COMP0169: Machine Learning for Visual Computing

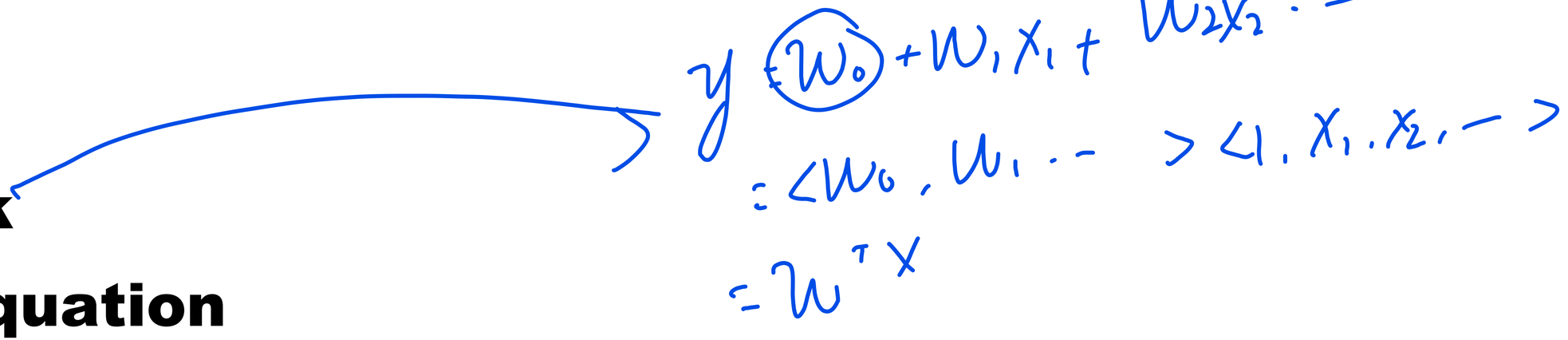
Linear Classifier and MultiLayer Perceptron

Lectures will be Recorded

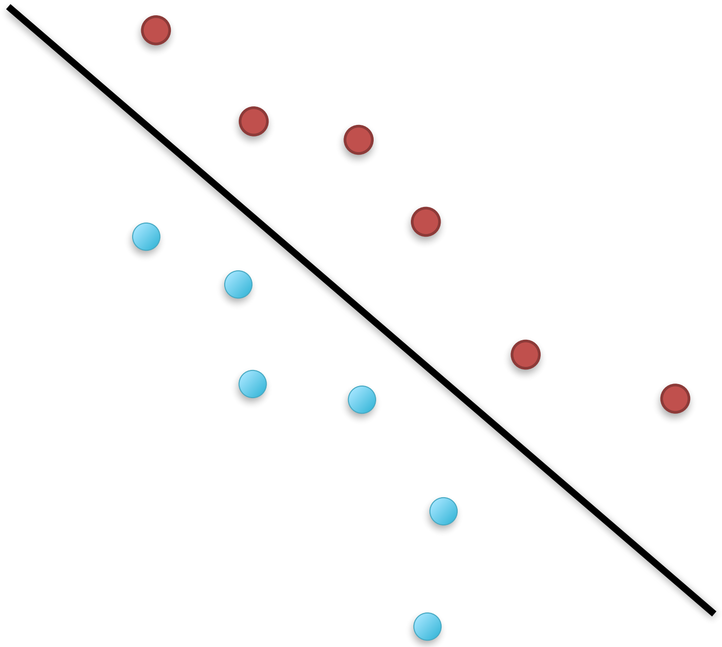
CW1 submission date

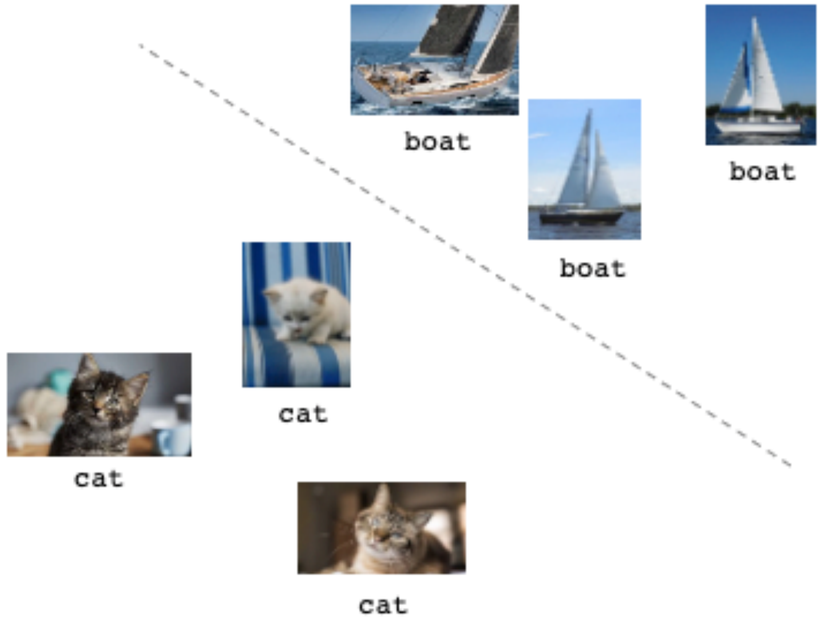
Recap

- **PCA**
- **Bias Trick**
- **Normal Equation**
- **Extension to polynomial fit**

