CS 461: Machine Learning
Lecture 4

Dr. Kiri Wagstaff
kiri.wagstaff@calstatela.edu

Plan for Today

- Solution to HW 2
- Support Vector Machines
  - Review of Classification
  - Non-separable data: the Kernel Trick
  - Regression
- Neural Networks
  - Perceptrons
  - Multilayer Perceptrons
  - Backpropagation

Review from Lecture 3

- Decision trees
  - Regression trees, pruning
- Evaluation
  - One classifier:
    - Errors, confidence intervals, significance
    - Baselines
  - Comparing two classifiers: McNemar's test
  - Cross-validation
- Support Vector Machines
  - Classification
    - Linear discriminants, maximum margin
    - Learning (optimization): gradient descent, QP
    - Non-separable classes

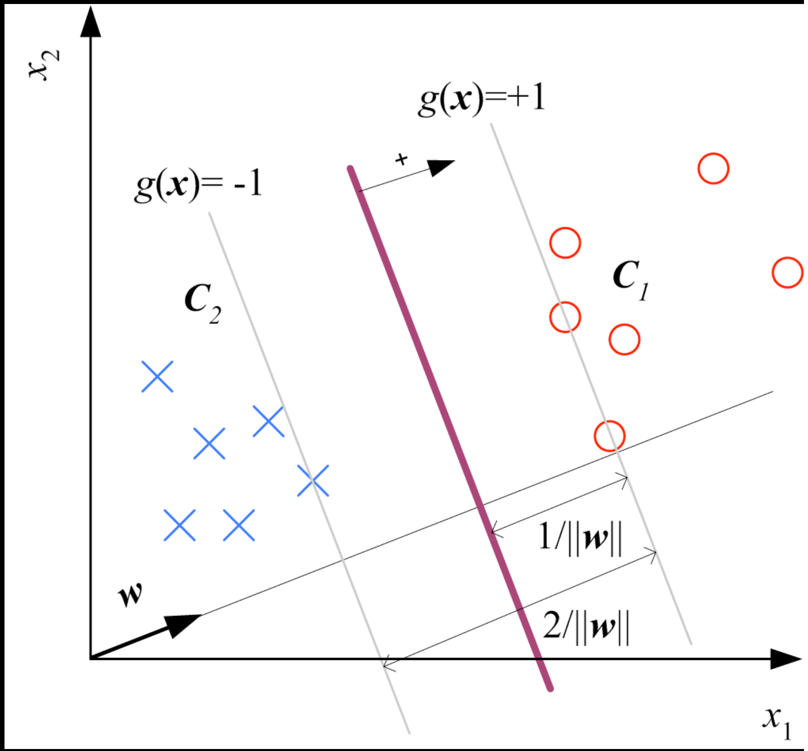
Support Vector Machines

Chapter 10

Support Vector Machines

- Maximum-margin linear classifiers

$$g(\mathbf{x} \mid \mathbf{w}, w_0) = \mathbf{w}^T\mathbf{x} + w_0 = \sum_{j=1}^{d} w_j x_j + w_0$$



  - Support vectors

- How to find best w, b?

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y^t\left(\mathbf{w}^T\mathbf{x}^t + b\right) \geq +1, \forall t$$

Optimization in action: SVM applet

What if Data isn't Linearly Separable?

1. Add "slack" variables to permit some errors
   - [Andrew Moore's slides]
2. Embed data in higher-dimensional space
   - Explicit: Basis functions (new features)
   - Implicit: Kernel functions (new dot product)
   - Still need to find a linear hyperplane

Build a Better Dot Product: Basis Functions

- Preprocess input ***x*** by basis functions

$$\mathbf{z} = \boldsymbol{\varphi}(\mathbf{x}) \qquad g(\mathbf{z}) = \mathbf{w}^T\mathbf{z}$$

$$g(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x})$$

$$\begin{aligned}
g(\mathbf{x}) &= \mathbf{w}^T x + b \\
&= \sum_t \alpha^t y^t \varphi(\mathbf{x}^t)^T \varphi(\mathbf{x}) + b \\
&= \sum_t \alpha^t y^t K(\mathbf{x}^t, \mathbf{x}) + b
\end{aligned}$$

Build a Better Dot Product: Basis Functions -> Kernel Functions

$$g(\mathbf{x}) = \mathbf{w}^T x + b$$
$$= \sum_t \alpha^t y^t \varphi(\mathbf{x}^t)^T \varphi(\mathbf{x}) + b$$
$$= \sum_t \alpha^t y^t K(\mathbf{x}^t, \mathbf{x}) + b$$

- Linear $K(\mathbf{x}^t, \mathbf{x}) = (\mathbf{x}^T \mathbf{x}^t)$
- Polynomial

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T \boldsymbol{y} + 1)^2$$
$$= (x_1 y_1 + x_2 y_2 + 1)^2$$
$$= 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2$$
$$\phi(\boldsymbol{x}) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2\right]^T$$

