

SYDE 556/750

Simulating Neurobiological Systems
Lecture 8: Learning

Andreas Stöckel

February 25 & 27, 2020

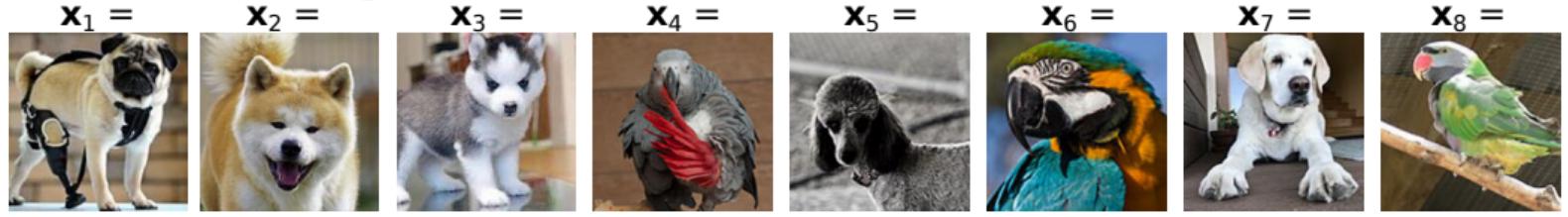


UNIVERSITY OF
WATERLOO

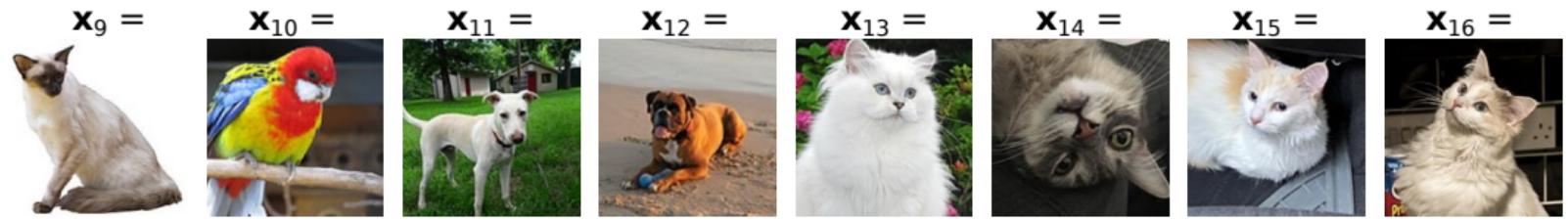
FACULTY OF
ENGINEERING



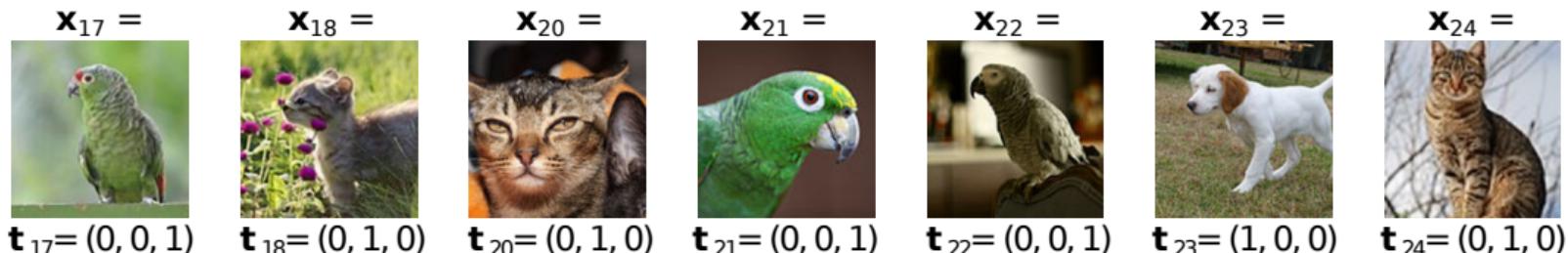
Supervised Learning



$$\mathbf{t}_1 = (1, 0, 0) \quad \mathbf{t}_2 = (1, 0, 0) \quad \mathbf{t}_3 = (1, 0, 0) \quad \mathbf{t}_4 = (0, 0, 1) \quad \mathbf{t}_5 = (1, 0, 0) \quad \mathbf{t}_6 = (0, 0, 1) \quad \mathbf{t}_7 = (1, 0, 0) \quad \mathbf{t}_8 = (0, 0, 1)$$

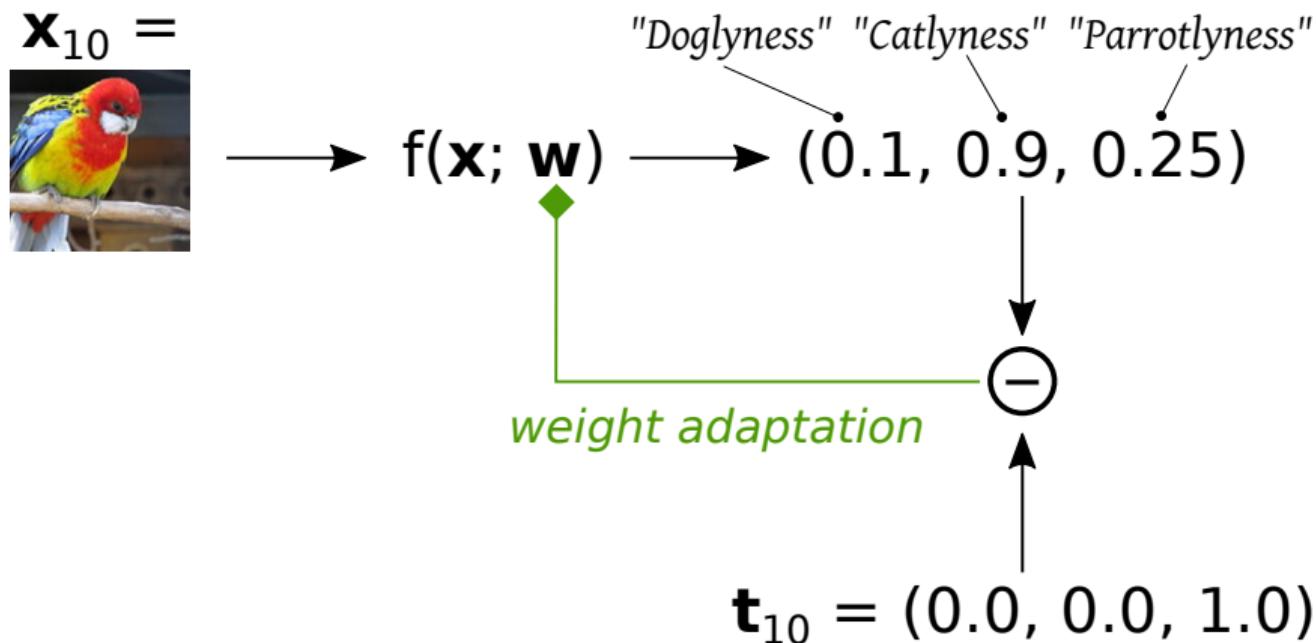


$$\mathbf{t}_9 = (0, 1, 0) \quad \mathbf{t}_{10} = (0, 0, 1) \quad \mathbf{t}_{11} = (1, 0, 0) \quad \mathbf{t}_{12} = (1, 0, 0) \quad \mathbf{t}_{13} = (0, 1, 0) \quad \mathbf{t}_{14} = (0, 1, 0) \quad \mathbf{t}_{15} = (0, 1, 0) \quad \mathbf{t}_{16} = (0, 1, 0)$$

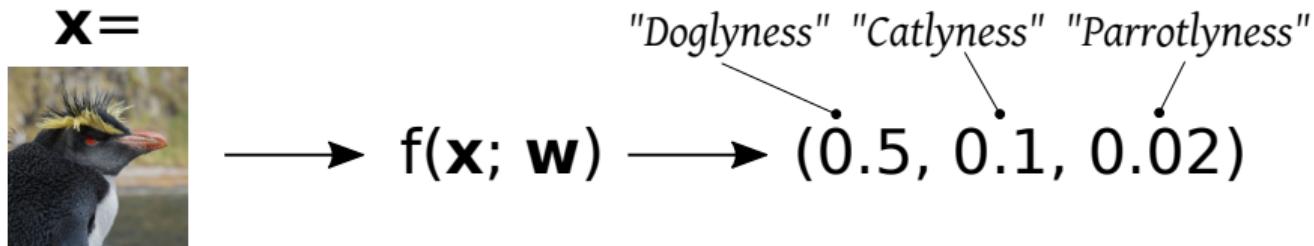


$$\mathbf{t}_{17} = (0, 0, 1) \quad \mathbf{t}_{18} = (0, 1, 0) \quad \mathbf{t}_{20} = (0, 1, 0) \quad \mathbf{t}_{21} = (0, 0, 1) \quad \mathbf{t}_{22} = (0, 0, 1) \quad \mathbf{t}_{23} = (1, 0, 0) \quad \mathbf{t}_{24} = (0, 1, 0)$$

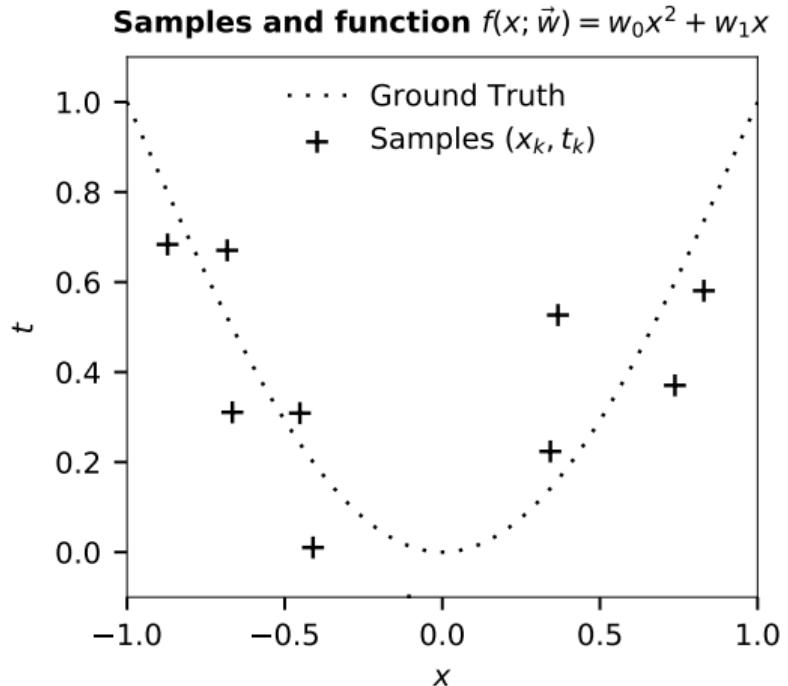
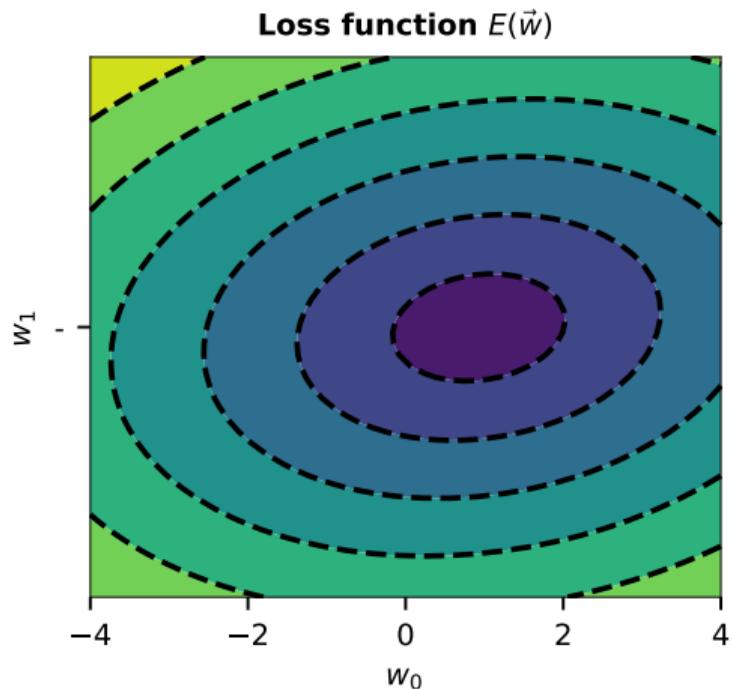
Supervised Learning – Training