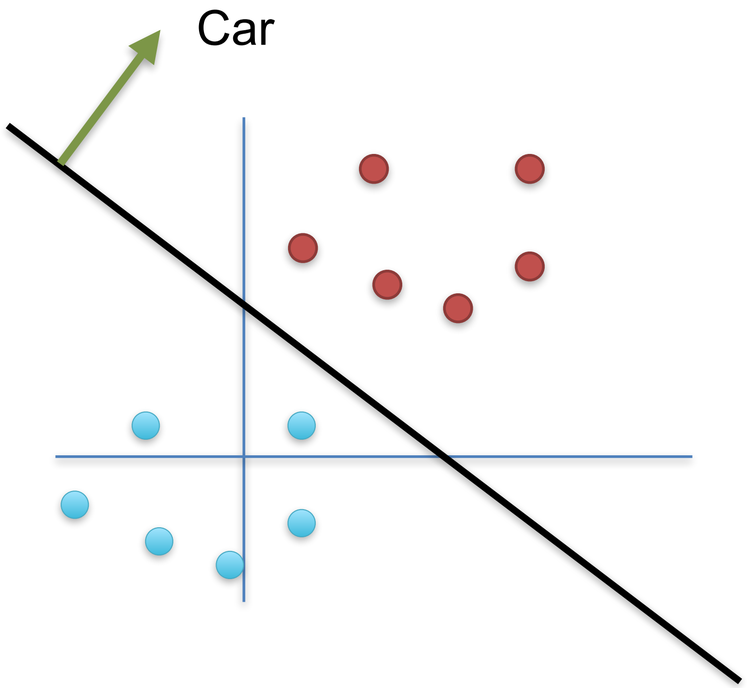


Simple Classifier

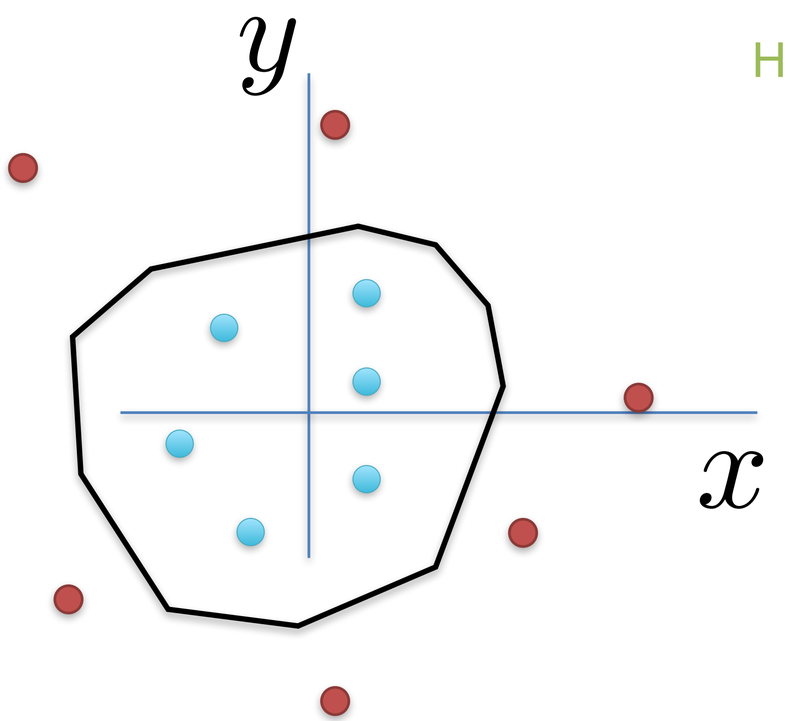




Classifier Geometric Viewpoint



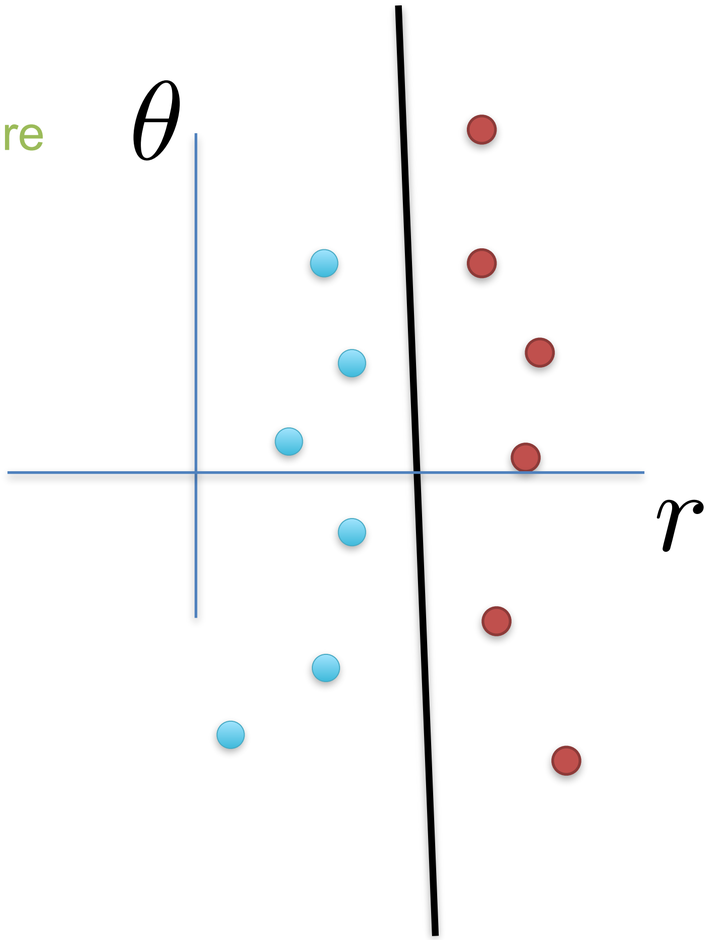
Classifier Geometric Viewpoint



Hand-coded feature

$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1} y/x$$

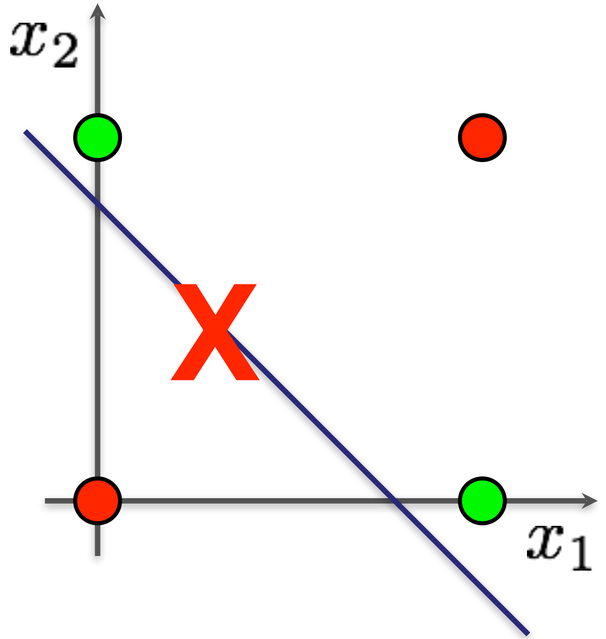


Data-driven/learned separator

XOR Problem

$$y = f(x_1, x_2)$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



← no line can do it.

$$y = f(w_0, w_1, w_2) = \mathcal{H}(w_0 + w_1 x_1 + w_2 x_2) = \mathcal{H}(\mathbf{w}^T \mathbf{x})$$

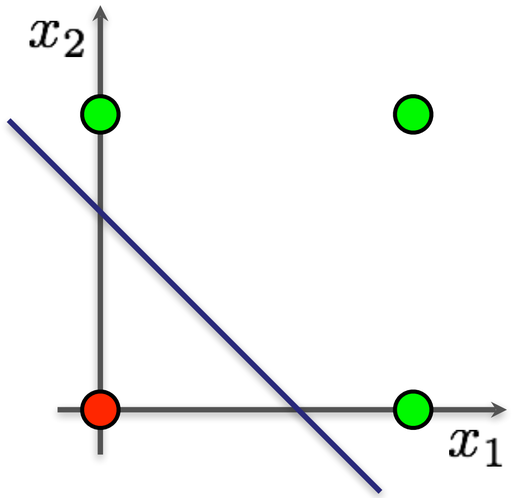
XOR Problem

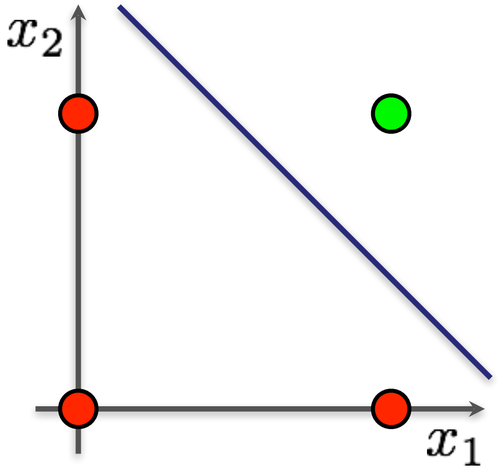
$$y = f(z_1, z_2) = f(g_1(x_1, x_2), g_2(x_1, x_2))$$