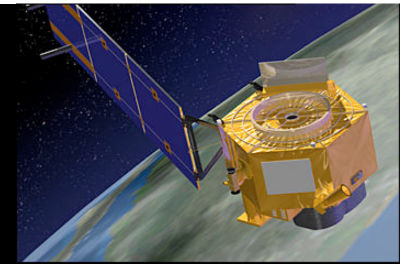
- RBF

$$K(\boldsymbol{x}^t, \boldsymbol{x}) = \exp\left[-\frac{\|\boldsymbol{x}^t - \boldsymbol{x}\|^2}{\sigma^2}\right]$$

Optimization in action: [SVM applet]

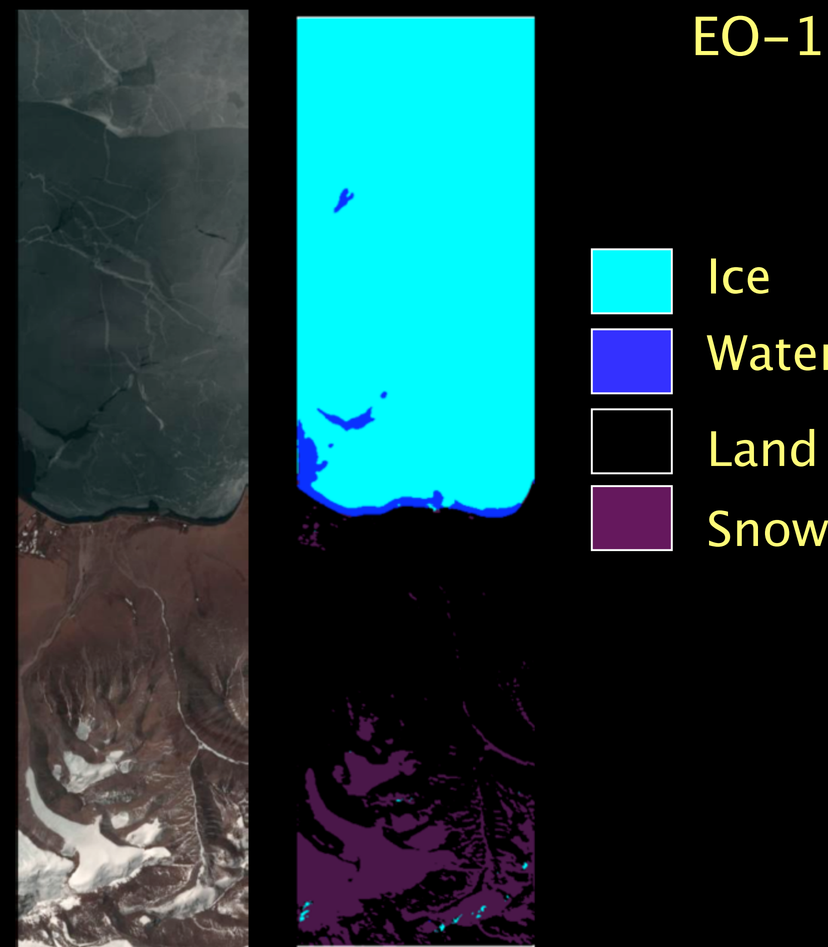
- Sigmoid $K(\boldsymbol{x}^t, \boldsymbol{x}) = \tanh(2\boldsymbol{x}^T \boldsymbol{x}^t + 1)$

Example: Orbital Classification



EO-1

- Linear SVM flying on EO-1 Earth Orbiter since Dec. 2004
  - Classify every pixel
  - Four classes
  - 12 features (of 256 collected)



Hyperion Classified

[Castano et al., 2005]

SVM Regression

- Use a linear model (possibly kernelized)

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

- Use the ϵ-insensitive error function

$$e_\varepsilon\left(g^t, f\left(\mathbf{x}^t\right)\right) = \begin{cases} 0 & \text{if } \left|y^t - g\left(\mathbf{x}^t\right)\right| < \varepsilon \\ \left|y^t - g\left(\mathbf{x}^t\right)\right| - \varepsilon & \text{otherwise} \end{cases}$$