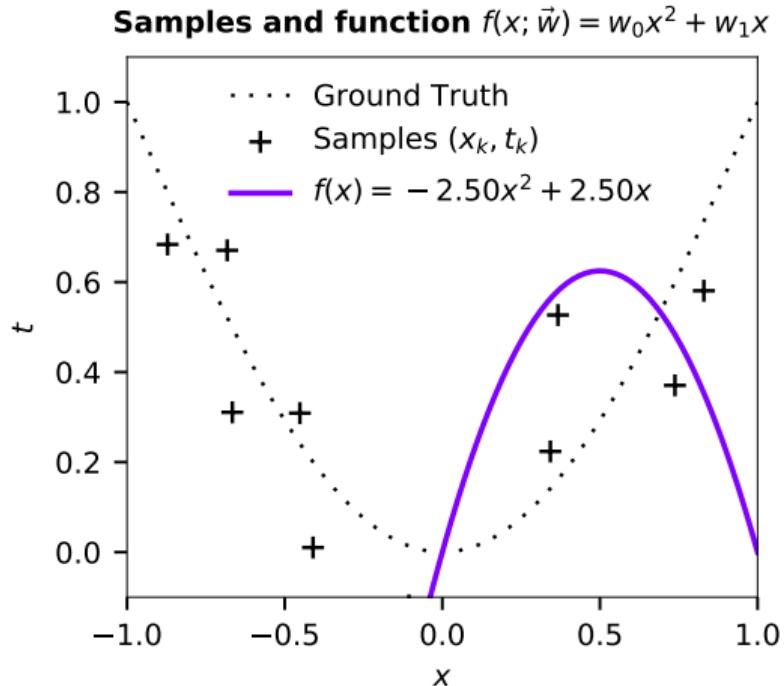
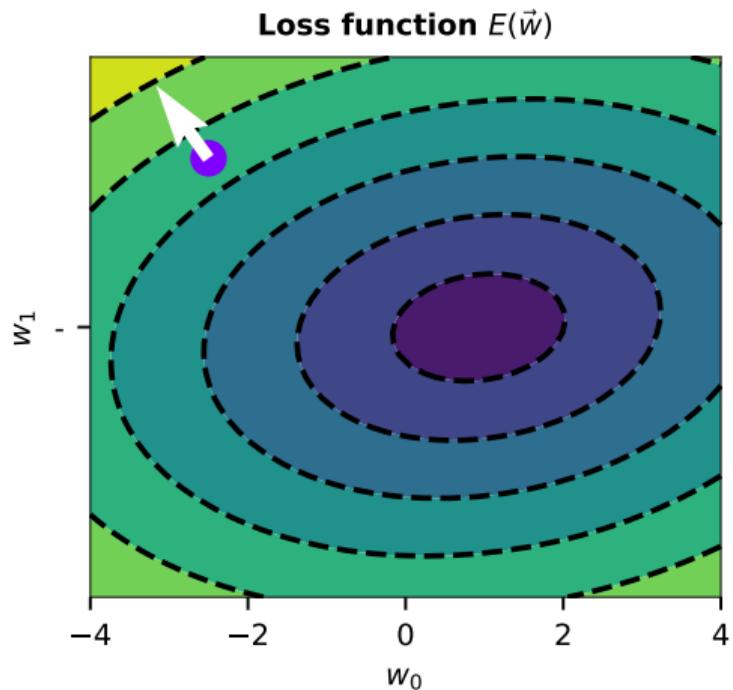
Supervised Learning – Inference



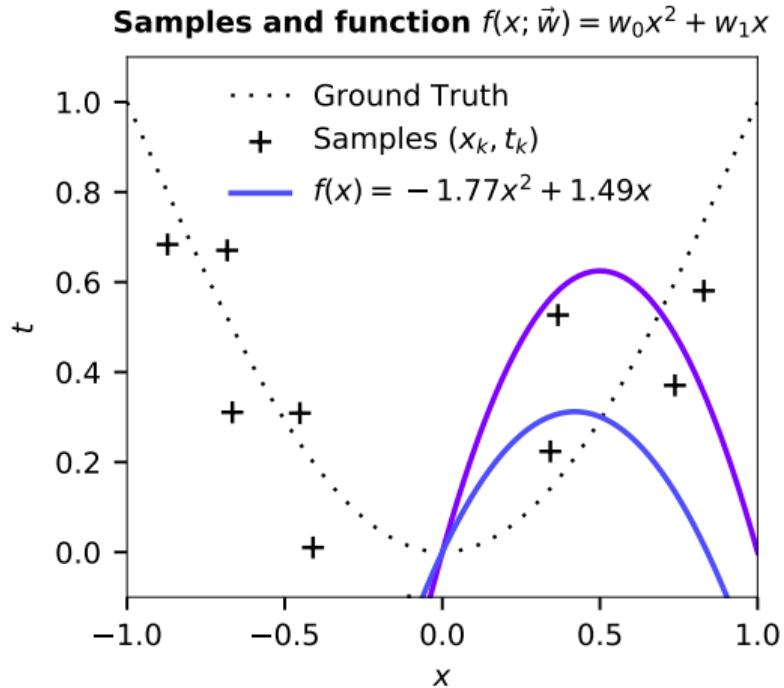
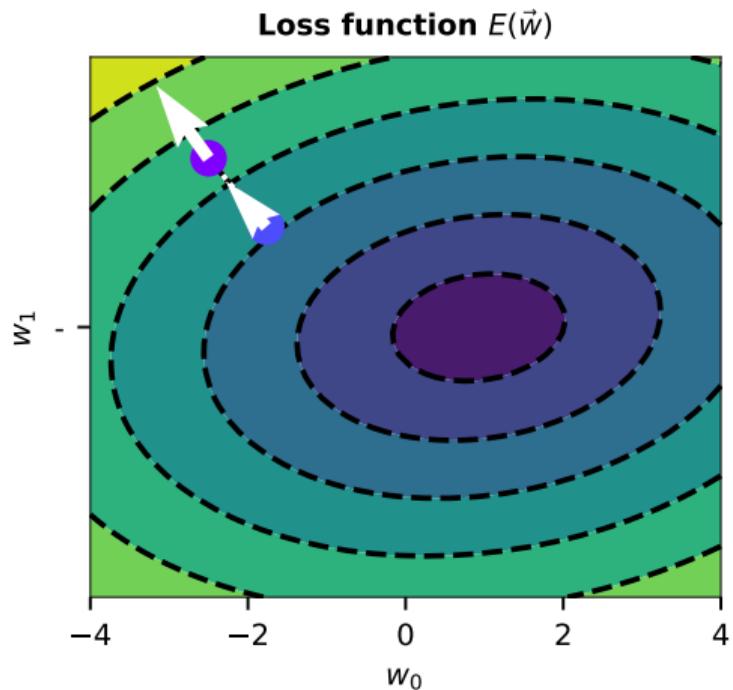
Gradient Descent – Example



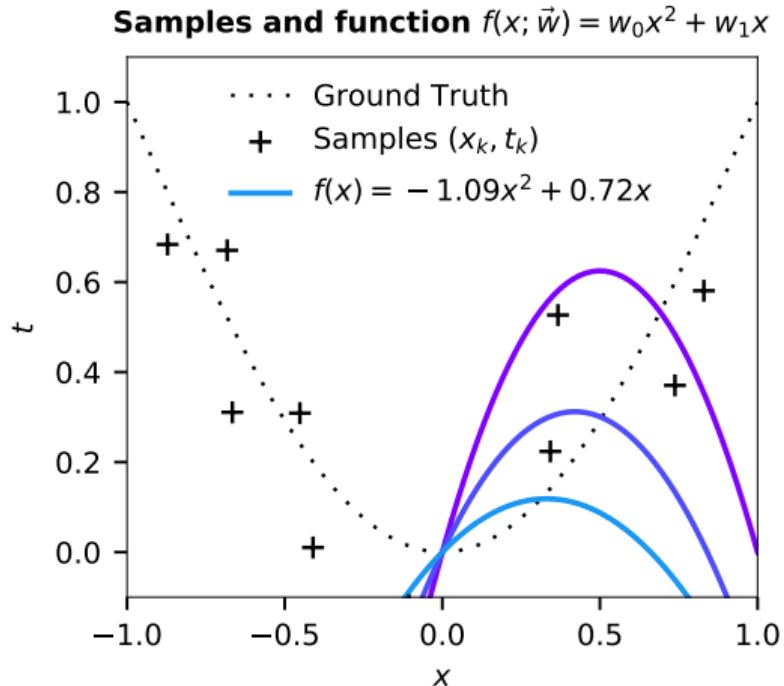
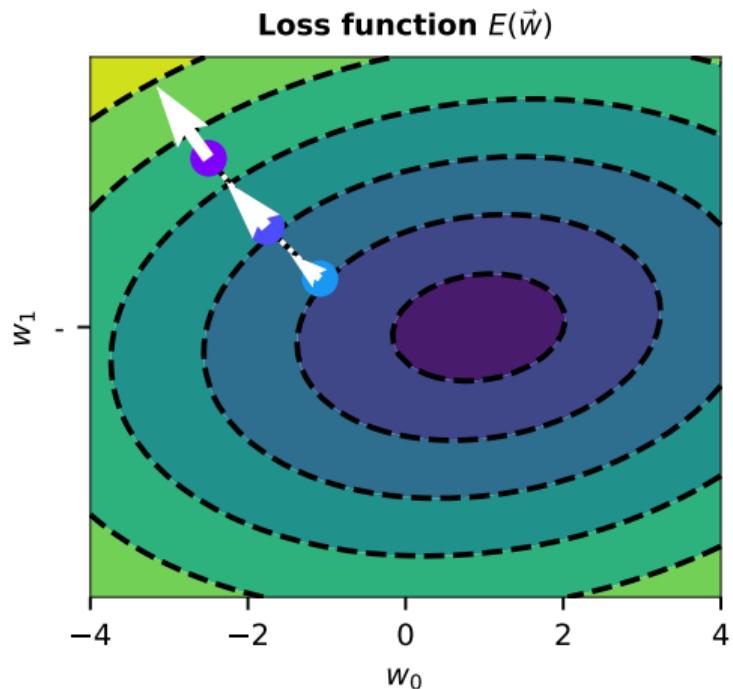
Gradient Descent – Example



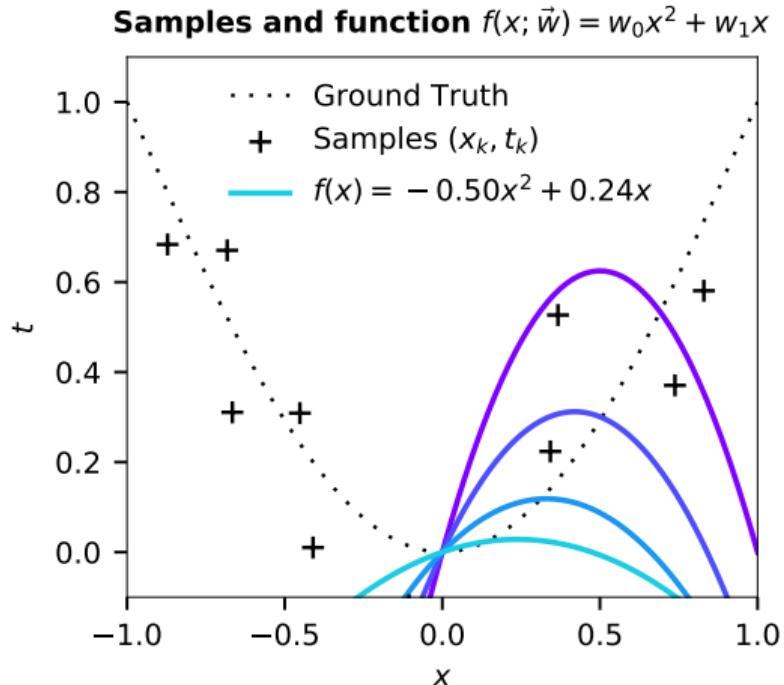
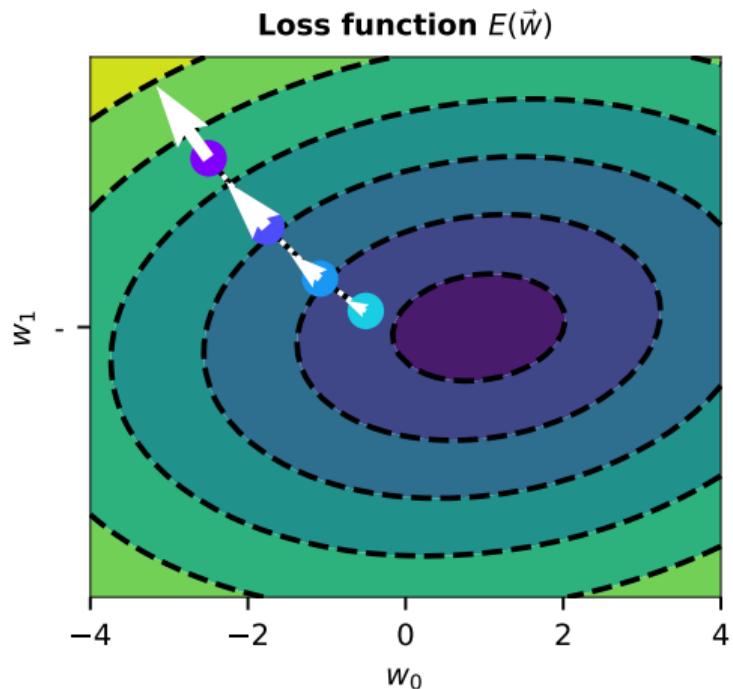
Gradient Descent – Example