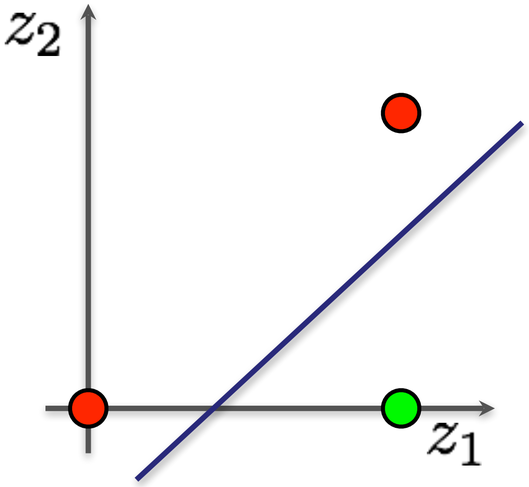
x_1	x_2	z_1
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	z_2
0	0	0
0	1	0
1	0	0
1	1	1

z_1	z_2	y
0	0	0
1	0	1
1	0	1
1	1	0





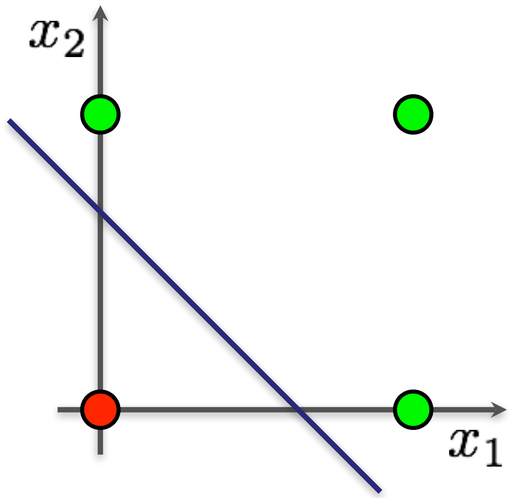


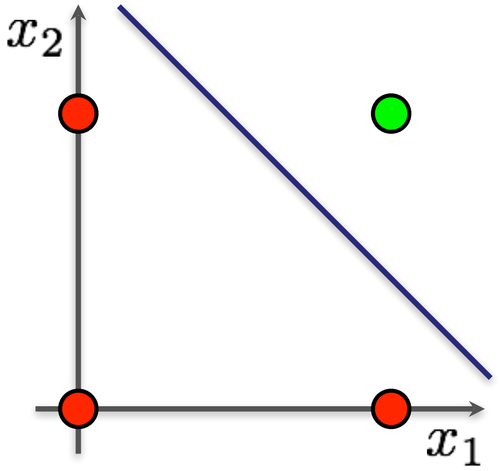
XOR Problem

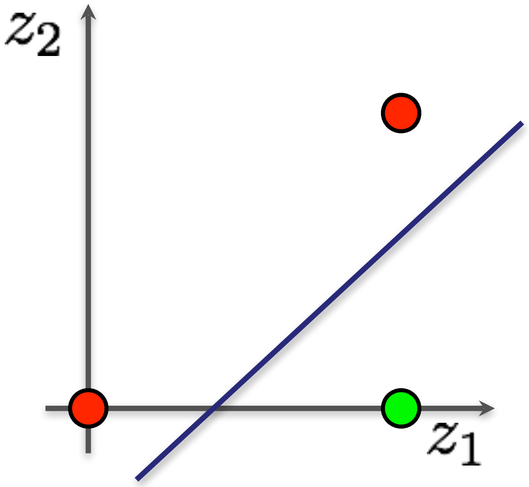
non-linear

$$
\begin{aligned}
y = f(z_1, z_2) &= f(g_1(x_1, x_2), g_2(x_1, x_2)) \\
&= f\left(\mathcal{H}(\mathbf{w}^1, x_1, x_2), \mathcal{H}(\mathbf{w}^2, x_1, x_2)\right) \\
&= \mathcal{H}(\mathbf{w}^3, \mathcal{H}(g_1(\mathbf{w}^1, x_1, x_2)), \mathcal{H}(g_2(\mathbf{w}^2, x_1, x_2)))
\end{aligned}
$$