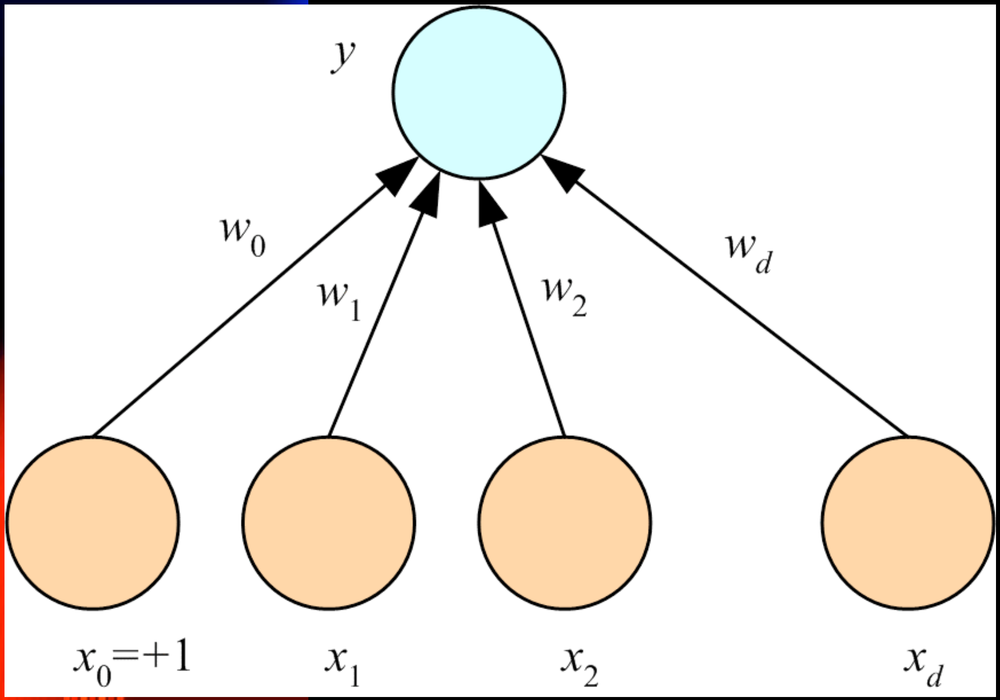
- $$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_t \left(\xi_+^t + \xi_-^t\right)$$

Neural Networks

Chapter 11

Perceptron

Graphical



Math

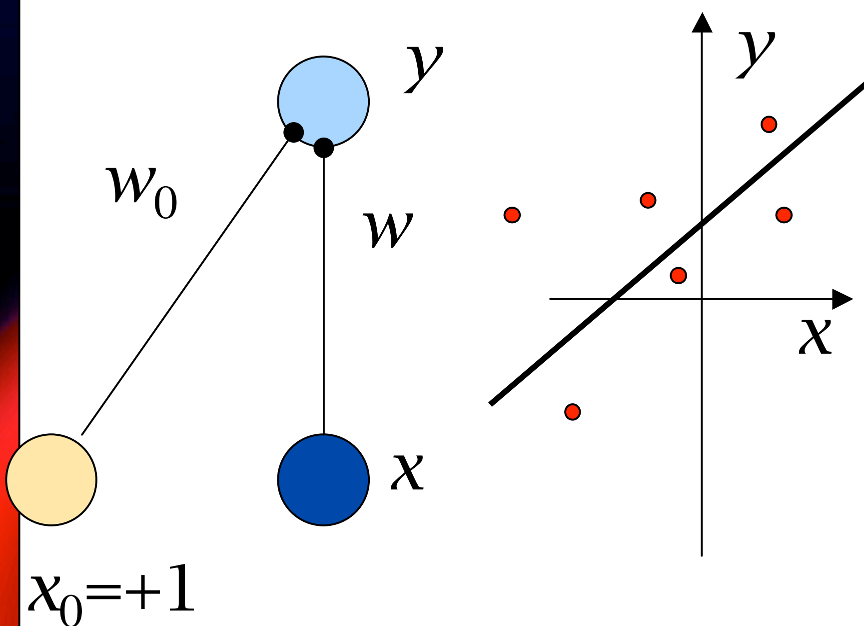
$$y = \sum_{j=1}^{d} w_j x_j + w_0 = \boldsymbol{w}^T \boldsymbol{x}$$

$$\boldsymbol{w} = \left[w_0, w_1, \ldots, w_d\right]^T$$

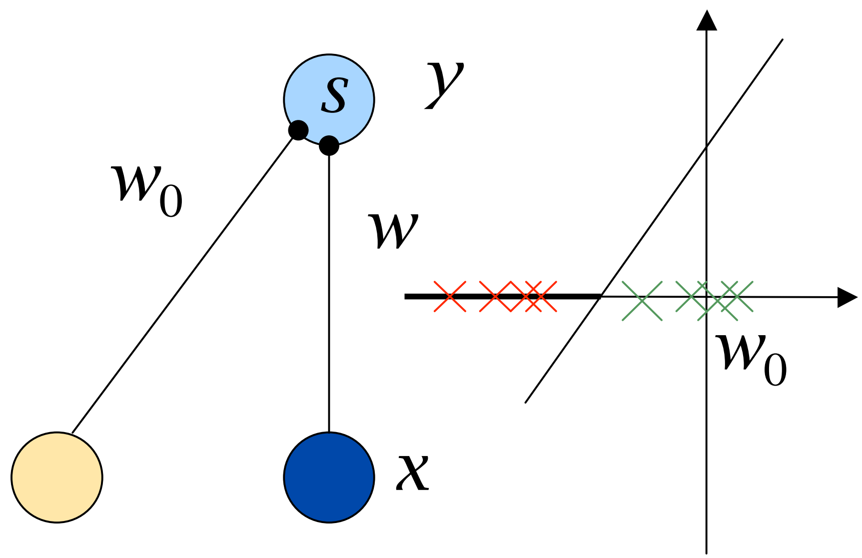
$$\boldsymbol{x} = \left[1, x_1, \ldots, x_d\right]^T$$