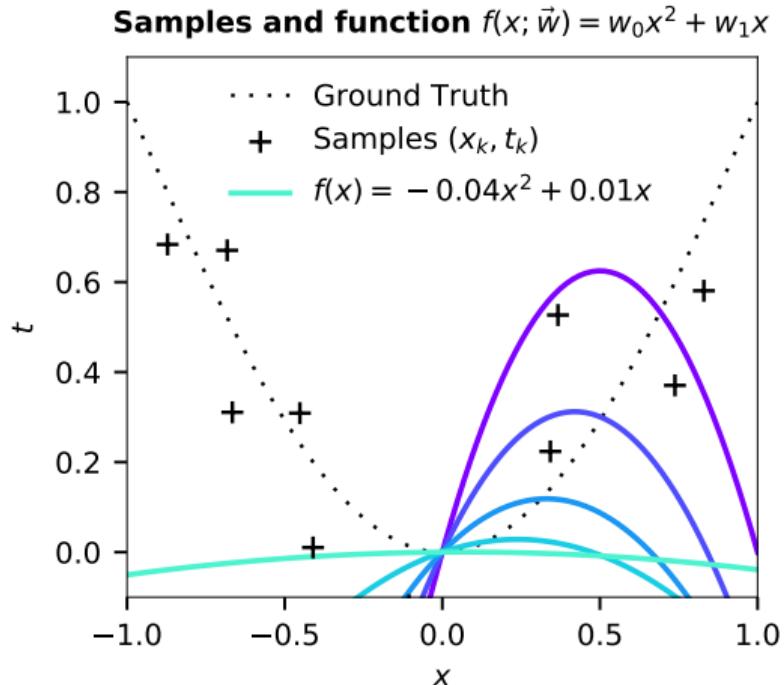
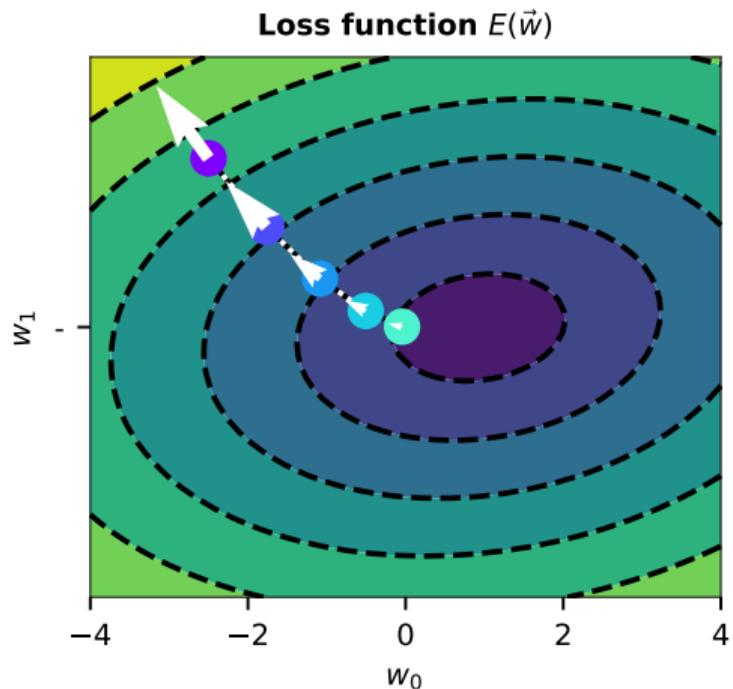
Gradient Descent – Example



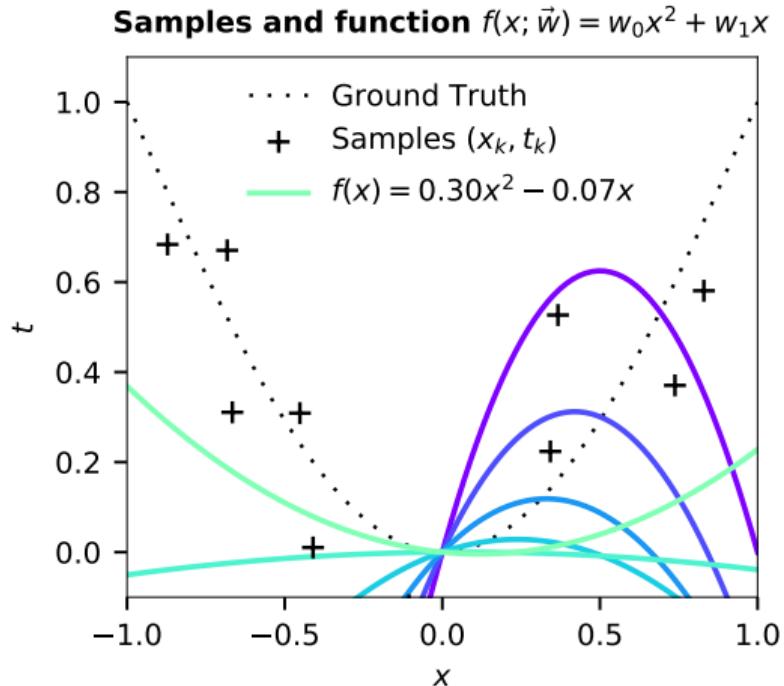
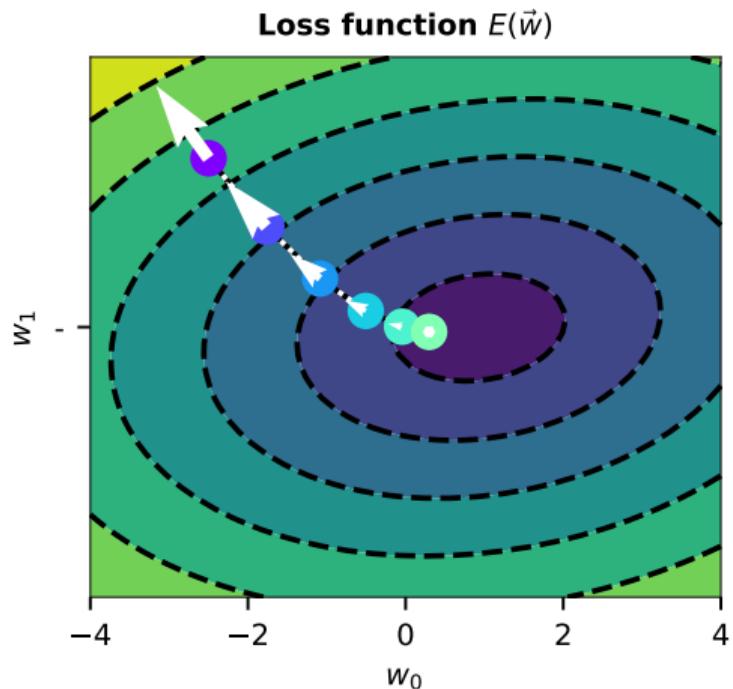
Gradient Descent – Example



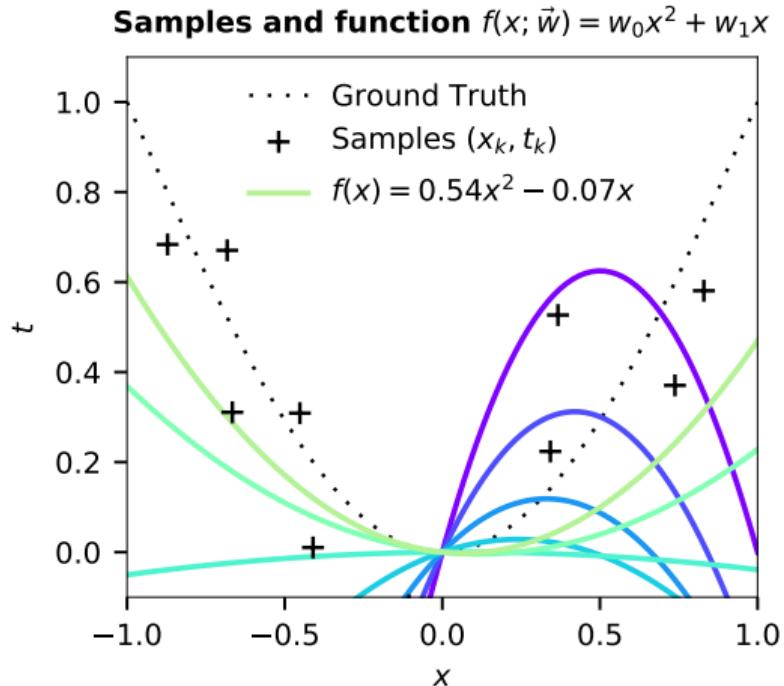
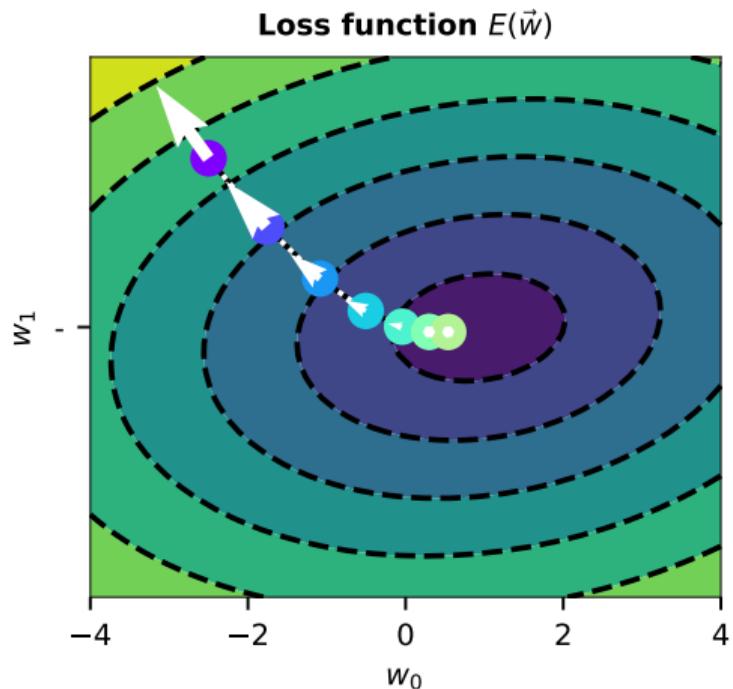
Gradient Descent – Example



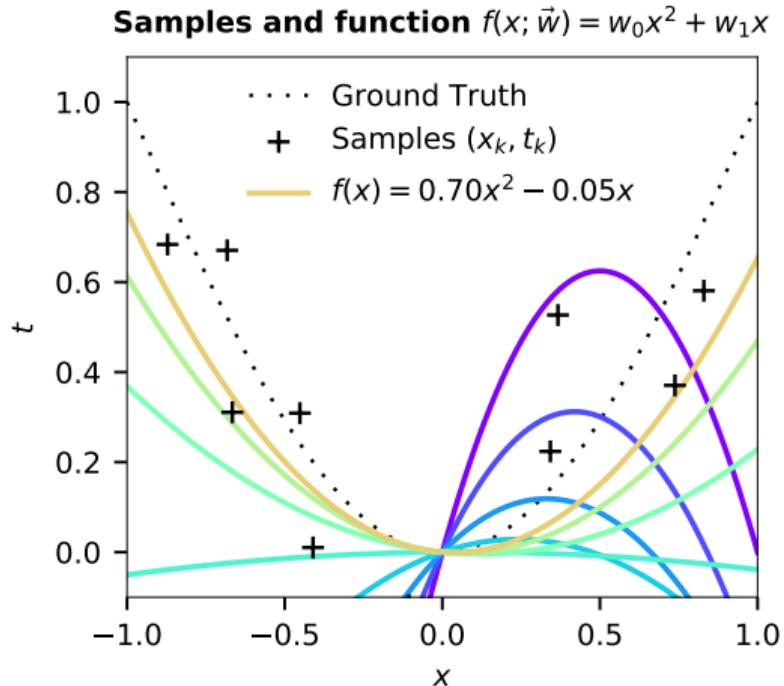
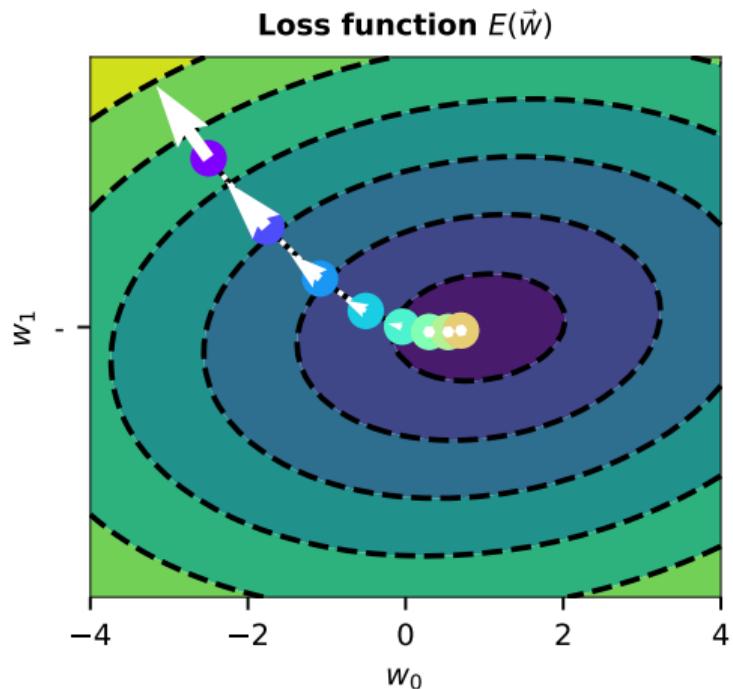
Gradient Descent – Example