z_1 z_2

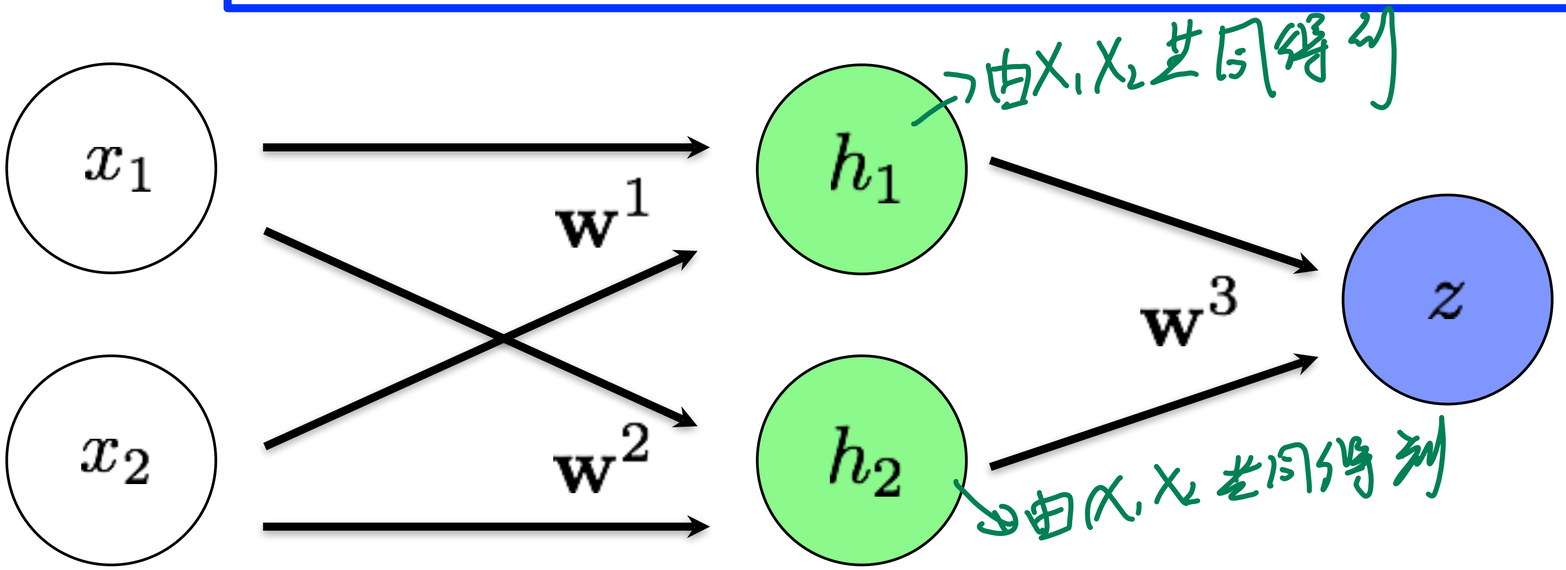




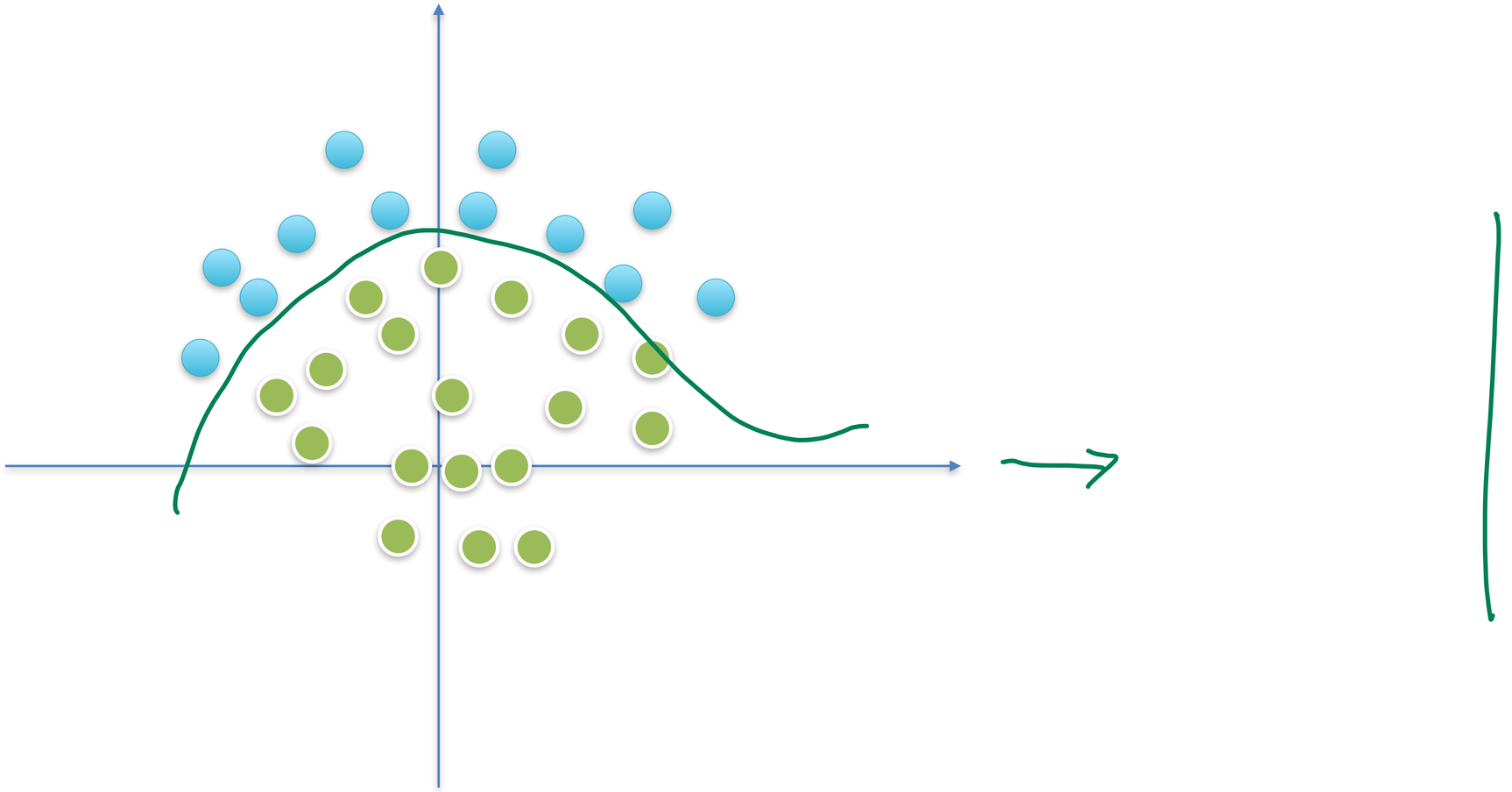


XOR Problem

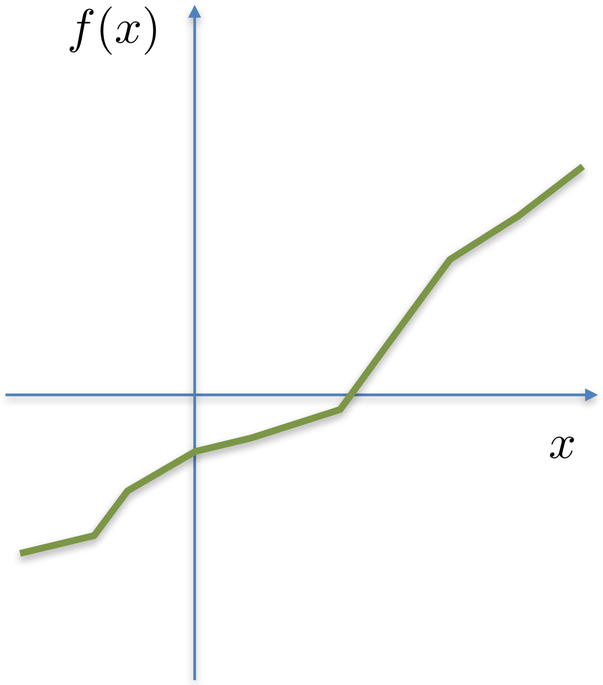
$$
\begin{aligned}
y = f(z_1, z_2) &= f(g_1(x_1, x_2), g_2(x_1, x_2)) \\
&= f\left(\mathcal{H}(\mathbf{w}^1, x_1, x_2), \mathcal{H}(\mathbf{w}^2, x_1, x_2)\right) \\
&= \mathcal{H}(\mathbf{w}^3, \mathcal{H}(g_1(\mathbf{w}^1, x_1, x_2)), \mathcal{H}(g_2(\mathbf{w}^2, x_1, x_2)))
\end{aligned}
$$



What are Neural Networks Doing?