Perceptrons in action

- Regression: $y = wx + w_0$



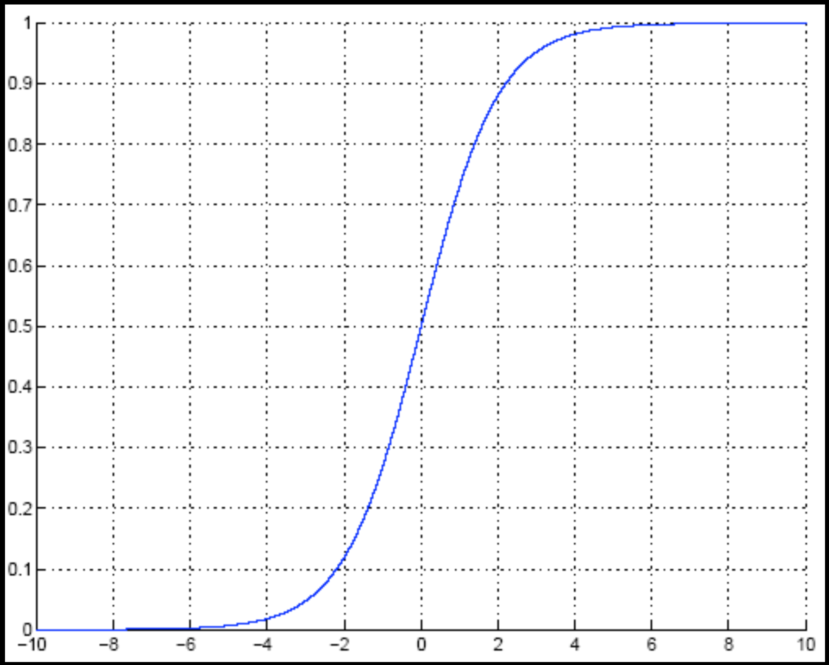
- Classification: $y = 1(wx + w_0 > 0)$



“Smooth” Output: Sigmoid Function

1. Calculate $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ and choose C_1 if $g(\mathbf{x}) > 0$, or

2. Calculate $y = \text{sigmoid}(\mathbf{w}^T\mathbf{x})$ and choose C_1 if $y > 0.5$

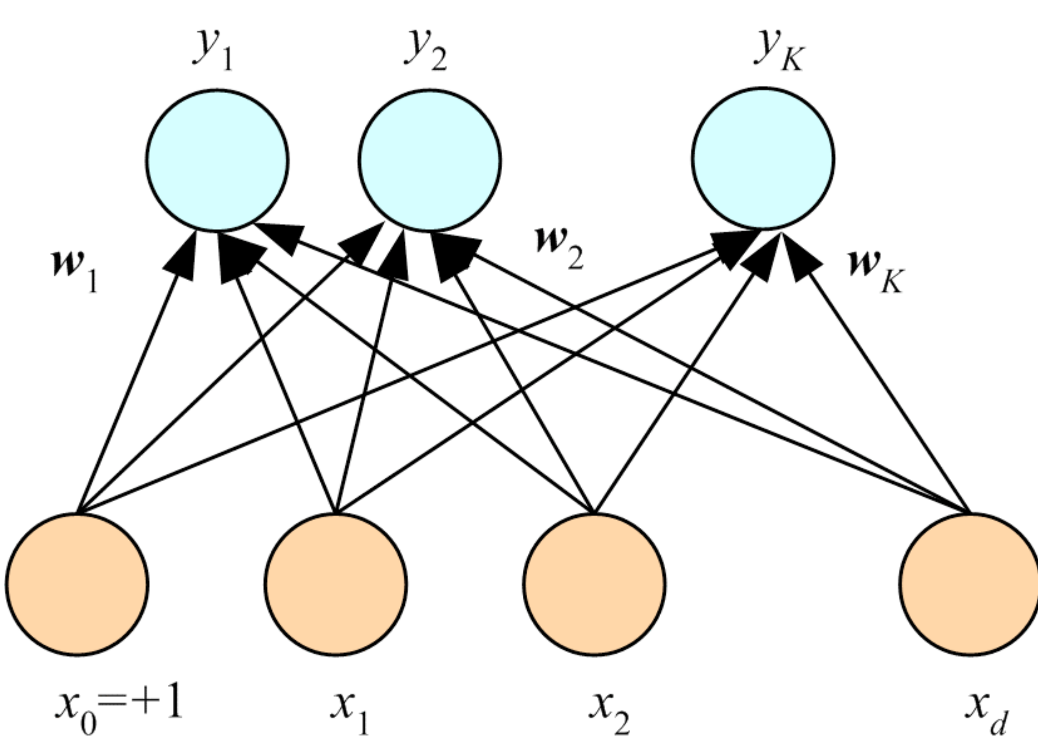


Why?