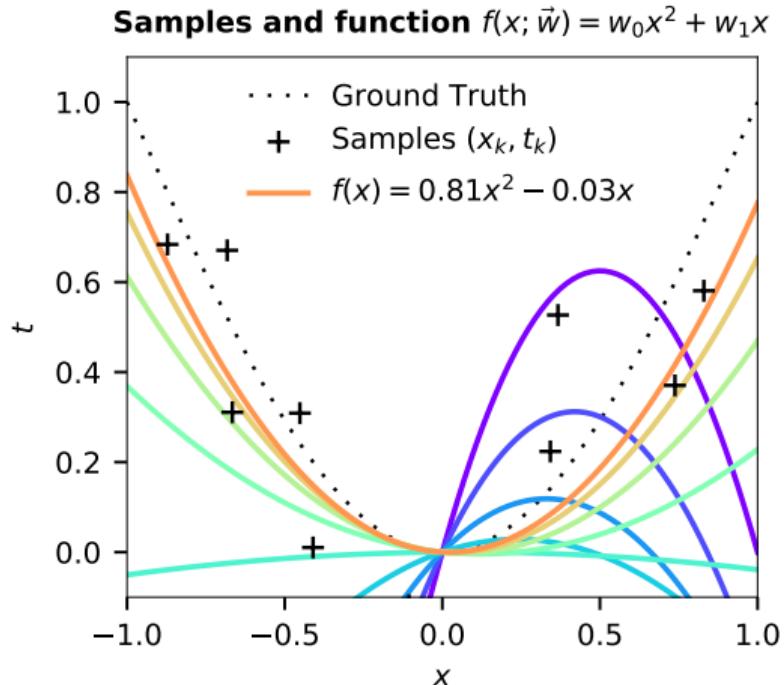
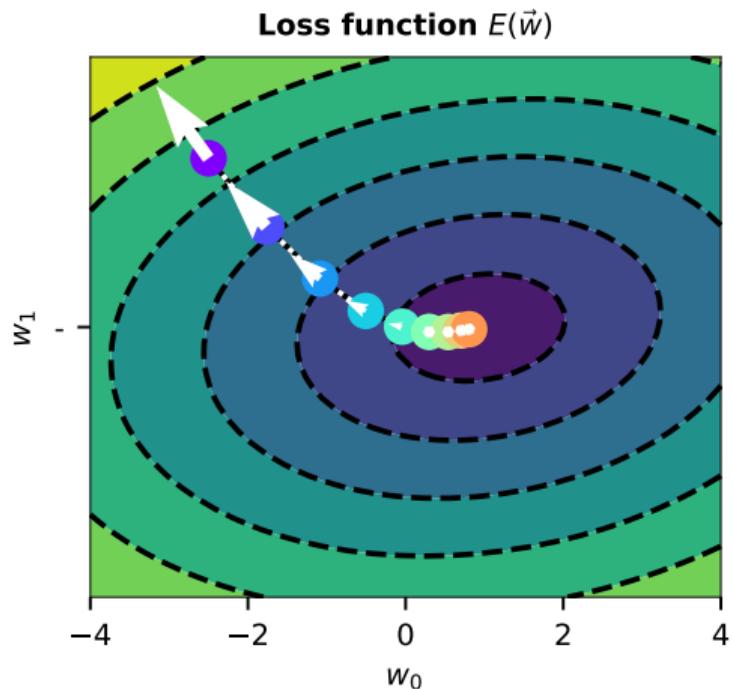
Gradient Descent – Example



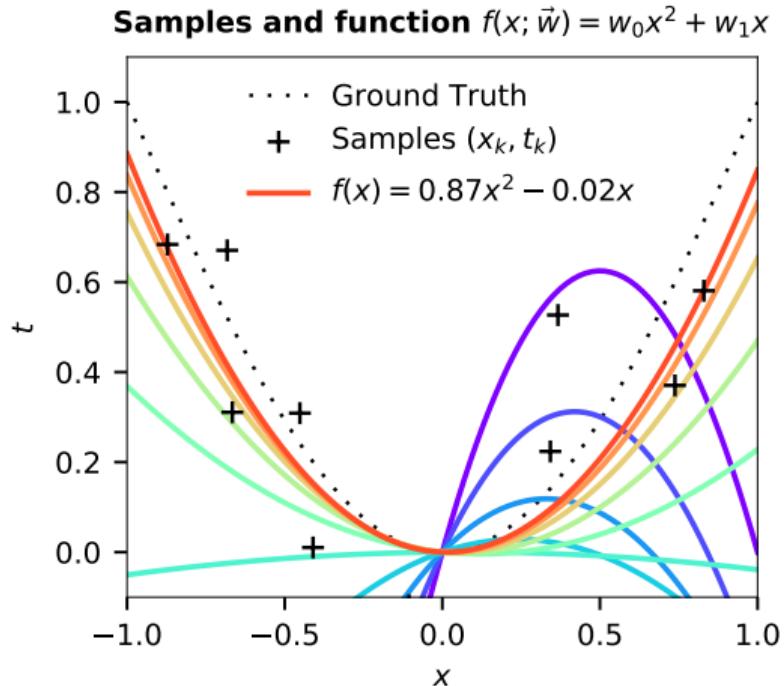
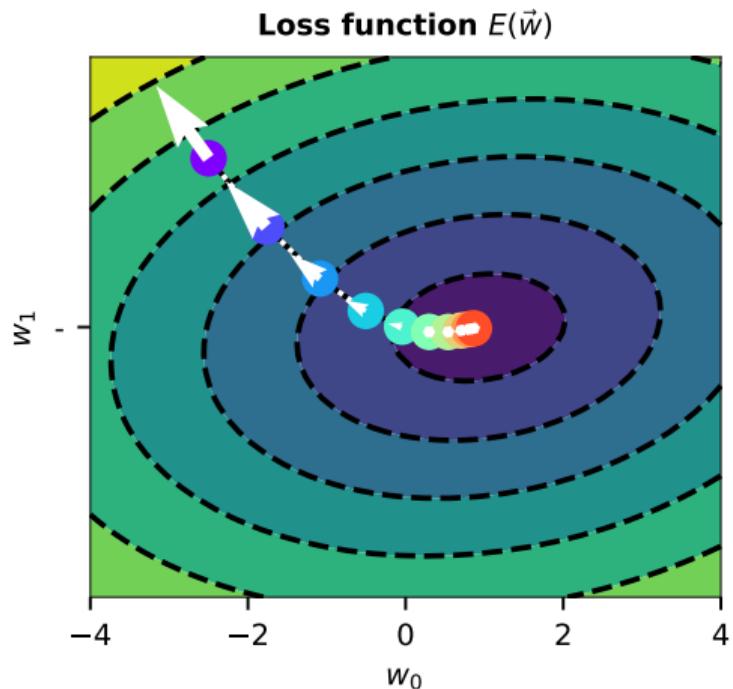
Gradient Descent – Example



Gradient Descent – Example

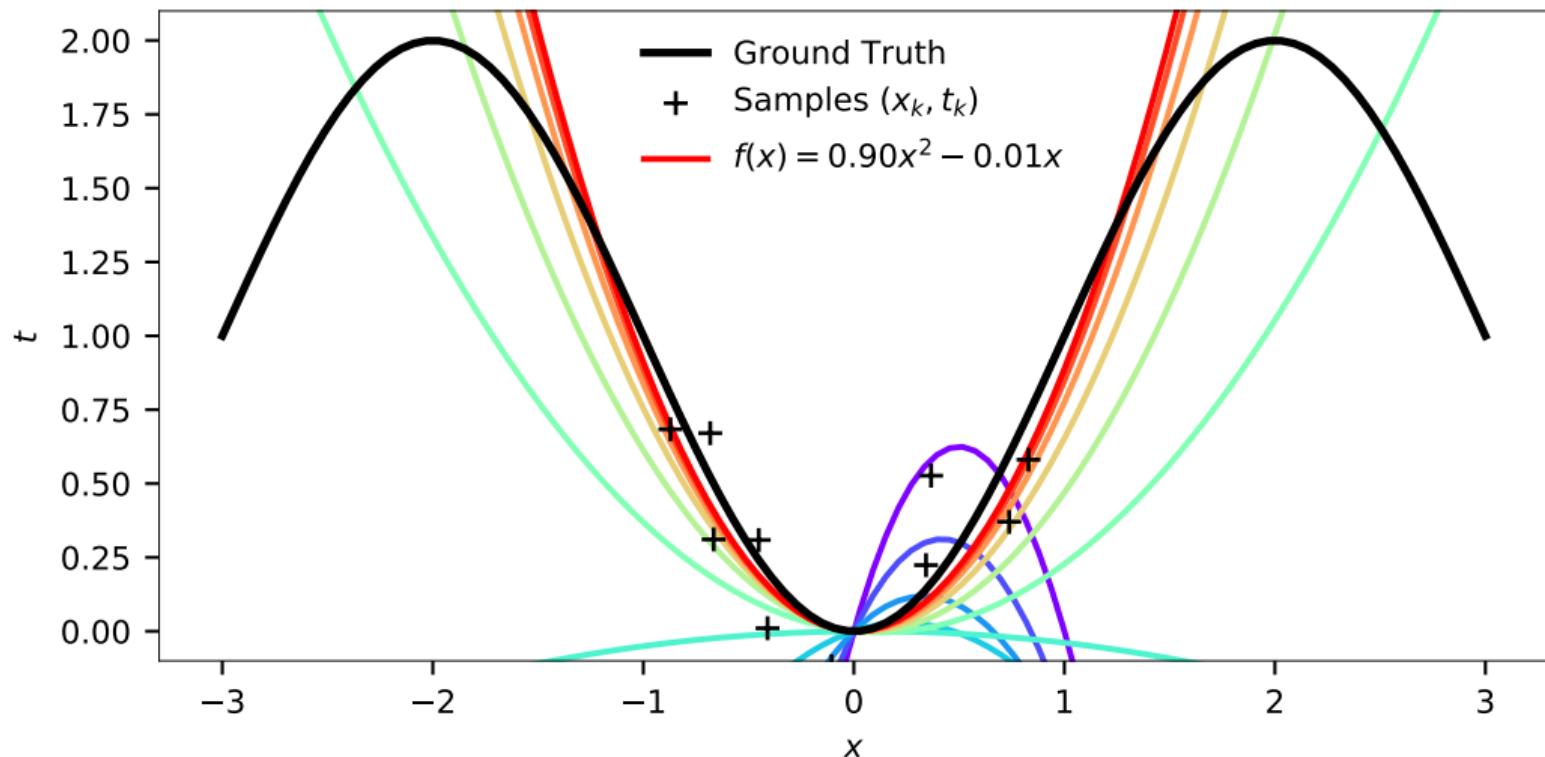


Gradient Descent – Example



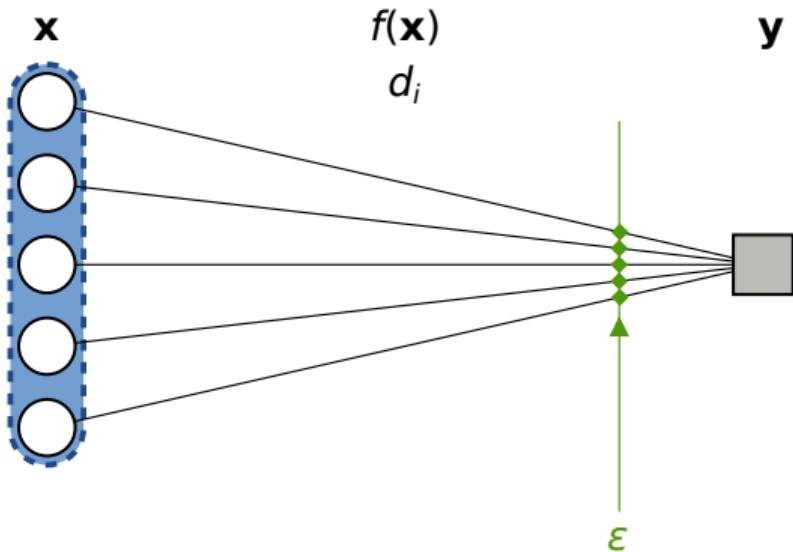
Supervised Learning – Generalisation

Samples and function $f(x; \vec{w}) = w_0x^2 + w_1x$



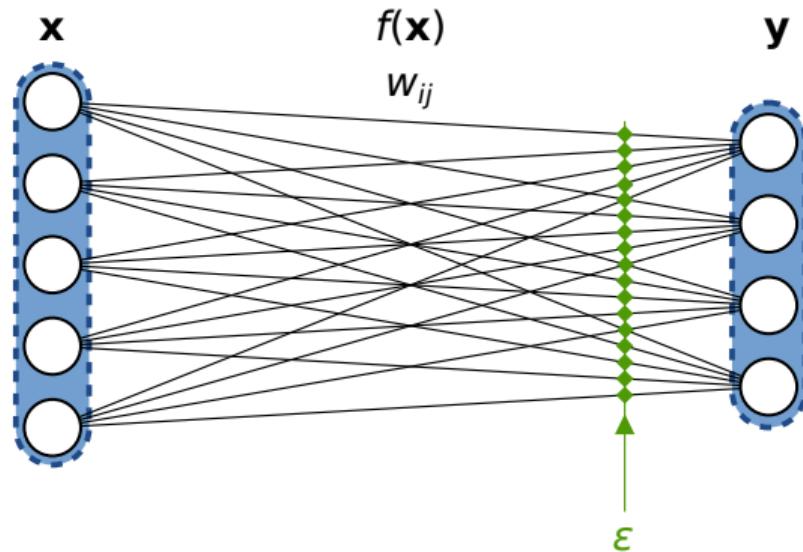
Learning Decoders and Learning Weights

Learning Decoders (Delta Rule)