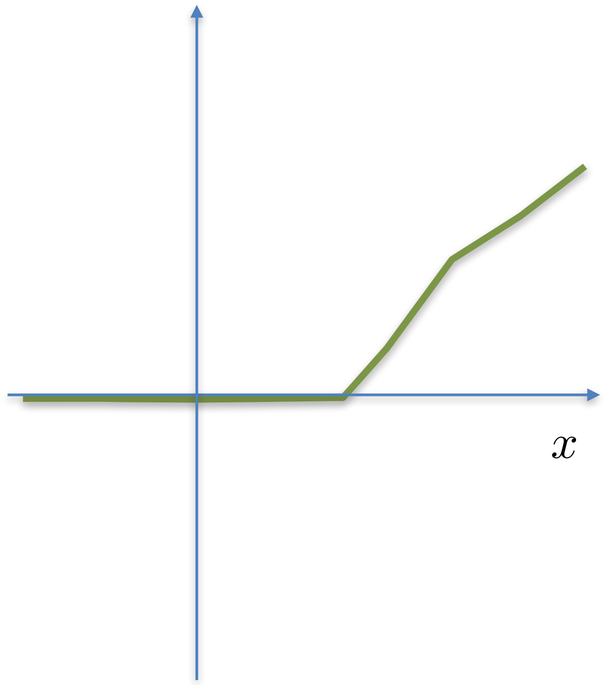


Importance of Non-Linearities



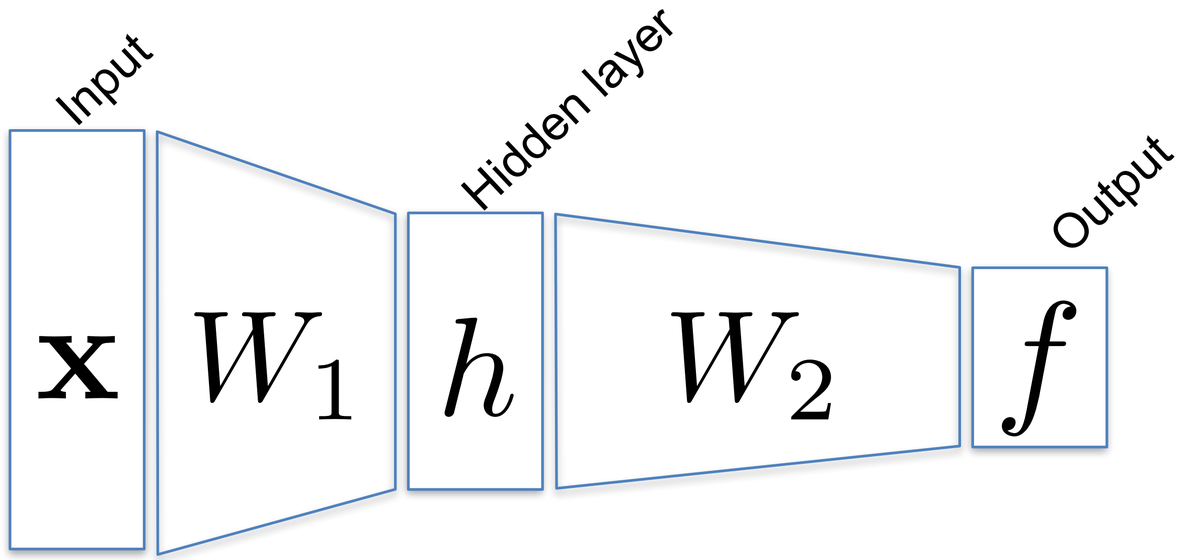
$$g(x) = \max(0, x)$$

$g(f(x)) = \max(0, f(x))$



2-layer Network

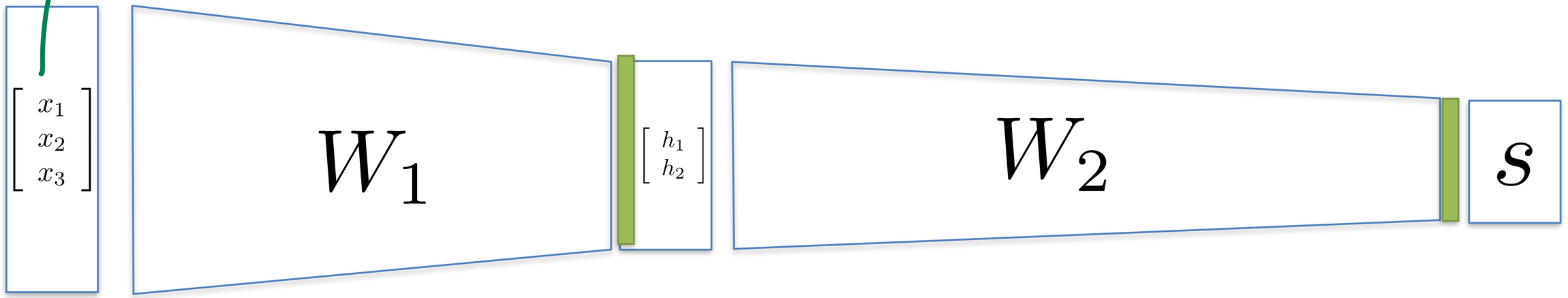
$h \Leftrightarrow$ non-linear applied



Multi-layer perceptron (MLP)

2-layer Network

一个数据点



$$h_1 = w_1^{11}x_1 + w_1^{12}x_2 + w_1^{13}x_3$$
$$h_2 = w_1^{21}x_1 + w_1^{22}x_2 + w_1^{23}x_3$$

$$s = w_2^{11}h_1 + w_2^{12}h_2$$

$$\mathbf{W}_1 = \begin{bmatrix} w_1^{11} & w_1^{12} & w_1^{13} \\ w_1^{21} & w_1^{22} & w_1^{23} \end{bmatrix}$$

$$\mathbf{W}_2 = \begin{bmatrix} w_2^{11} & w_2^{12} \end{bmatrix}$$

Activation Functions

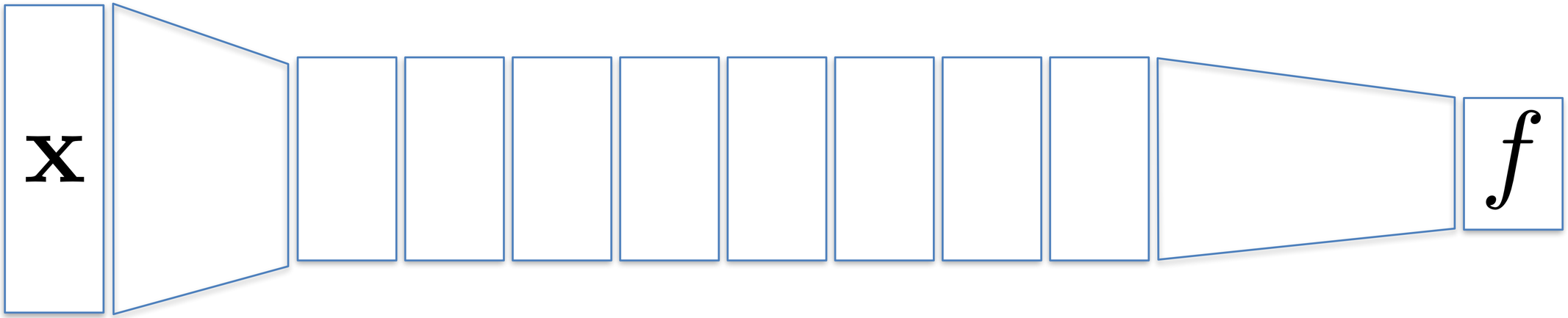
- **ReLU(x)** $\mathrm{RELU}(x) = \max(0, x)$

RELU梯度求导比下

- **Sigmoid** $\sigma(x) = \dfrac{1}{1 + \exp(-x)}$ $\sigma'(x) = \sigma(x)(1 - \sigma(x))$
- **Tanh** $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$
- **Leaky ReLU** $\max(\lambda x, x) \;\; \lambda << 1$
- **ELU**

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Deep Networks



$$f = W_i \max(0, W_{i-1} \max(0, \cdots W_0 \mathbf{x})$$

Loss function and Optimization

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} \left(\mathbf{w}^T \mathbf{x}^i - y^i\right)^2$$

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \qquad \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = 0$$