- Converts output to probability!
- Less “brittle” boundary

$$y = \text{sigmoid}(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp[-\mathbf{w}^T\mathbf{x}]}$$

K outputs



Regression:

$$y_i = \sum_{j=1}^{d} w_{ij} x_j + w_{i0} = \boldsymbol{w}_i^T \boldsymbol{x}$$

$$\boldsymbol{y} = \mathbf{W}\boldsymbol{x}$$

Classification:

$$o_i = \boldsymbol{w}_i^T \boldsymbol{x}$$

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

$$\text{choose } C_i$$

$$\text{if } y_i = \max_k y_k$$

Softmax

Training a Neural Network

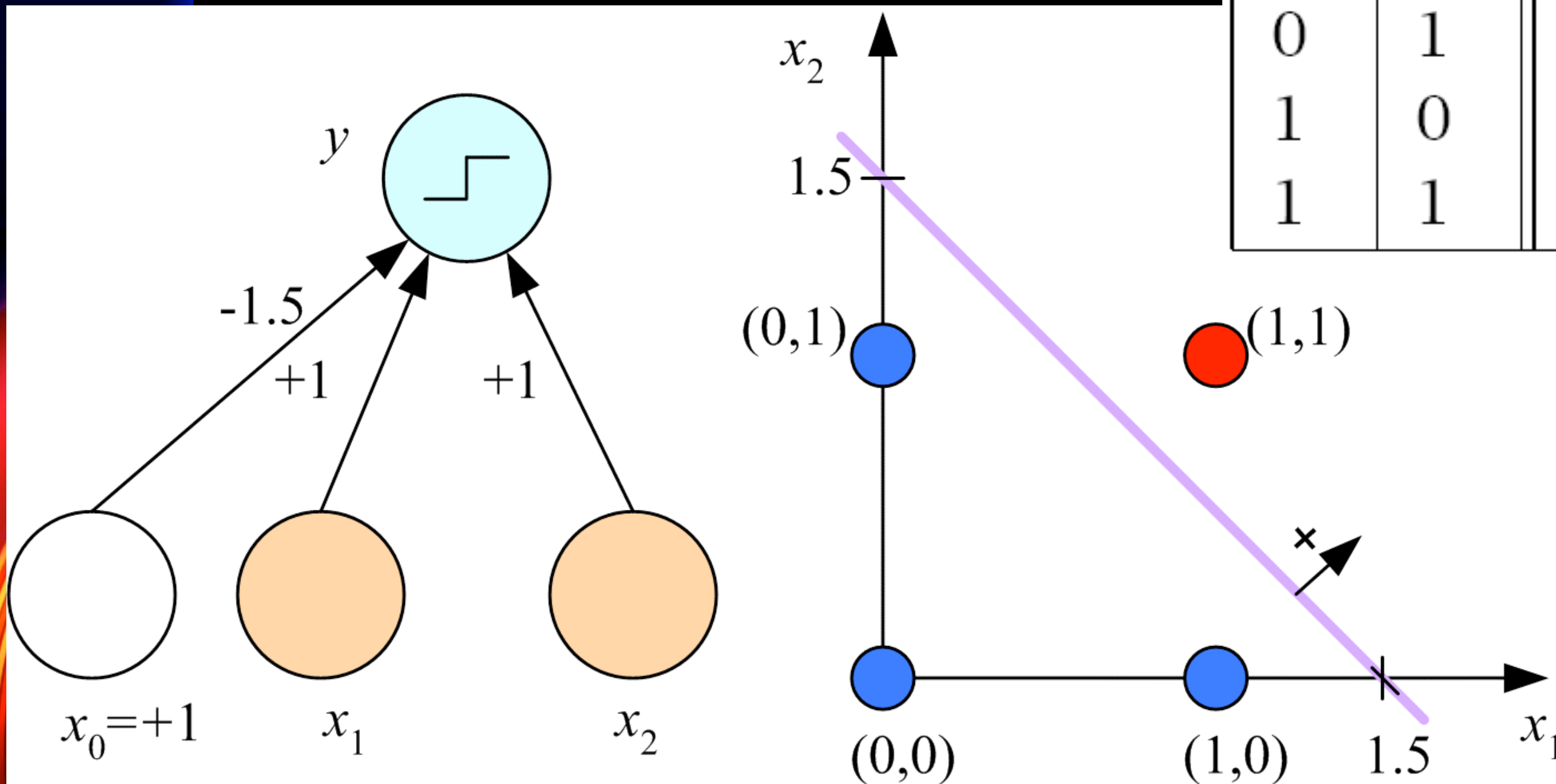
1. Randomly initialize weights
2. Update =
   Learning rate * (Desired - Actual) * Input