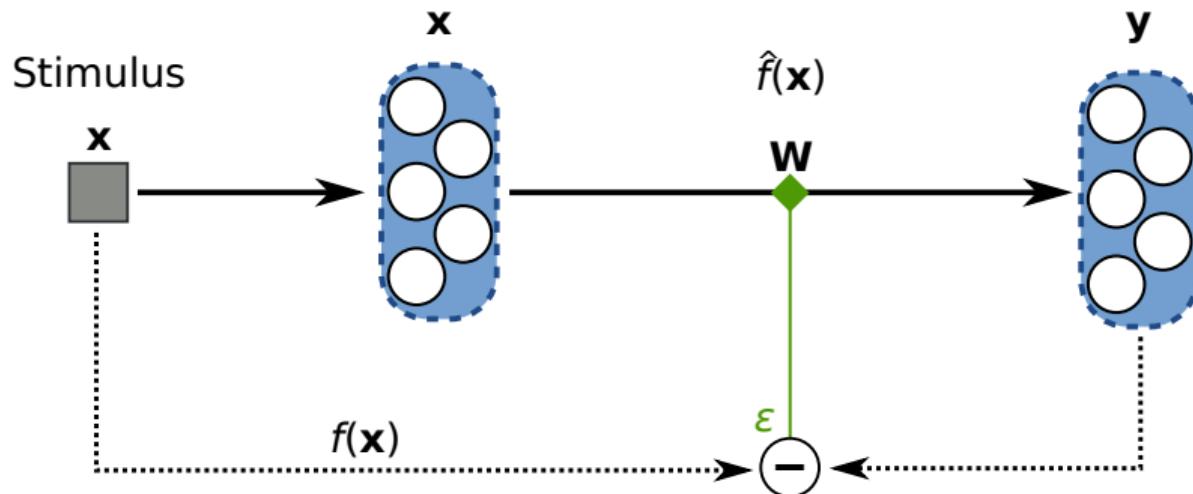
$$\Delta d_i = -\eta a_i(\mathbf{x}) \underbrace{(y(t) - y^d(t))}_{\varepsilon(t)}$$

Learning Weights (PES Rule)



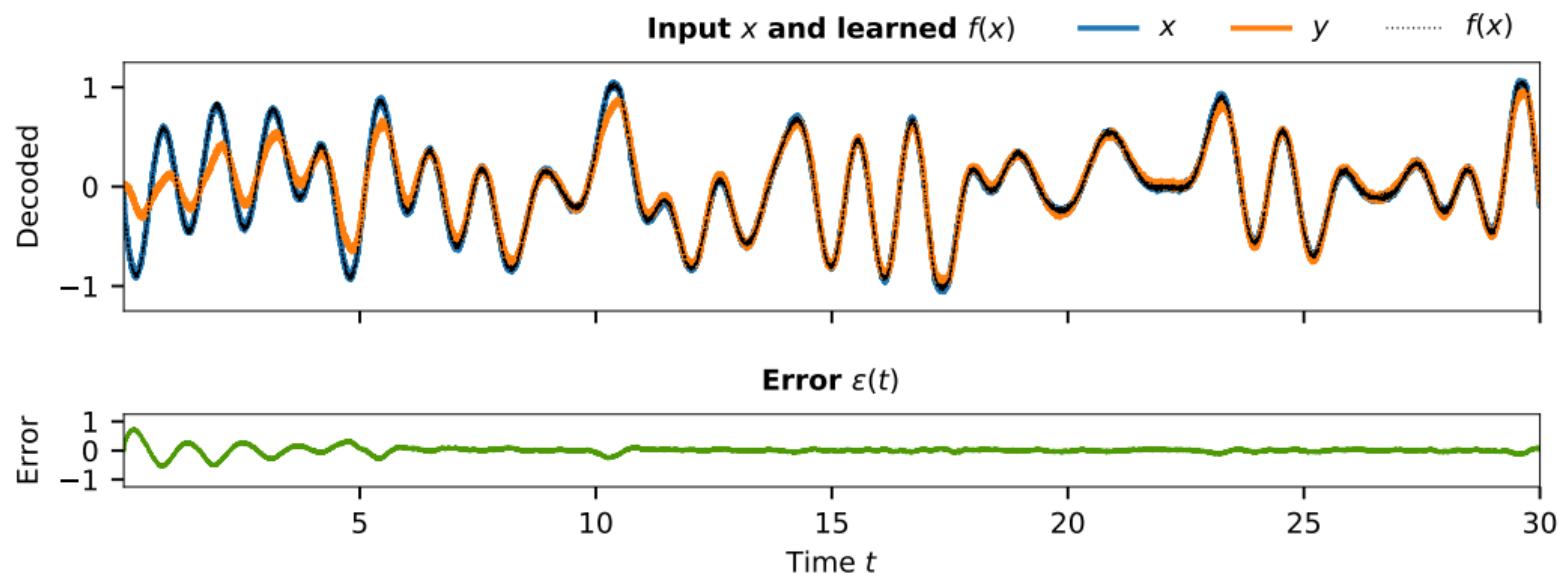
$$\Delta w_{ij} = -\eta a_i(\mathbf{x}) \left(\alpha_j \langle \mathbf{e}_j, \varepsilon(t) \rangle \right)$$

Example: Learning Functions (I)



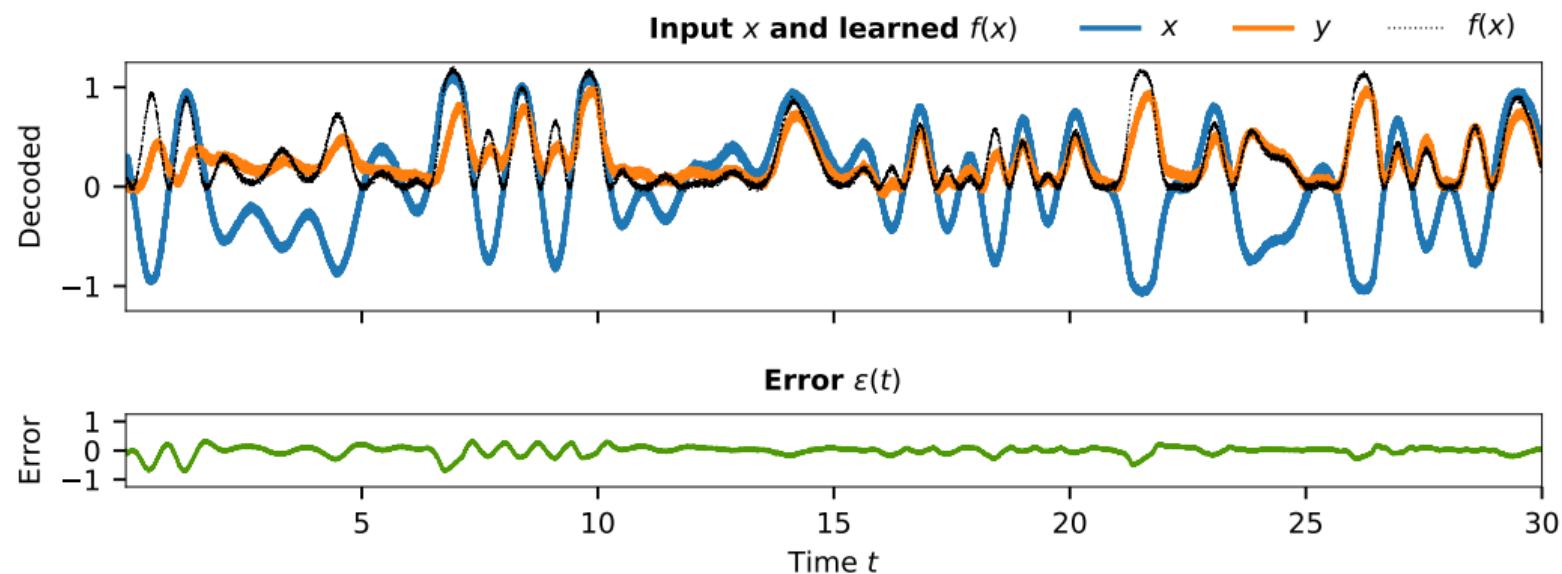
Example: Learning Functions (II)

Communication Channel $f(x) = x$



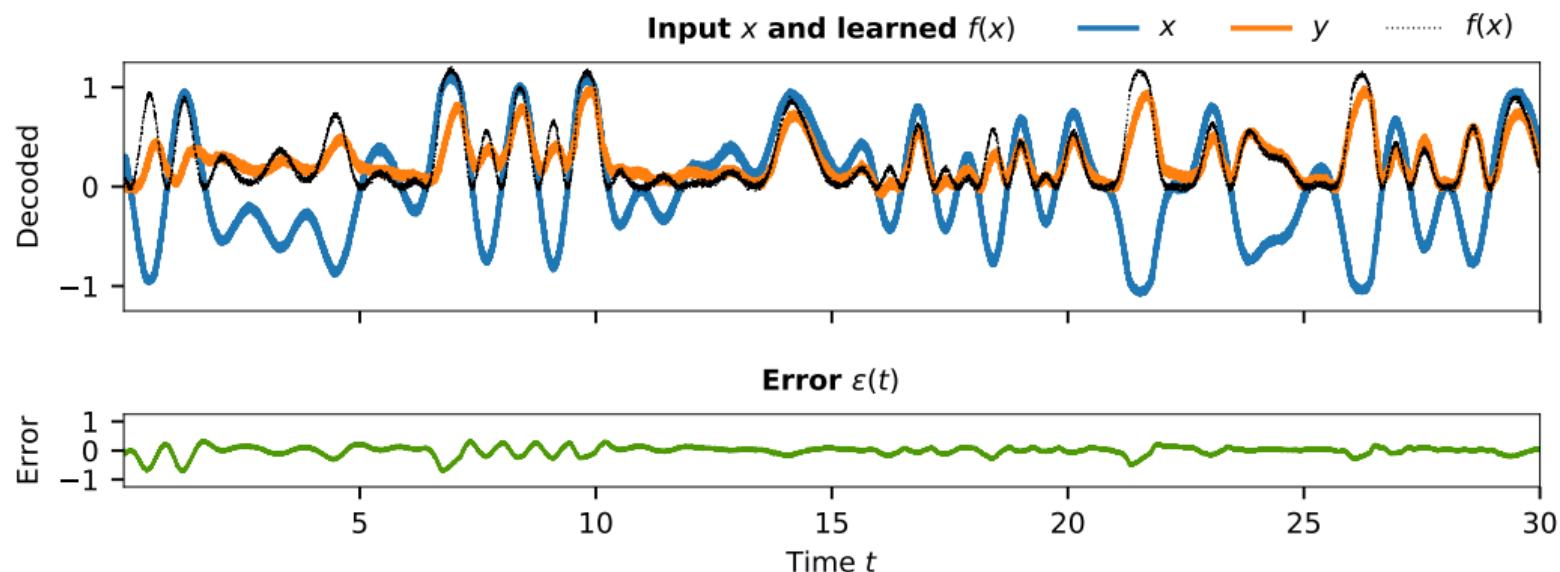
Example: Learning Functions (III)

$$\text{Square } f(x) = x^2$$



Example: Learning Functions (III)

Square $f(x) = x^2$

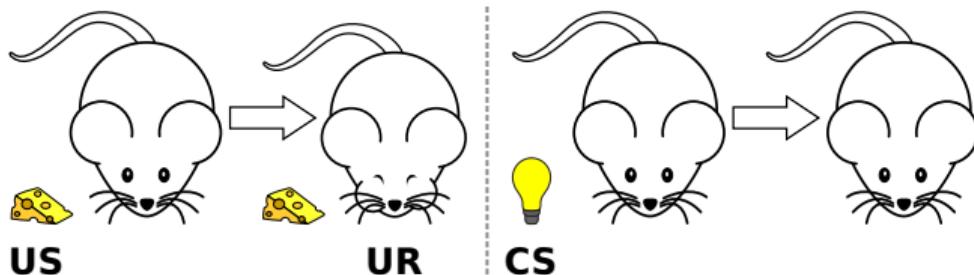


Works, but learns more slowly!

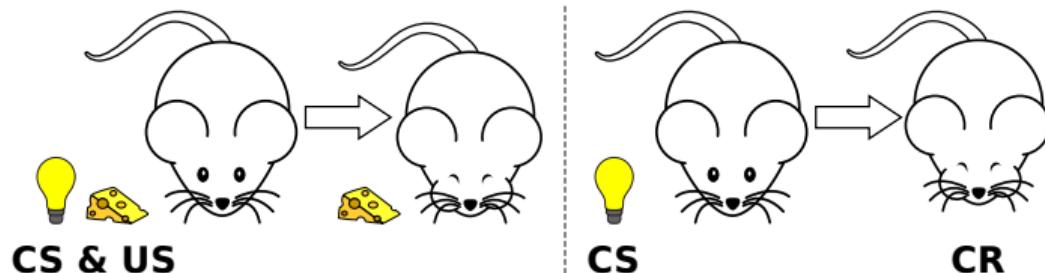
Where is the error signal $\varepsilon(t)$ coming from?

