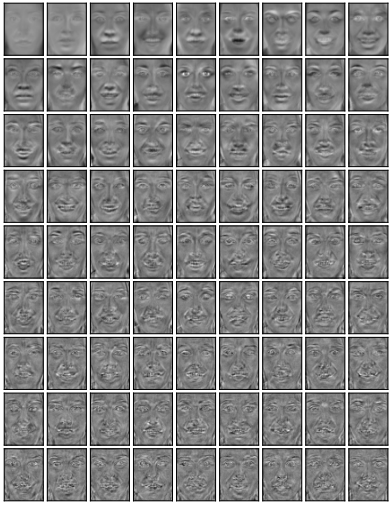
Image Sources. "Pain Expression Subset;" http://pics.stir.ac.uk/2D_face_sets.htm

PCA Example: Eigenfaces

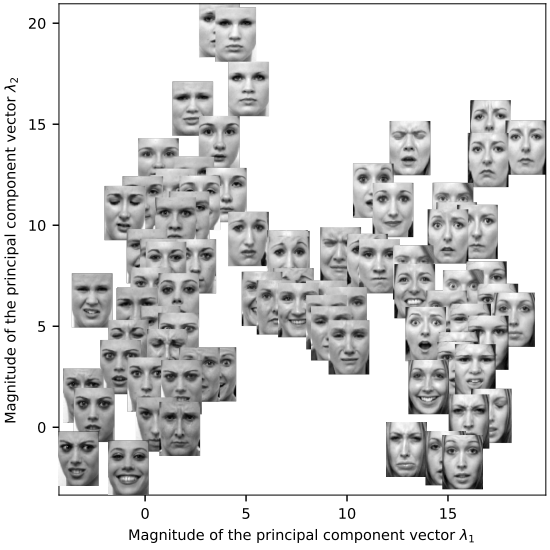
Identity Basis

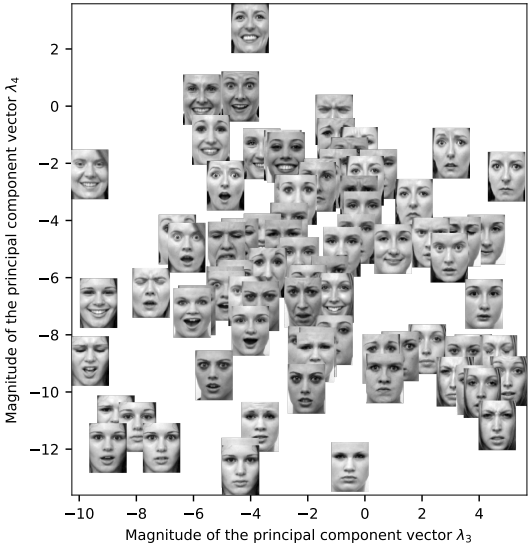


Principal Components



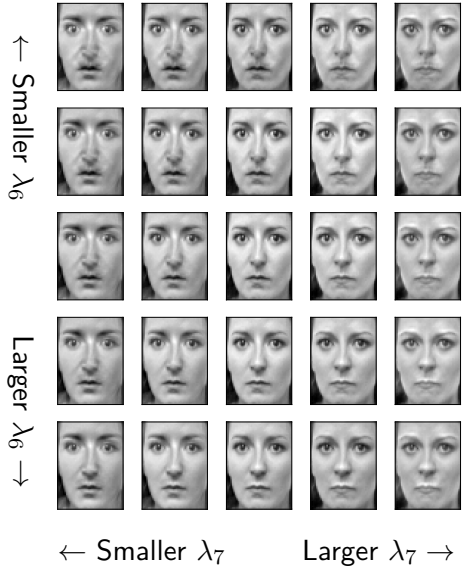
PCA Example: Face Spaces



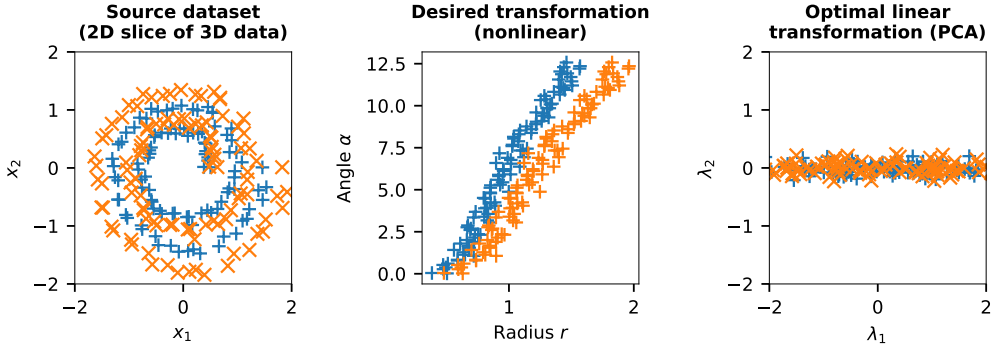


PCA Example: Sparse Vectors

PCA Example: Modifying the Latent Space



Limitations of PCA: Classifying Two Groups

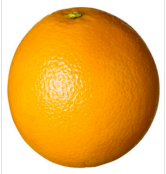


Limitations of PCA: Metaphorical Illustration

Optimally extracted manifold **X'**

Dataset **X**

PCA **X'**



PCA

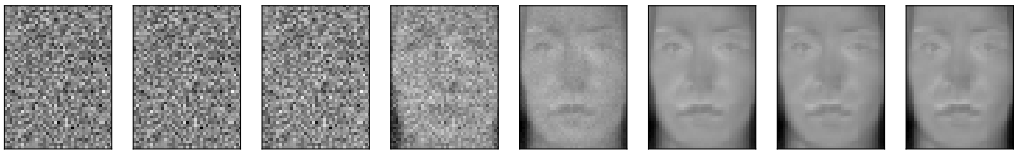
Projects ("squashes") data onto a hyperplane

Image Sources. Barbara Hammer, "Maschinelles Lernen im Web", Universität Bielefeld, WS 2014/15

Hebbian Learning

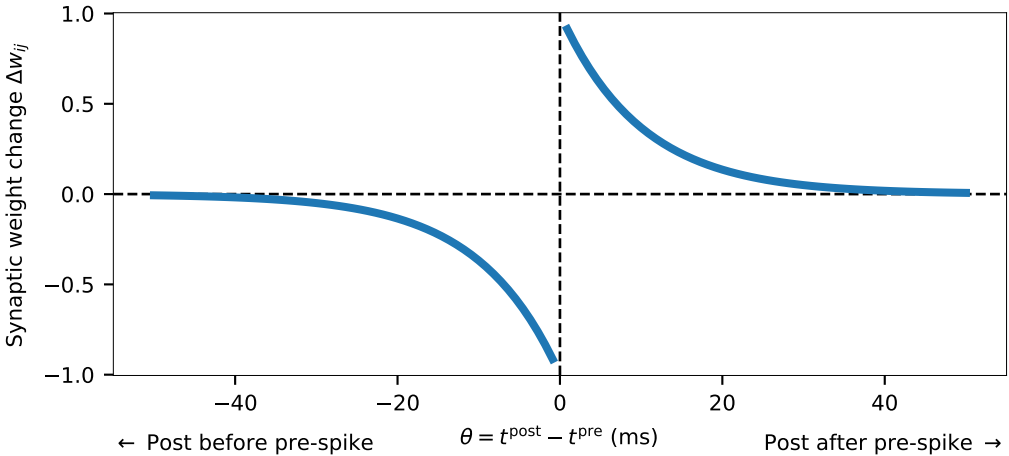
> When an axon of cell *A* is near enough to excite a cell *B* and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that *A*'s efficiency, as one of the cells firing *B*, is increased.