$$\Delta w_j^t = \eta\left(y^t - \hat{y}^t\right)x_j^t$$

Learning Boolean AND

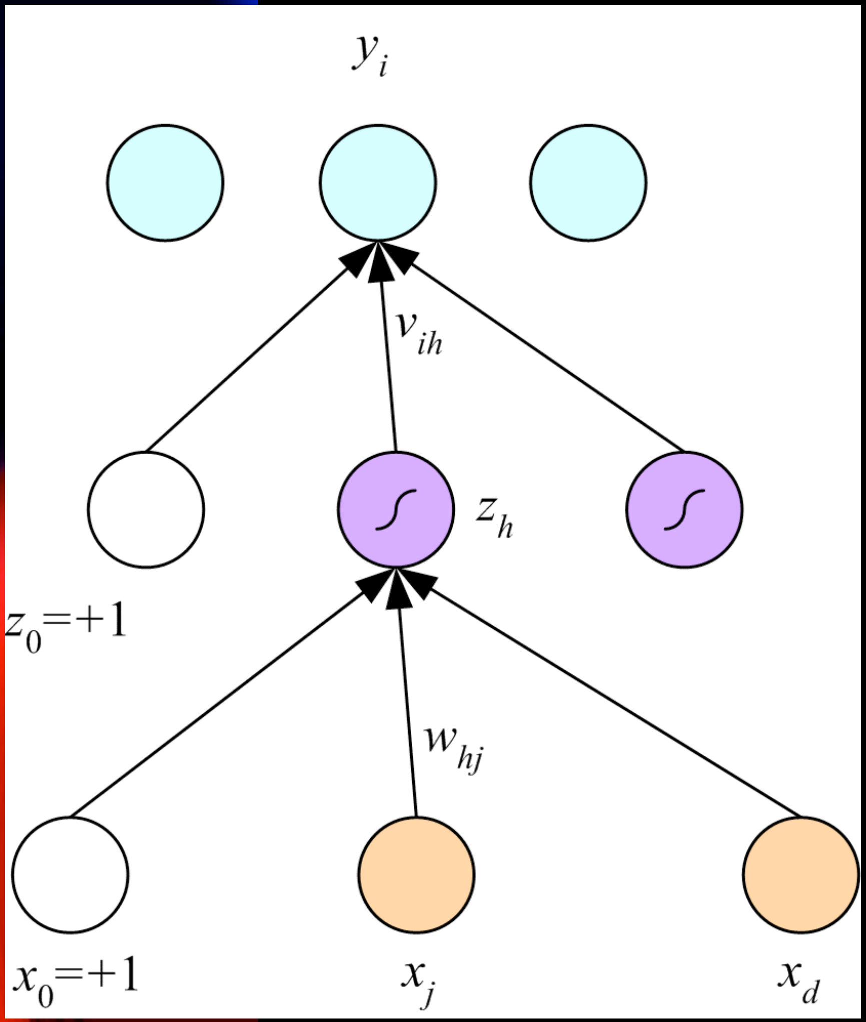
$$\Delta w_j^t = \eta\left(y^t - \hat{y}^t\right)x_j^t$$