Example: Classical Conditioning (I)

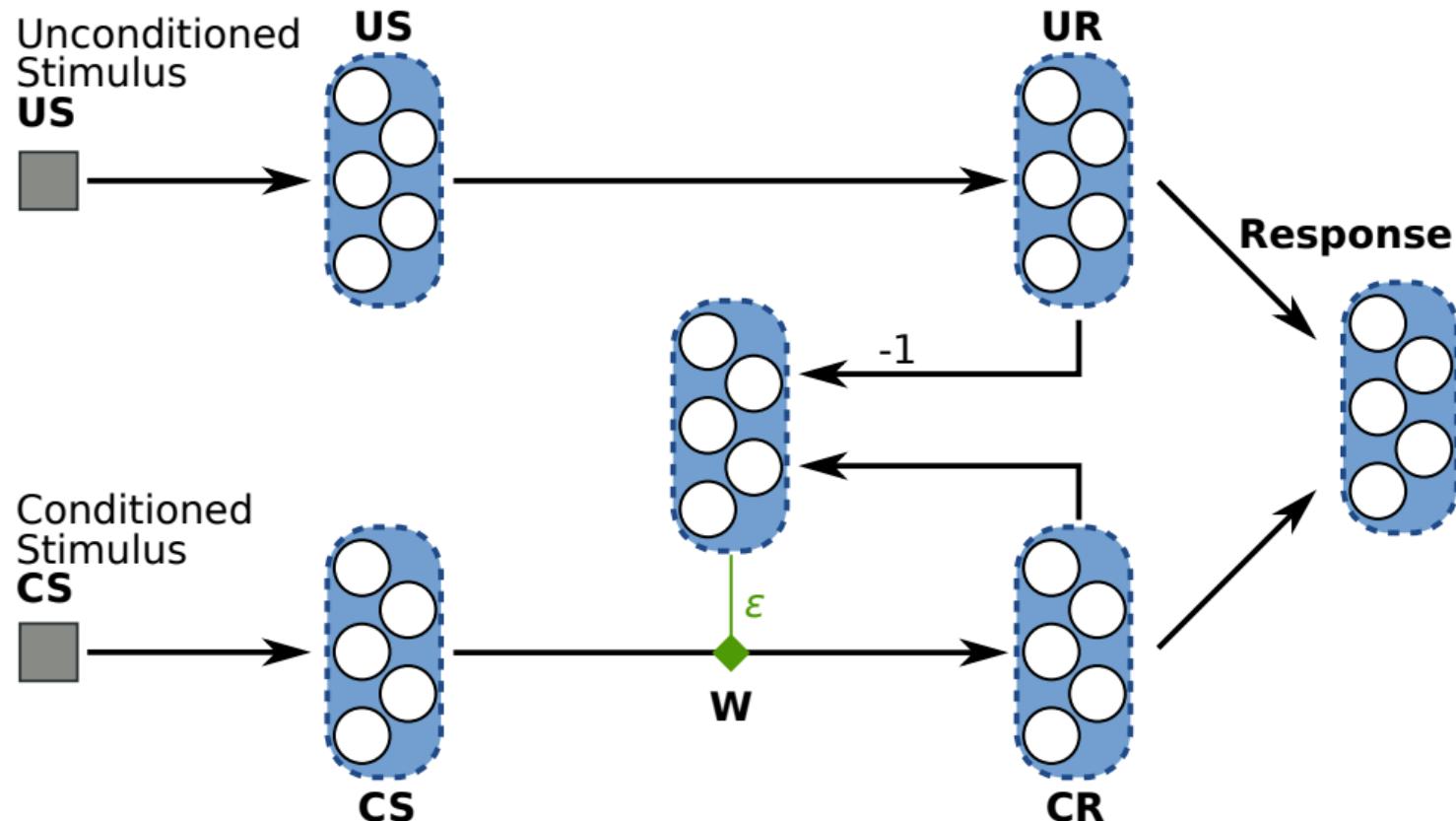
Before conditioning:



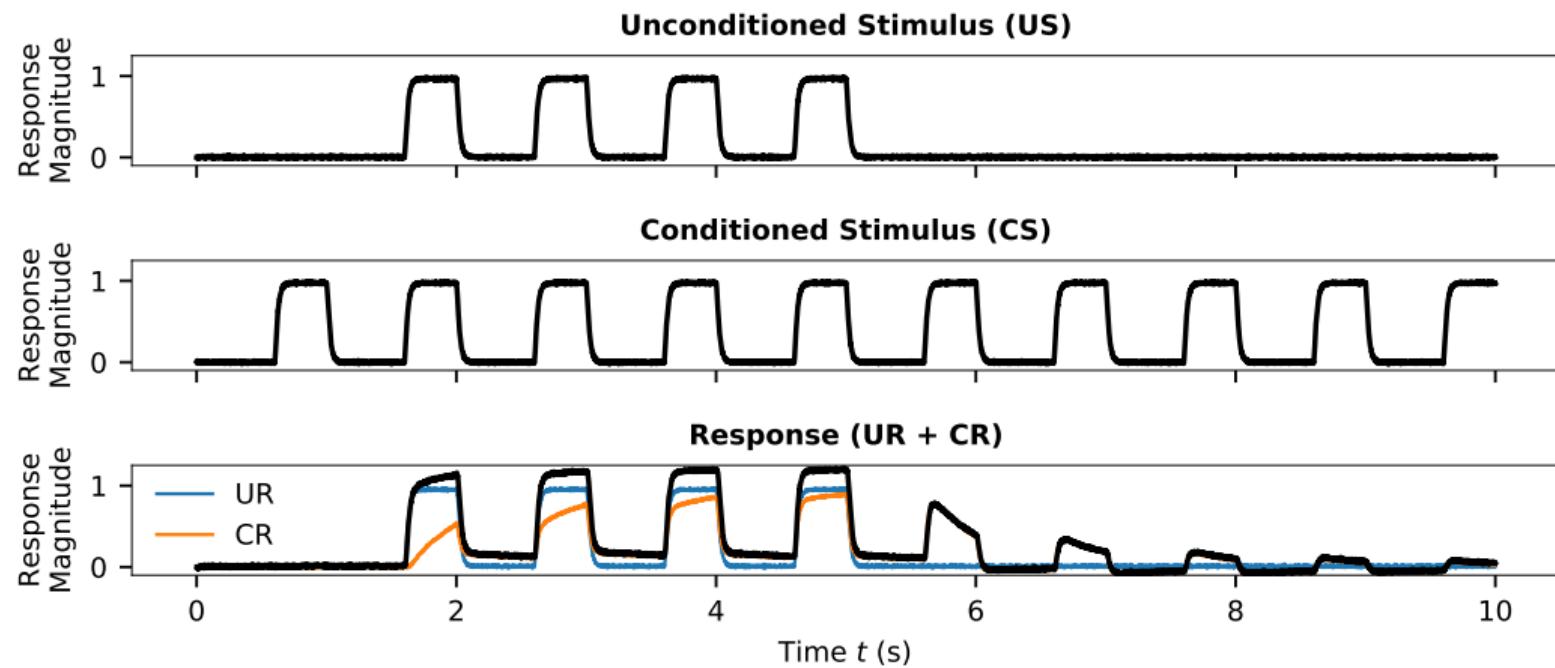
After conditioning:



Example: Classical Conditioning (II)

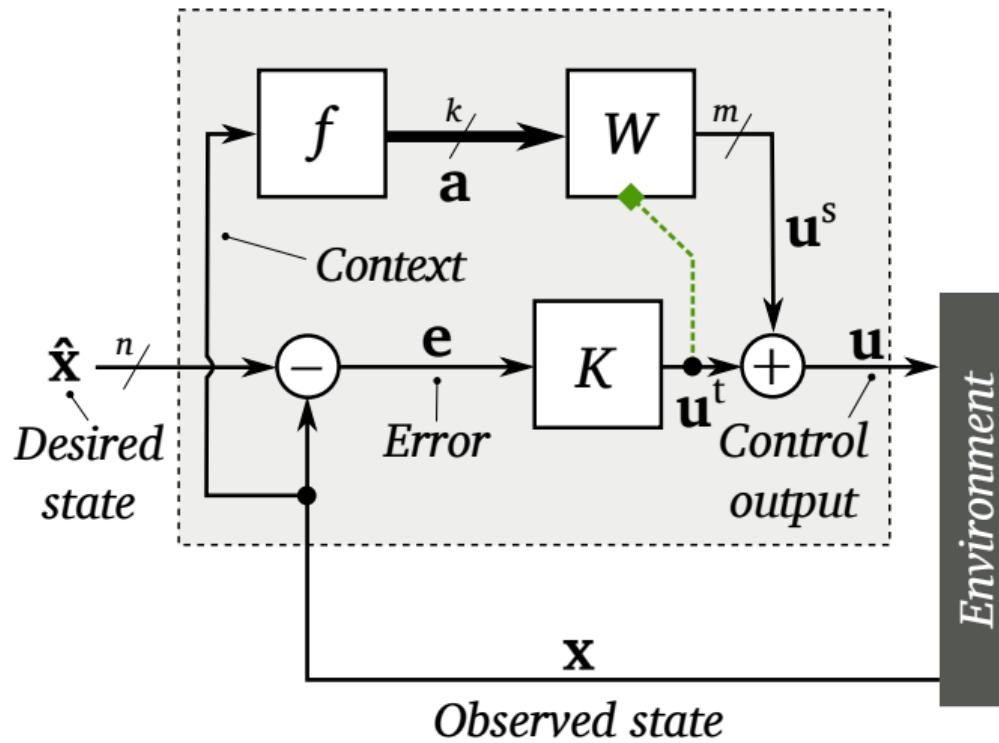


Example: Classical Conditioning (III)

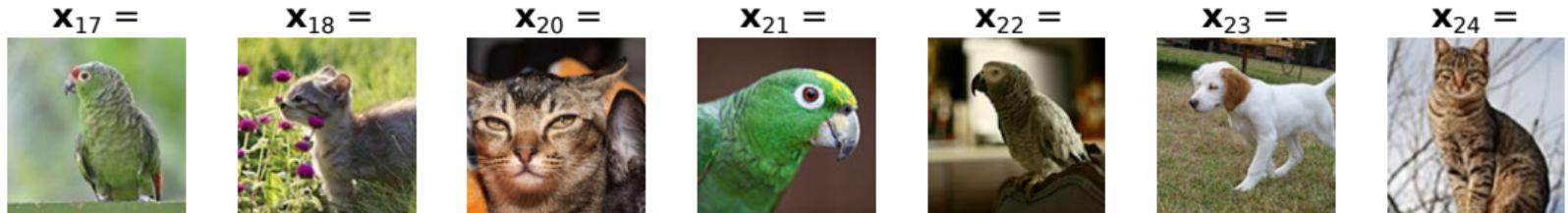
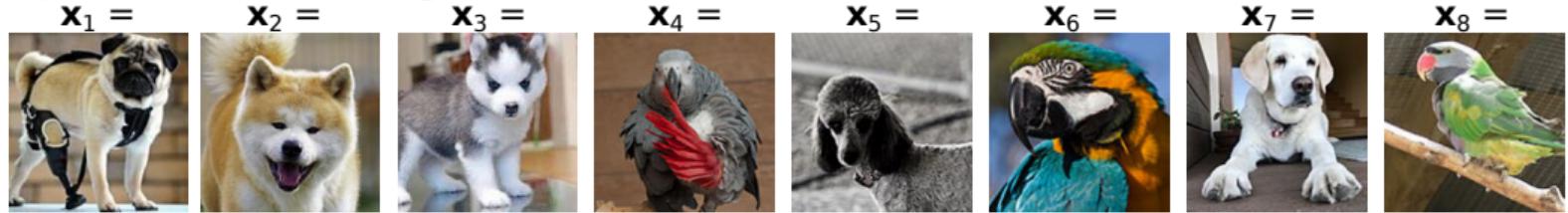