>
> — Donald O. Hebb, "The Organization of Behaviour", 1949

Example: Normalised Hebbian Learning



Learning an encoder $\mathbf{e}$, with $\|\mathbf{e}\| = 1$, 10000 steps, $\eta = 0.2 \times 10^{-4}$, $\Delta\mathbf{e} = \eta(\mathbf{x} \circ \mathbf{e})$

Spike-Time Dependent Plasticity



Conclusion

Supervised Learning

- Find $\mathbf{w}$ such that $f(\mathbf{x}_k; \mathbf{w}) \approx \mathbf{t}_k$
- *Hope:* $f(\mathbf{x}_k; \mathbf{w}) \approx f_{\mathrm{GT}}(\mathbf{x}_k)$
- Use gradient descent to find $\mathbf{w}$
- Delta, PES learning rules
- Modulatory synapses in the brain

Unsupervised Learning

- Dimensionality reduction $f(\mathbf{x}_k) = \lambda_k$
- *Hope:* latent dimensions λ are "meaningful"
- Autoencoders (nonlinear), PCA (linear)
- Hebbian learning $\Rightarrow$ learns PCA

Image sources

Title slide
Page from “Liber ethicorum des Henricus de Alemannia”. Title: “Henricus de Alemannia con i suoi studenti” (Henricus of Germany with his students), second half of 14th century.
From Wikimedia.