x_1	x_2	r
0	0	0
0	1	0
1	0	0
1	1	1



Perceptron demo

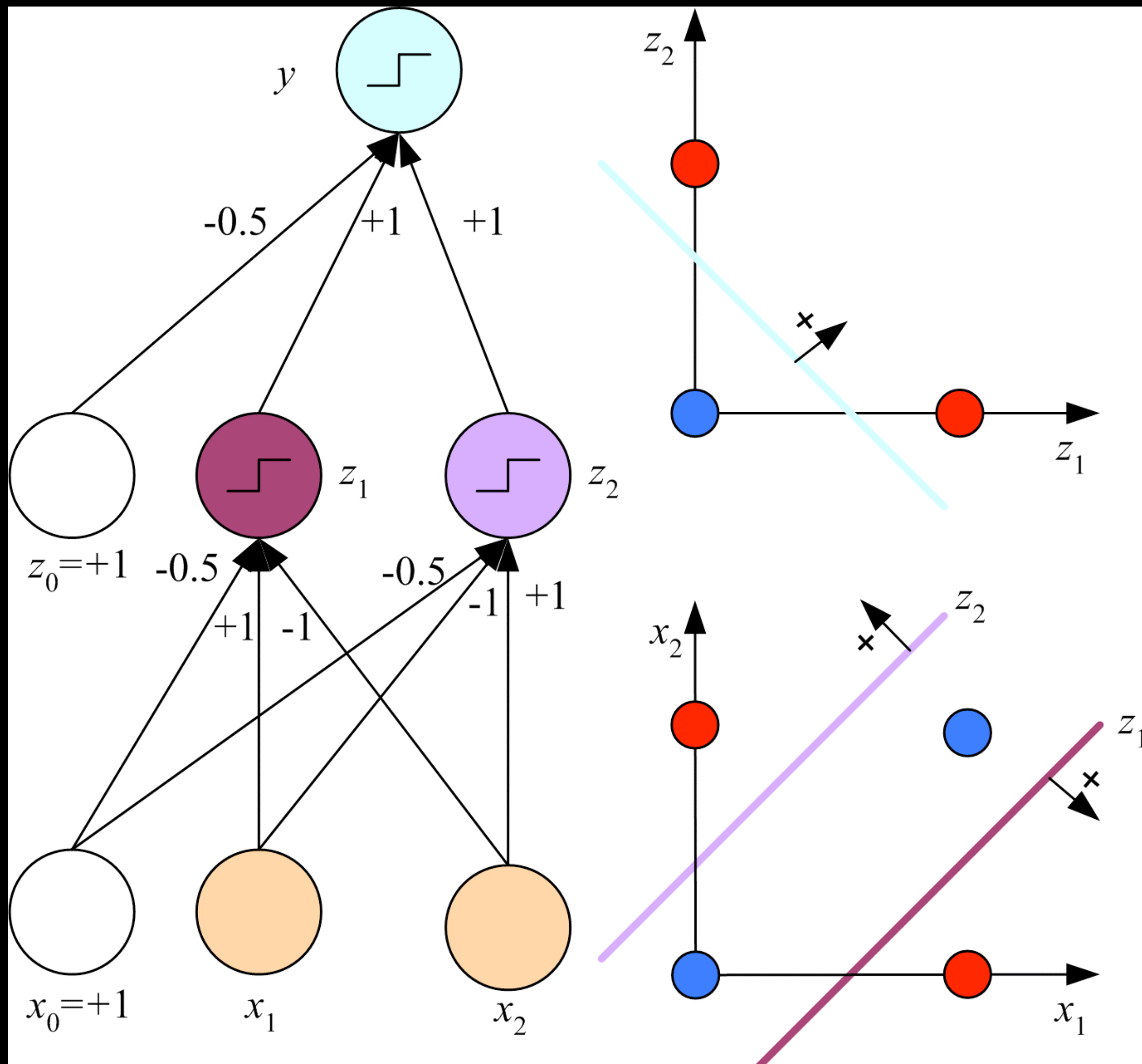
Multilayer Perceptrons = MLP = ANN



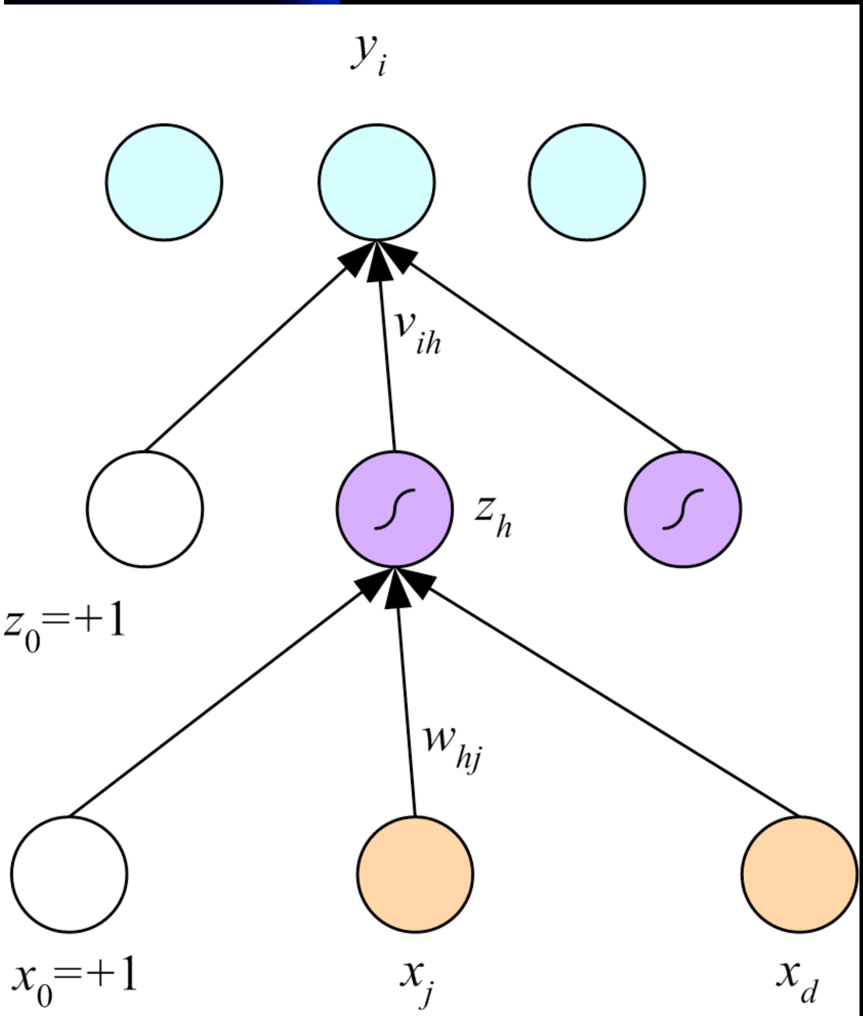
$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \operatorname{sigmoid}\left(\mathbf{w}_h^T \mathbf{x}\right) = \frac{1}{1 + \exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

x_1 XOR x_2 = (x_1 AND ~x_2) OR (~x_1 AND x_2)