LS Solution for Linear Regression

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

$$\mathbf{X} = \begin{bmatrix} \frac{(\mathbf{x}^1)^T}{(\mathbf{x}^2)^T} \\ \hline \vdots \\ \hline (\mathbf{x}^N)^T \end{bmatrix}$$

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Gradient-based Optimisation

Initialize θ

$$\min_{\theta} f_\theta(\{\mathbf{x}^{(i)}\}) \qquad \max_{\theta} f_\theta(\{\mathbf{x}^{(i)}\})$$

$$\theta \leftarrow \theta - \alpha \nabla_\theta f_\theta \qquad \theta \leftarrow \theta + \alpha \nabla_\theta f_\theta$$

↗ scalar (how far we go)

gradient

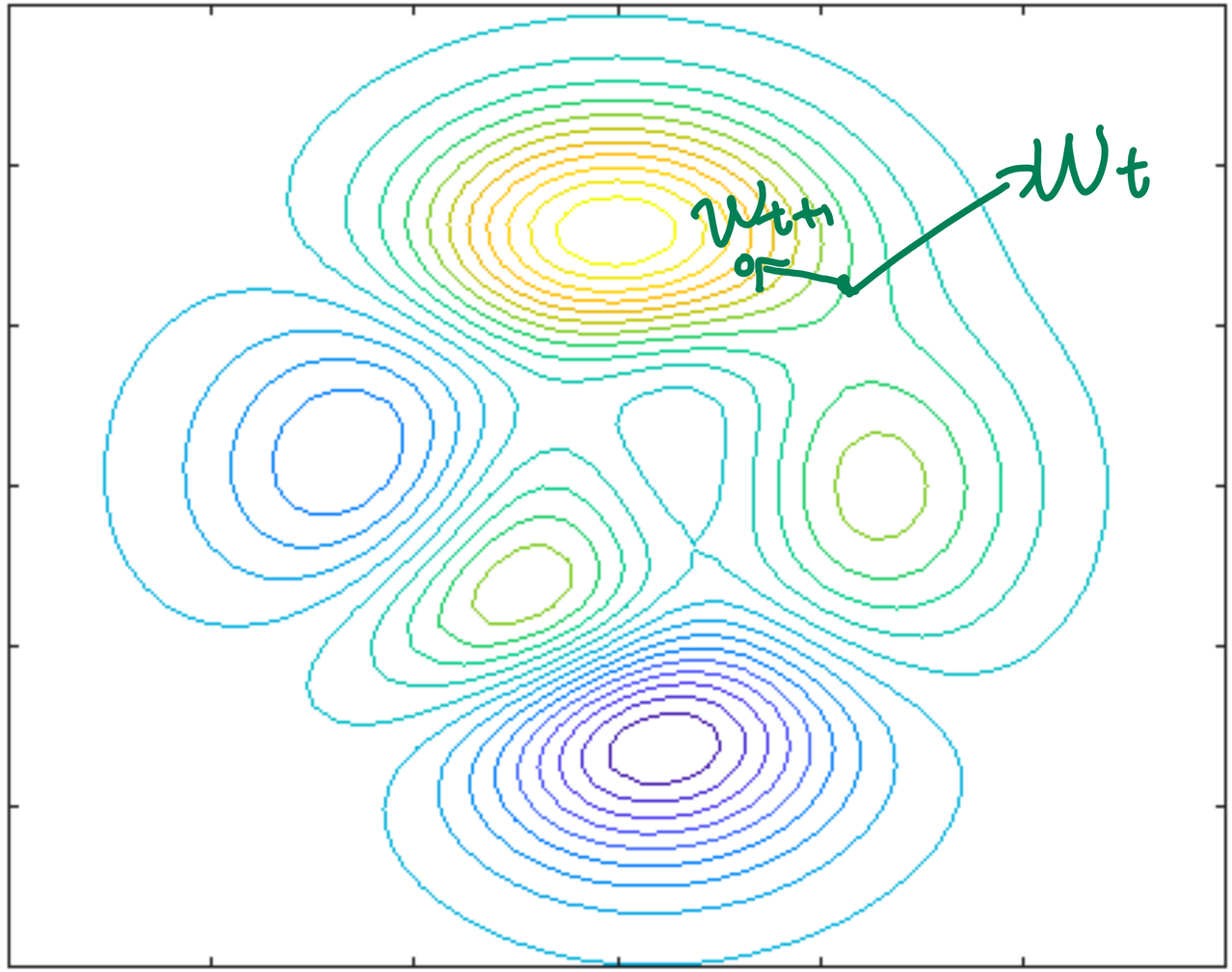
Gradient Descent

more yellow => value ↓

learning rate

$$w_{t+1} = w_t - \alpha \nabla f(w_t)$$

```
w = initialize_weights()
for t in range(num_steps):
  dw = compute_gradient(loss_fn, data, w)
  w -= learning_rate * dw
```



Hyperparameters?
a) intialize_weights
b) learning_rate
c) number of steps/truncation condition

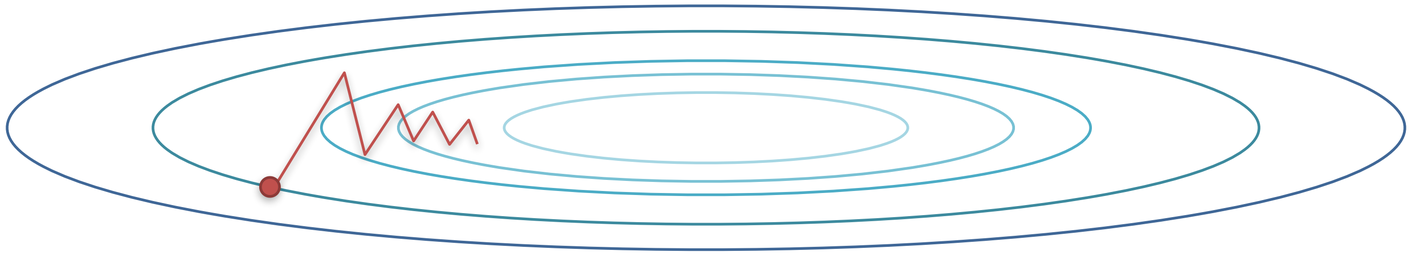
Stochastic Gradient Descent

$$J(\mathbf{w}) = \mathbb{E}_{\mathbf{x},y\sim p_{\text{data}}} L(\mathbf{x}, y, \mathbf{w}) = \frac{1}{N}\sum_i L(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w})$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N}\sum_i \nabla_{\mathbf{w}} L(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w})$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}) \approx \frac{1}{m}\sum_i \nabla_{\mathbf{w}} L(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w})$$

Problems with Gradient Descent



1. Bad condition number → slow convergence

go beyond solution
(一大步跨太大了 orz)

2. Local minima

3. Saddle point

Stochastic/Batch Gradient Descent

$$J(\mathbf{w}) = \mathbb{E}_{\mathbf{x},y\sim p_{\text{data}}} L(\mathbf{x}, y, \mathbf{w}) = \frac{1}{N}\sum_{i}^{N} L(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w})$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N}\sum_{i} \nabla_{\mathbf{w}} L(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w})$$