Example: Adaptive Controller

Adaptive Controller



Unsupervised Learning



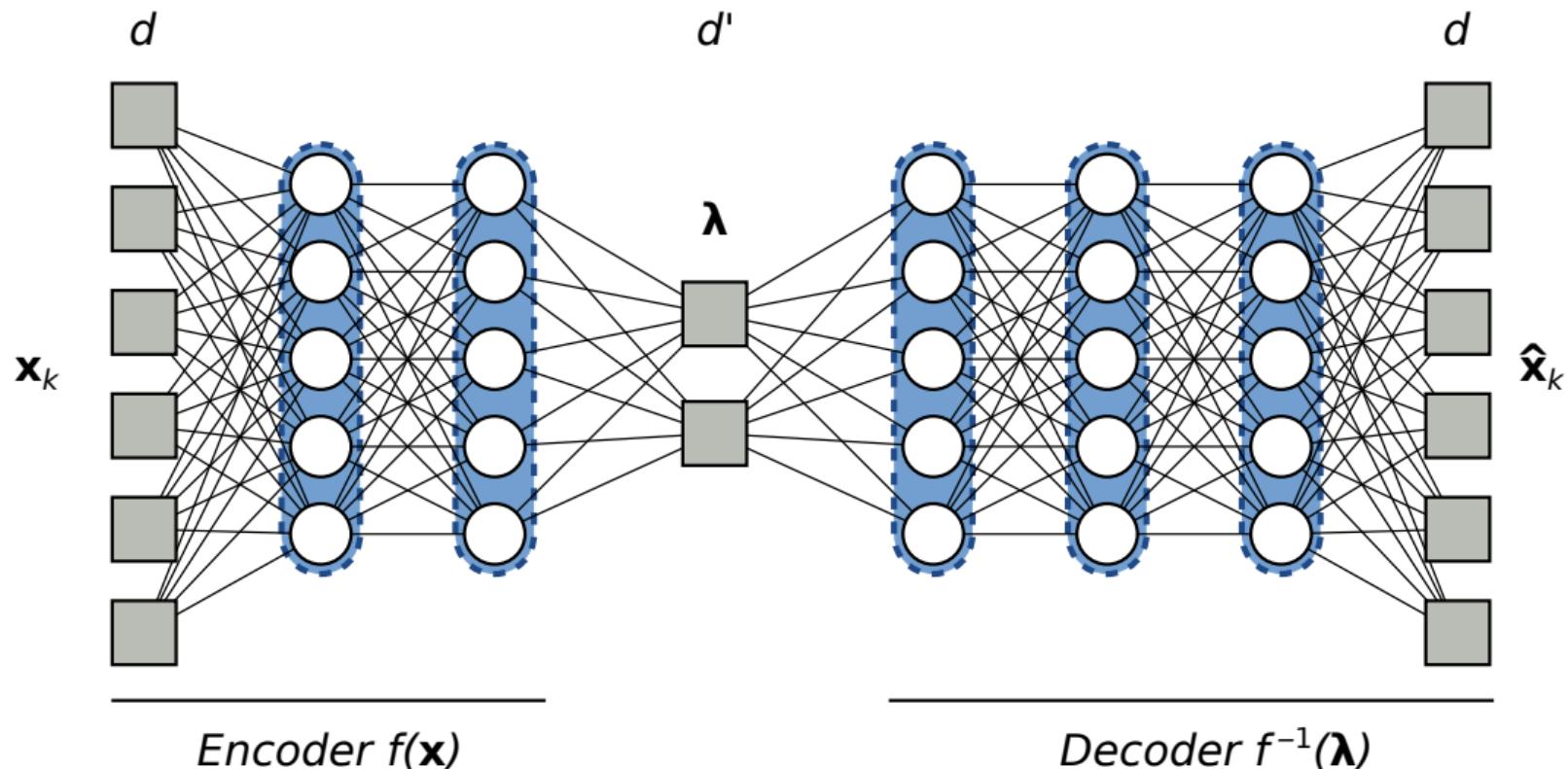
Unsupervised Learning – Training

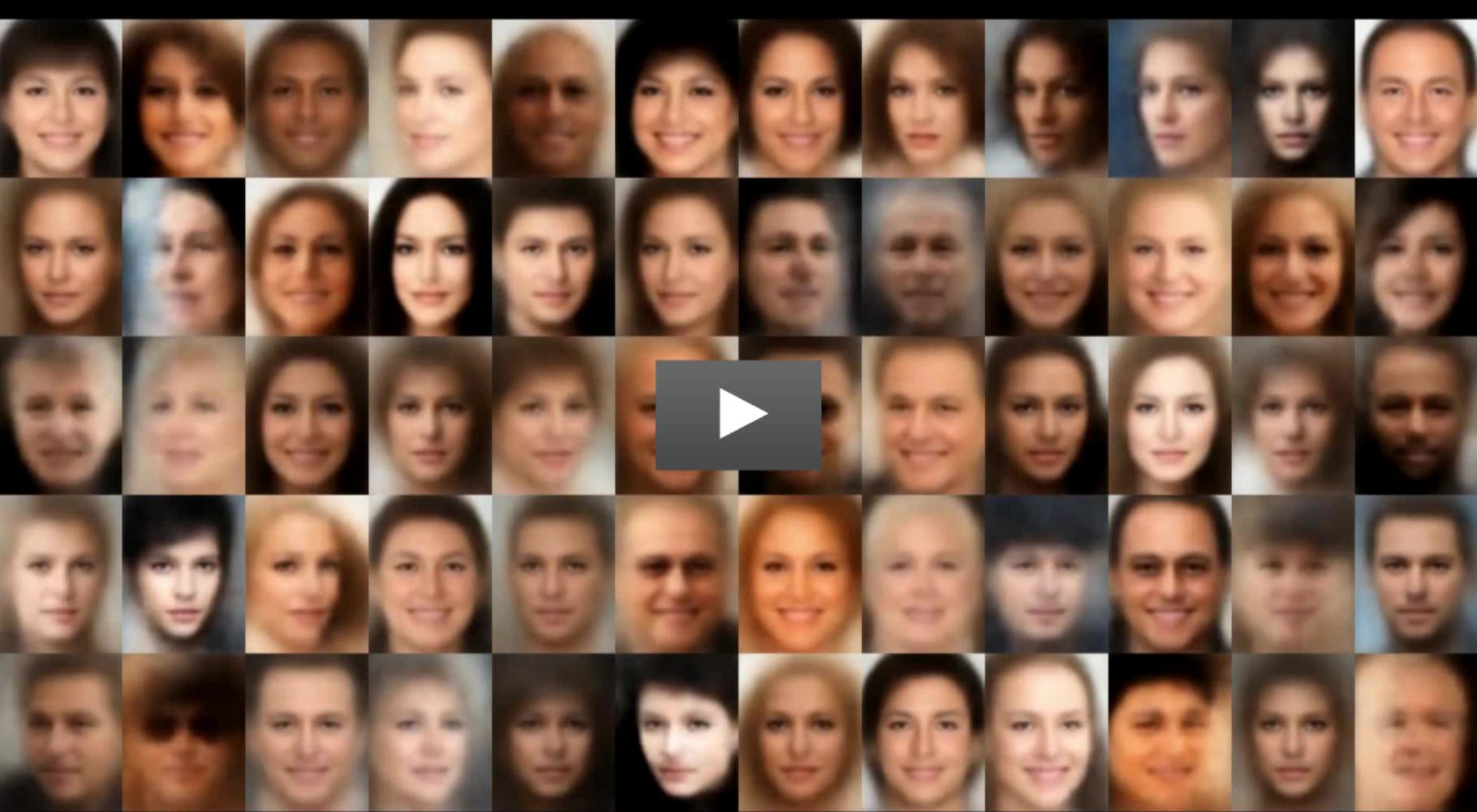


Unsupervised Learning – Inference



Autoencoder





PCA in 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
```

```
🐍 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

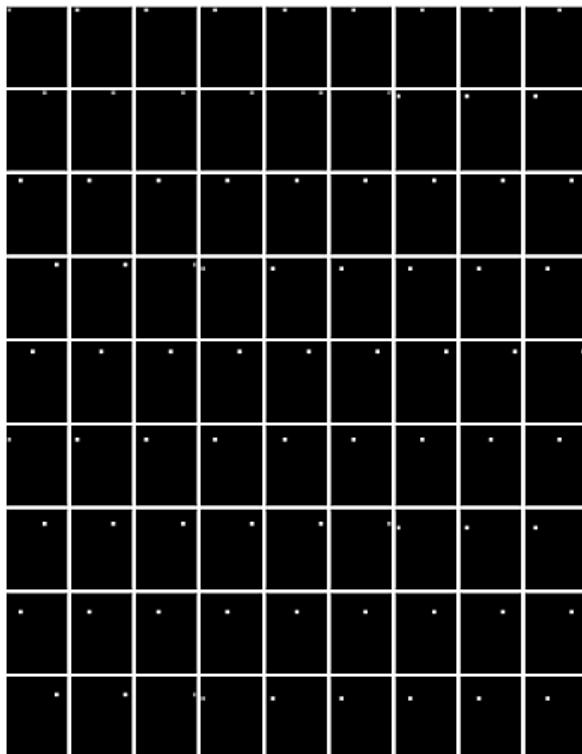
- ▶ 84 images of 12 women with 7 different expressions
- ▶ Normalised eye location
- ▶ 45×60 pixels (2700 dimensions)
- ▶ Greyscale



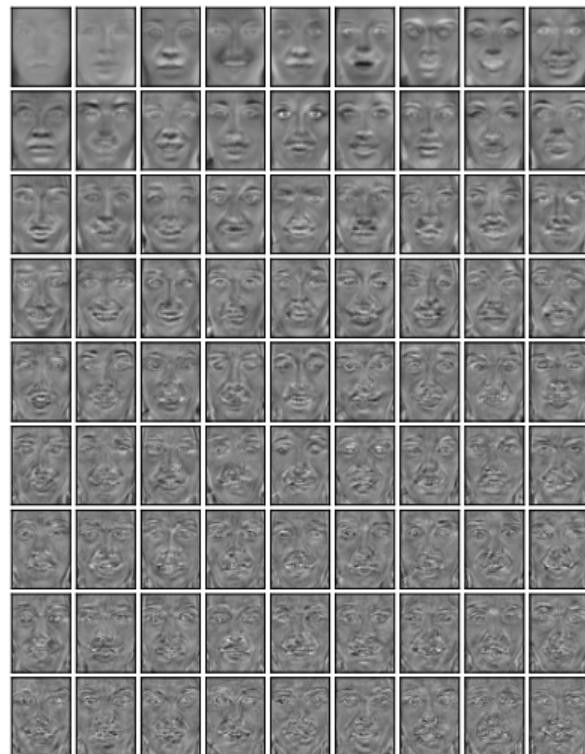
Image Sources. "Pain Expression Subset;" http://pics.stir.ac.uk/2D_face_sets.htm

PCA Example: Eigenfaces

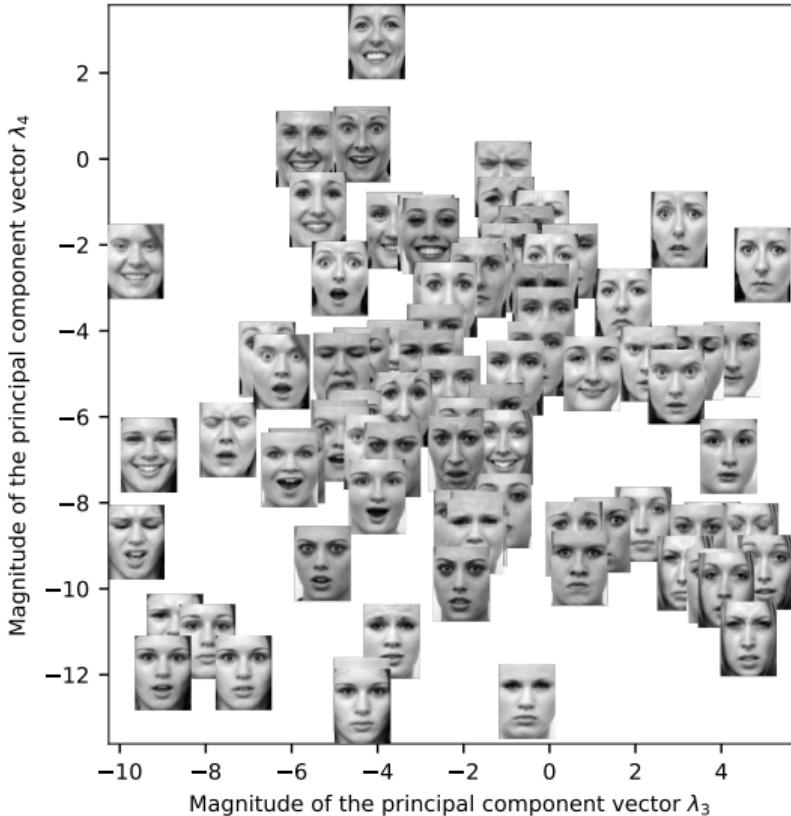
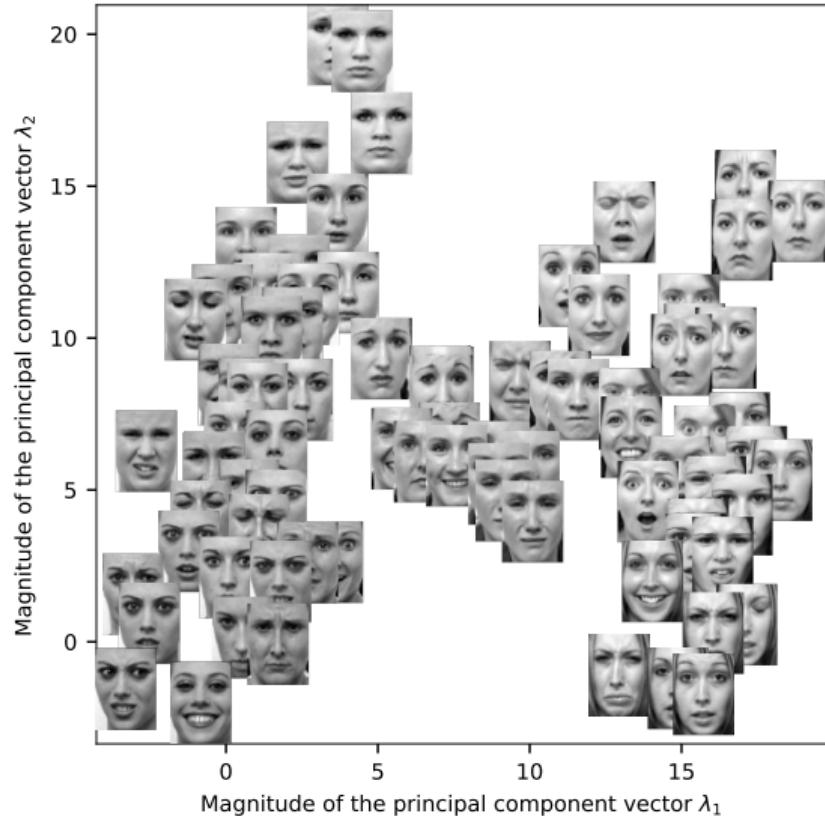
Identity Basis