Backpropagation: MLP training



$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}\left(\mathbf{w}_h^T \mathbf{x}\right)$$

$$= \frac{1}{1 + \exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

$$\boxed{\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial z_h} \frac{\partial z_h}{\partial w_{hj}}}$$

Backpropagation: Regression

$$E(\mathbf{W},\mathbf{v}\mid\mathcal{X}) = \frac{1}{2}\sum_t \left(y^t - \hat{y}^t\right)^2$$

$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_0$$

$$\Delta v_h = \eta \sum_t \left(y^t - \hat{y}^t\right) z_h^t$$

Backward

Forward

$$z_h = \boxed{\text{sigmoid}}\left(\boldsymbol{w}_h^T \boldsymbol{x}\right)$$

$$\begin{aligned}
\Delta w_{hj} &= -\eta \frac{\partial E}{\partial w_{hj}} \\
&= -\eta \sum_t \frac{\partial E}{\partial \hat{y}^t} \frac{\partial \hat{y}^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}} \\
&= -\eta \sum_t -\left(y^t - \hat{y}^t\right) v_h \boxed{z_h^t \left(1 - z_h^t\right)} x_j^t \\
&= \eta \sum_t \left(y^t - \hat{y}^t\right) v_h z_h^t \left(1 - z_h^t\right) x_j^t
\end{aligned}$$

$\boldsymbol{x}$

Backpropagation: Classification

$$E(\mathbf{W},\mathbf{v} \mid \mathcal{X}) = -\sum_t y^t \log \hat{y}^t + \left(1-y^t\right)\log\left(1-\hat{y}^t\right)$$

$$y^t = \text{sigmoid}\left(\sum_{h=1}^{H} v_h z_h^t + v_0\right)$$

Forward

$$z_h = \boxed{\text{sigmoid}}\left(\boldsymbol{w}_h^T \boldsymbol{x}\right)$$

$$\boldsymbol{x}$$

$$\Delta v_h = \eta \sum_t \left(y^t - \hat{y}^t\right) z_h^t$$

Backward

$$\begin{aligned}
\Delta w_{hj} &= -\eta \frac{\partial E}{\partial w_{hj}} \\
&= -\eta \sum_t \frac{\partial E}{\partial \hat{y}^t} \frac{\partial \hat{y}^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}} \\
&= -\eta \sum_t -\left(y^t - \hat{y}^t\right) v_h \boxed{z_h^t\left(1-z_h^t\right)} x_j^t \\
&= \eta \sum_t \left(y^t - \hat{y}^t\right) v_h z_h^t \left(1-z_h^t\right) x_j^t
\end{aligned}$$

Backpropagation Algorithm

Initialize all v_{ih} and w_{hj} to rand$(-0.01, 0.01)$

Repeat

For all $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$ in random order

For $h = 1, \ldots, H$

$z_h \leftarrow \mathrm{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}^t)$

For $i = 1, \ldots, K$

$y_i = \boldsymbol{v}_i^T \boldsymbol{z}$

For $i = 1, \ldots, K$

$\Delta \boldsymbol{v}_i = \eta (r_i^t - y_i^t) \boldsymbol{z}$

For $h = 1, \ldots, H$

$\Delta \boldsymbol{w}_h = \eta (\sum_i (r_i^t - y_i^t) v_{ih}) z_h (1 - z_h) \boldsymbol{x}^t$

For $i = 1, \ldots, K$

$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i$

For $h = 1, \ldots, H$

$\boldsymbol{w}_h \leftarrow \boldsymbol{w}_h + \Delta \boldsymbol{w}_h$

Until convergence

Examples

- Digit Recognition
- Ball Balancing

ANN vs. SVM

- SVM with sigmoid kernel = 2-layer MLP
- Parameters
  - ANN: # hidden layers, # nodes
  - SVM: kernel, kernel params, C
- Optimization
  - ANN: local minimum (gradient descent)
  - SVM: global minimum (QP)
- Interpretability? About the same…
- So why SVMs?
  - Sparse solution, geometric interpretation, less likely to overfit data

Summary: Key Points for Today

- Support Vector Machines
  - Non-separable data: basis functions, the Kernel Trick
  - Regression
- Neural Networks
  - Perceptrons
    - Sigmoid
    - Training by gradient descent
  - Multilayer Perceptrons
    - Backpropagation (gradient descent)
- ANN vs. SVM

Review for Midterm

Next Time

- Midterm Exam!
  - 9:00 - 10:30 a.m.
  - Open book, open notes (no computer)
  - Covers all material through today
- Bayesian Methods
  (read Ch. 3.1, 3.2, 3.7, 3.9)
  - If you're rusty on probability, read Appendix A
- Questions to answer from the reading
  - Posted on the website (calendar)