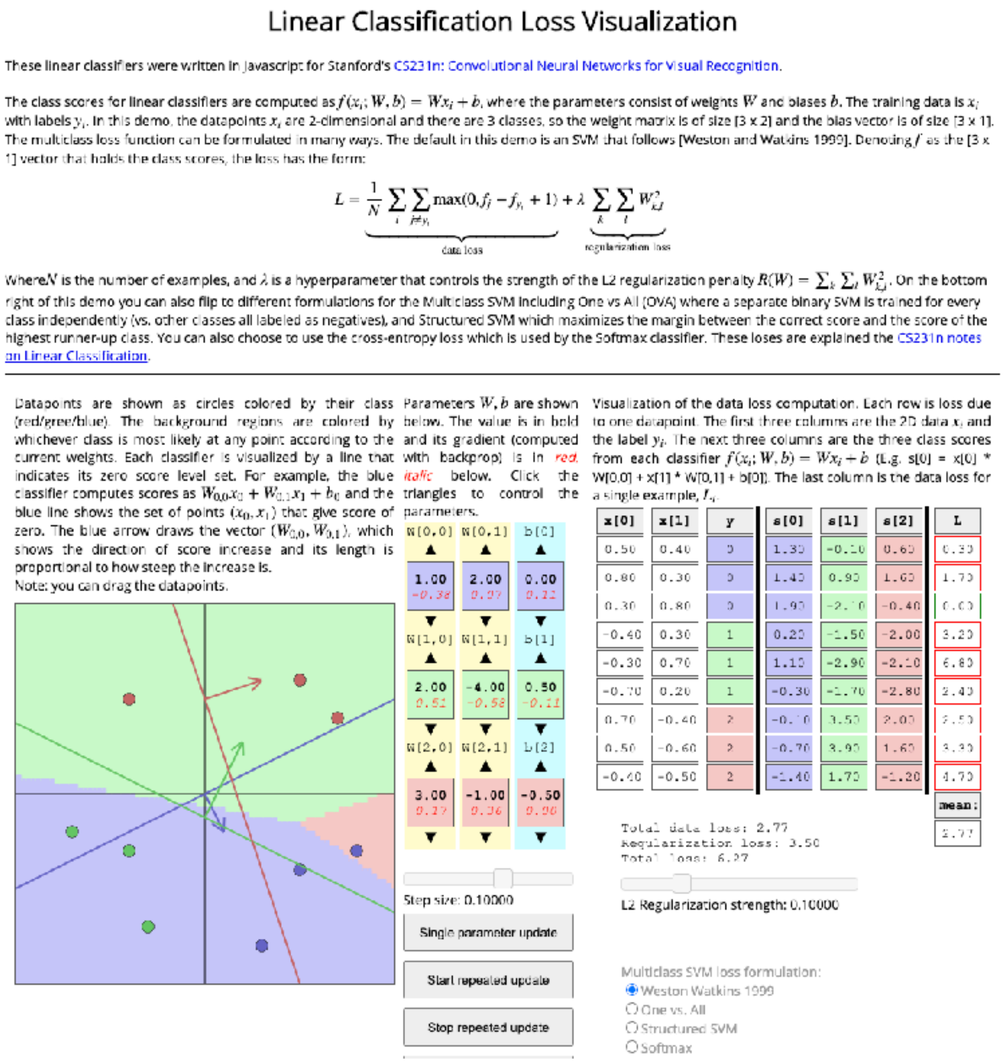
$$\nabla_{\mathbf{w}} J(\mathbf{w}) \approx \frac{1}{m}\sum_{i} \nabla_{\mathbf{w}} L(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w})$$

$$m \ll N$$

Hyperparameters?
a) intialize_weights
b) learning_rate
c) number of steps/truncation condition
d) batch size (8, 16, 32, …)
e) sampling

http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/

Interactive Demo



Linear Binary Classifier

$$y_i = \begin{cases} 1 & \text{if} \quad \mathbf{x}_i \in C \\ -1 & \quad\quad\; \mathbf{x}_i \notin C \end{cases}$$

Find weights $\mathbf{w}$ such that

$$\begin{aligned} \mathbf{x}_i^T \mathbf{w} &\geq 0 \quad \text{if} \quad y_i = 1 \\ \mathbf{x}_i^T \mathbf{w} &< 0 \quad \text{if} \quad y_i = -1 \end{aligned}$$

$$y_i \left(\mathbf{x}_i^T \mathbf{w} \right) \geq 0$$

Perceptron Algorithm [Rosenblatt ’57]

- **Initialize** $\mathbf{w}$
- **While** $\mathscr{L}(\mathbf{w} > 0)$
  - $v \leftarrow$ **set of indices with** $y_i(\mathbf{x}_i^T \mathbf{w}) < 0$
  - $\mathbf{w} \leftarrow \mathbf{w} - \alpha \sum_{i \in V} (-y_i \mathbf{x}_i) = \mathbf{w} + \alpha \sum_{i \in V} y_i \mathbf{x}_i$

Linear Classifier via Regression

Linear Classifier via Regression

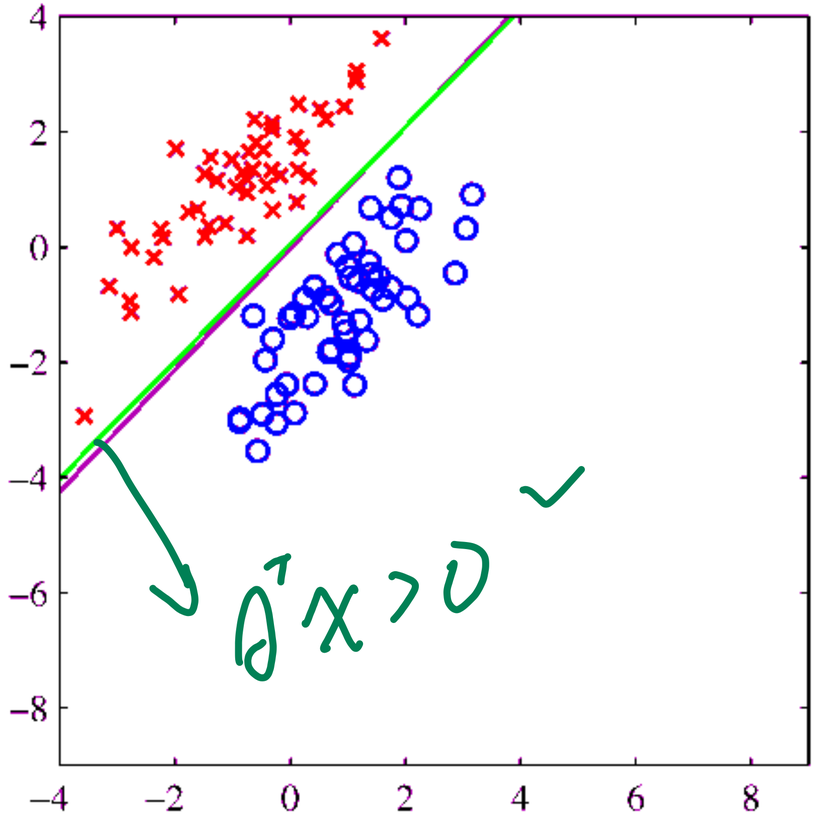
$$L(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta(\mathbf{x}^{(i)}) - y^{(i)}\right)^2$$