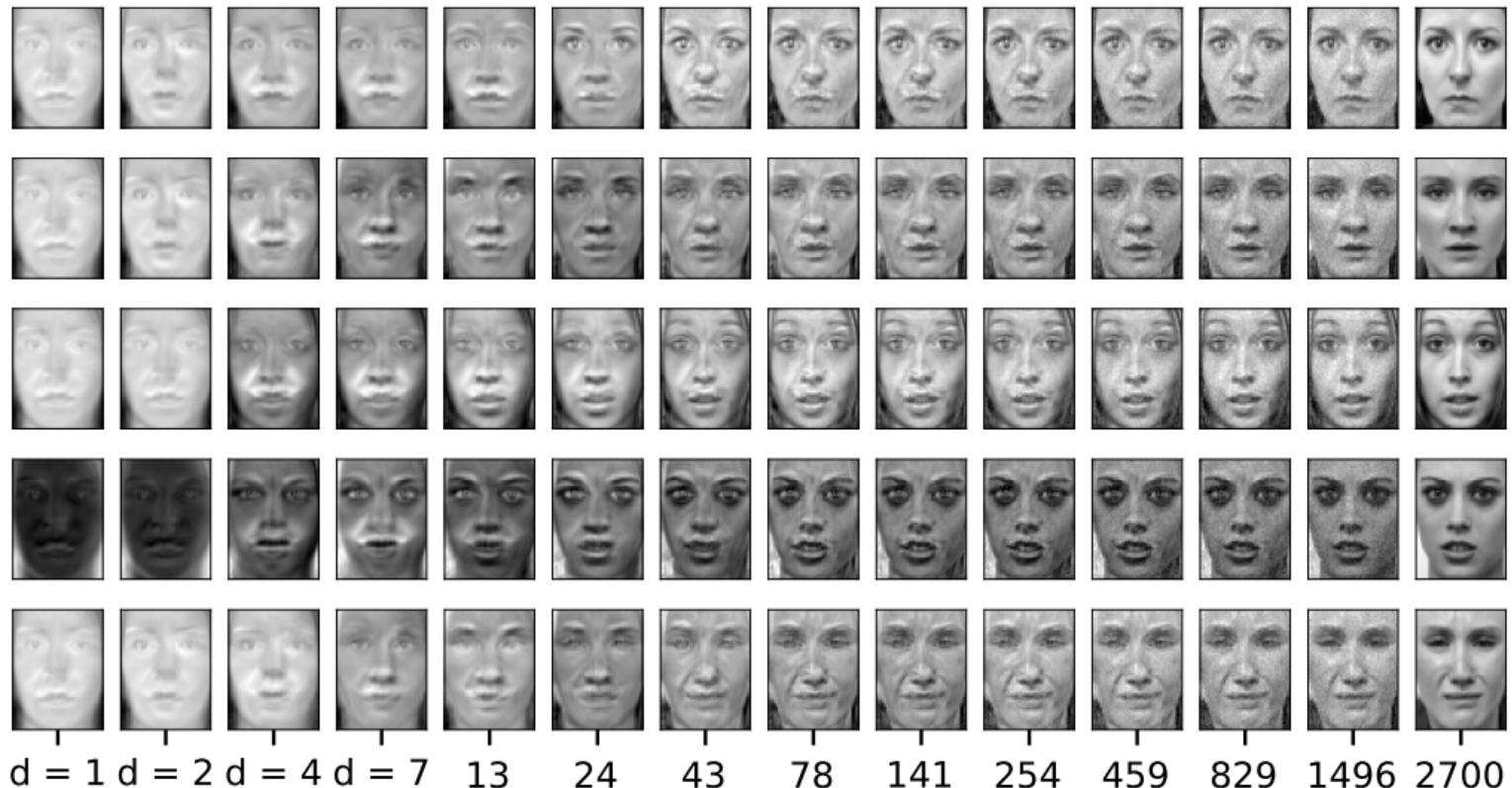
Principal Components



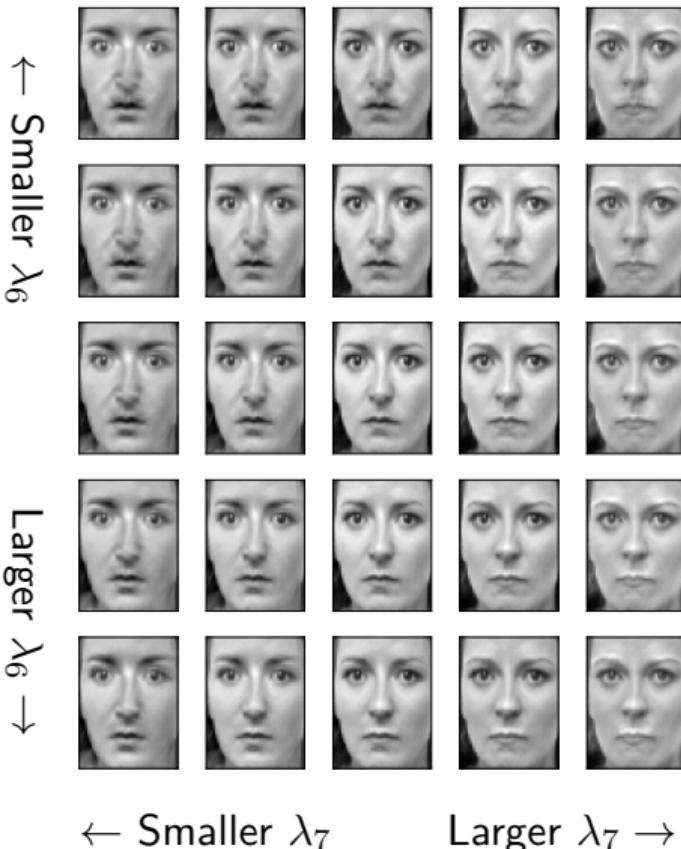
PCA Example: Face Spaces



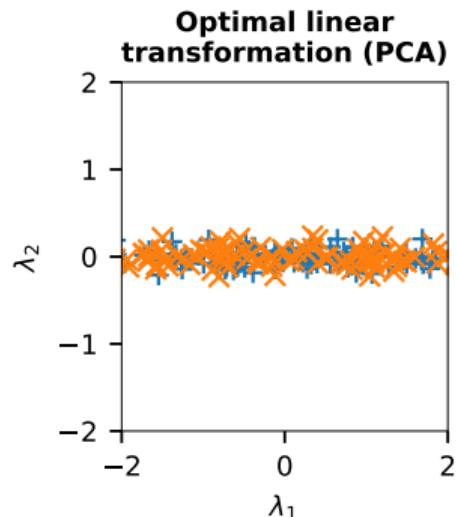
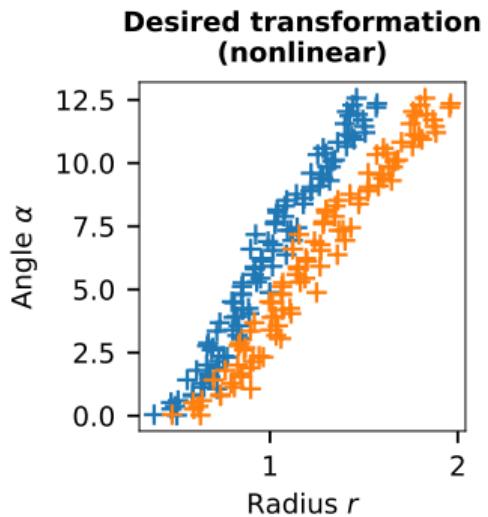
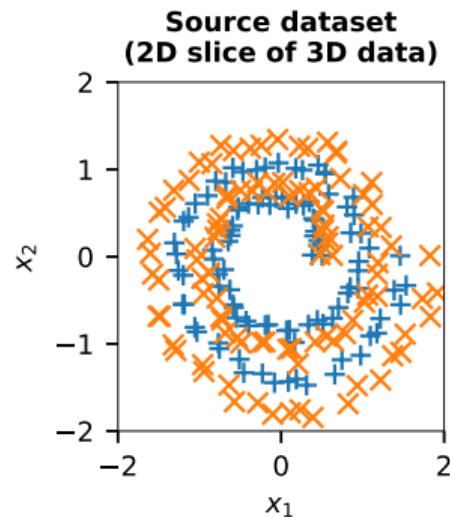
PCA Example: Sparse Vectors



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

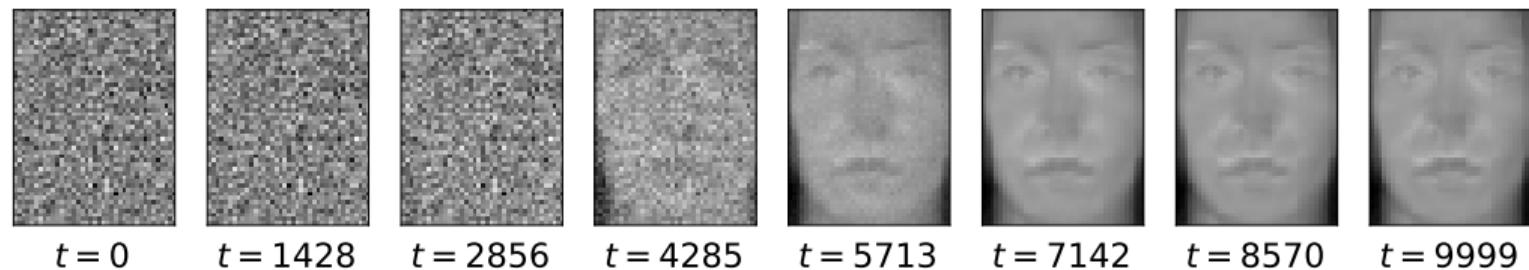


Hebbian Learning

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.

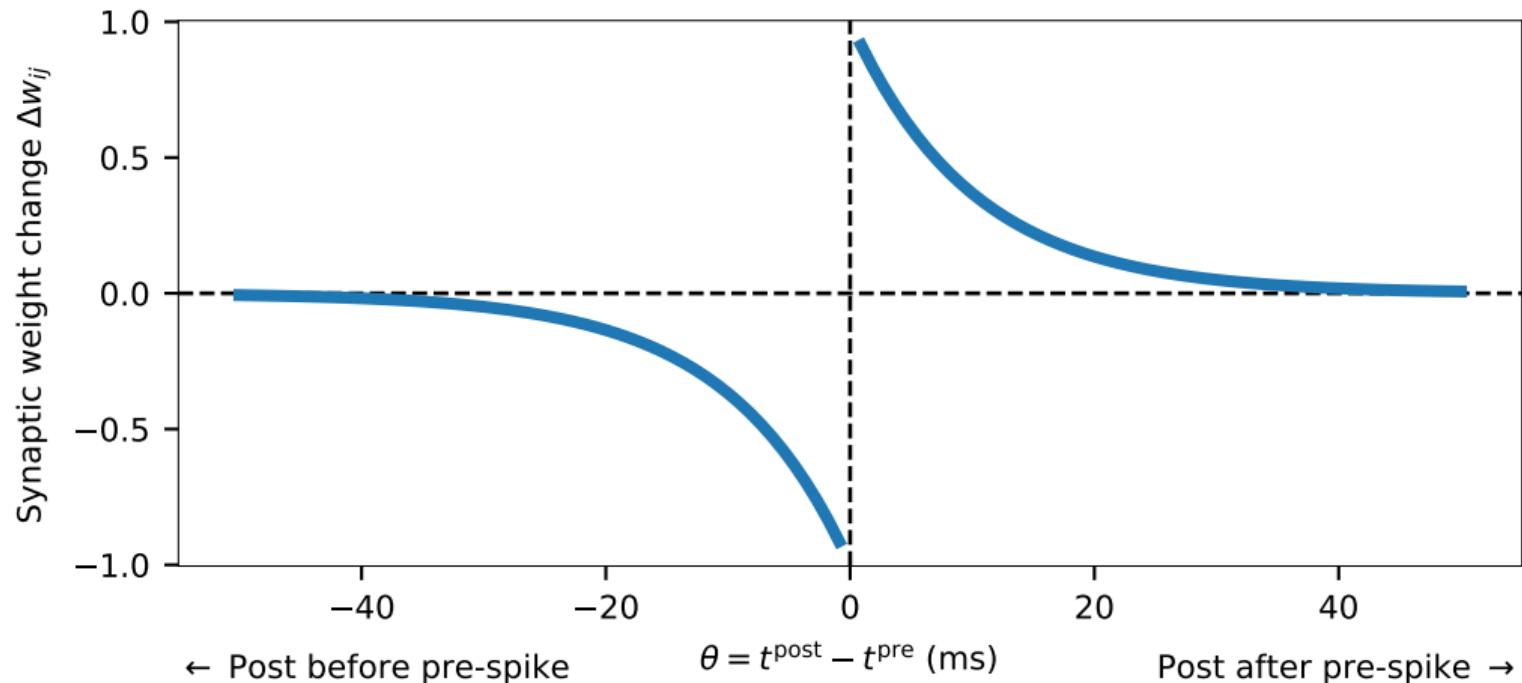
— Donald O. Hebb, “The Organization of Behaviour”, 1949

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:* $f(\mathbf{x}_k; \mathbf{w}) \approx f_{\text{GT}}(\mathbf{x}_k)$
- ▶ 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.