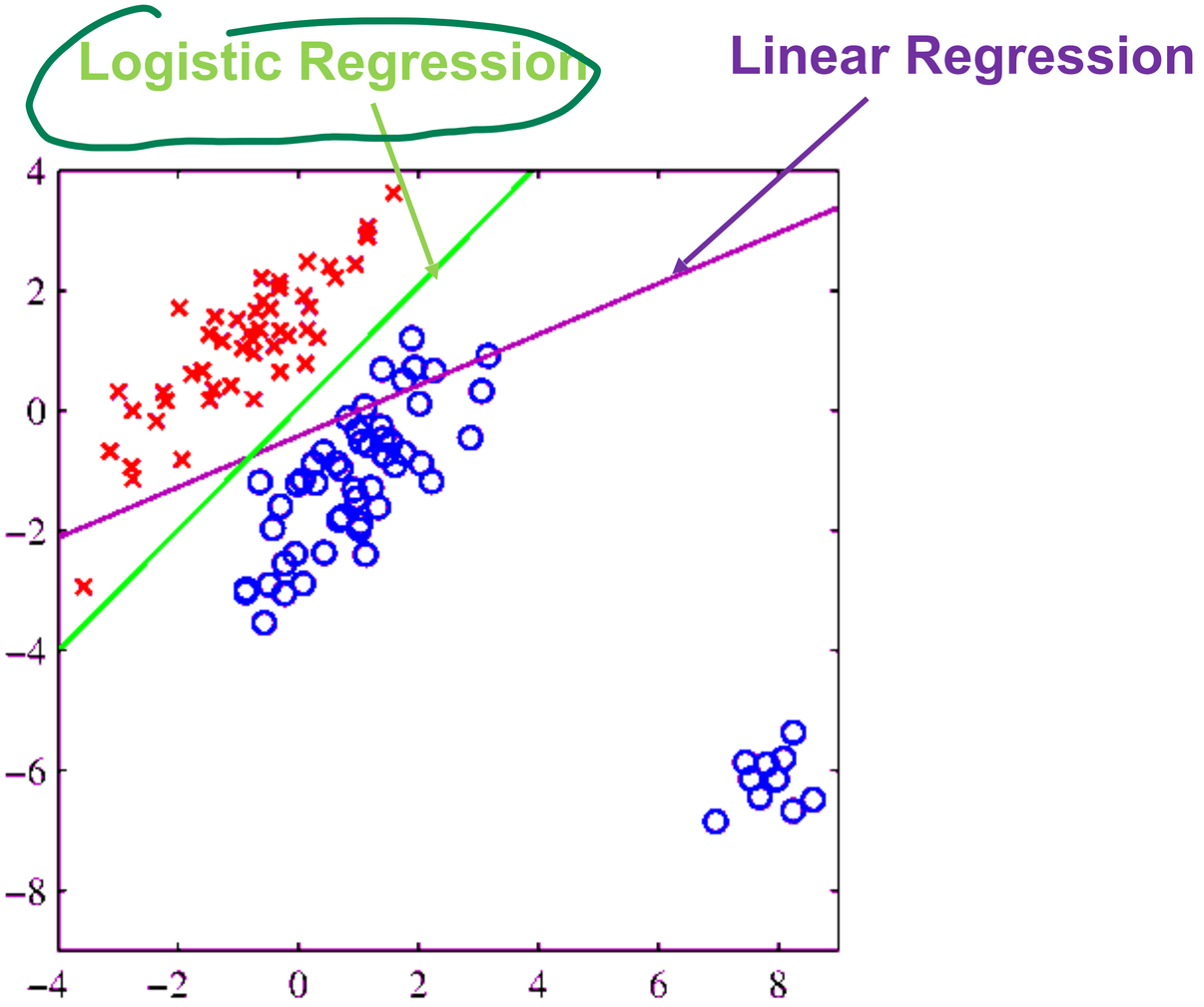
$$h(\mathbf{x}) = \sum_{i=1}^{p}\theta_i x_i = \theta^T\mathbf{x}$$

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

Compute gradient

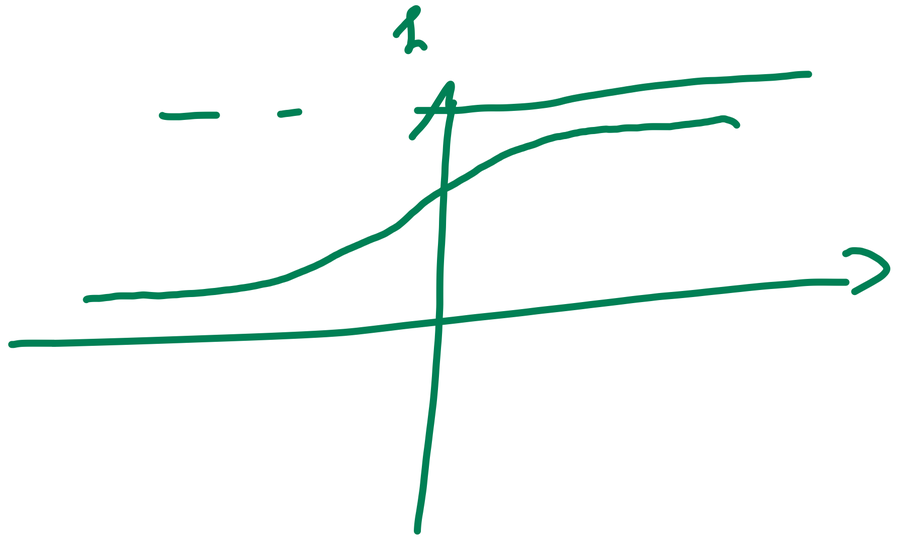
Linear vs Logistic Regression





Logistic Regression

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



$$\sigma'(z) = \frac{d\sigma(z)}{dz}$$

Logistic Regression

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

$$h(\mathbf{x}) = \sigma(\theta^T \mathbf{x}) = \frac{1}{1+\exp(\theta^T \mathbf{x})}$$

$$p(y=1|\mathbf{x};\theta) =$$

$$p(y=0|\mathbf{x};\theta) =$$

$$p(y|\mathbf{x};\theta) = \sigma(\theta^T \mathbf{x})^y (1-\sigma(\theta^T \mathbf{x}))^{1-y}$$

Logistic Regression

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

$$h(\mathbf{x}) = \sigma(\theta^T \mathbf{x}) = \frac{1}{1 + \exp(\theta^T \mathbf{x})}$$

$$p(y|\mathbf{x};\theta) = \sigma(\theta^T \mathbf{x})^y (1 - \sigma(\theta^T \mathbf{x}))^{1-y}$$

$$p(y|\mathbf{x};\theta) = \Pi_i \sigma(\theta^T \mathbf{x}^{(i)})^{y^{(i)}} (1 - \sigma(\theta^T \mathbf{x}^{(i)}))^{1-y^{(i)}}$$

Logistic Regression

$$p(y|\mathbf{x};\theta) = \Pi_i \sigma(\theta^T \mathbf{x}^{(i)})^{y^{(i)}} (1 - \sigma(\theta^T \mathbf{x}^{(i)}))^{1-y^{(i)}}$$

$$\log(p(y|\mathbf{x};\theta))$$

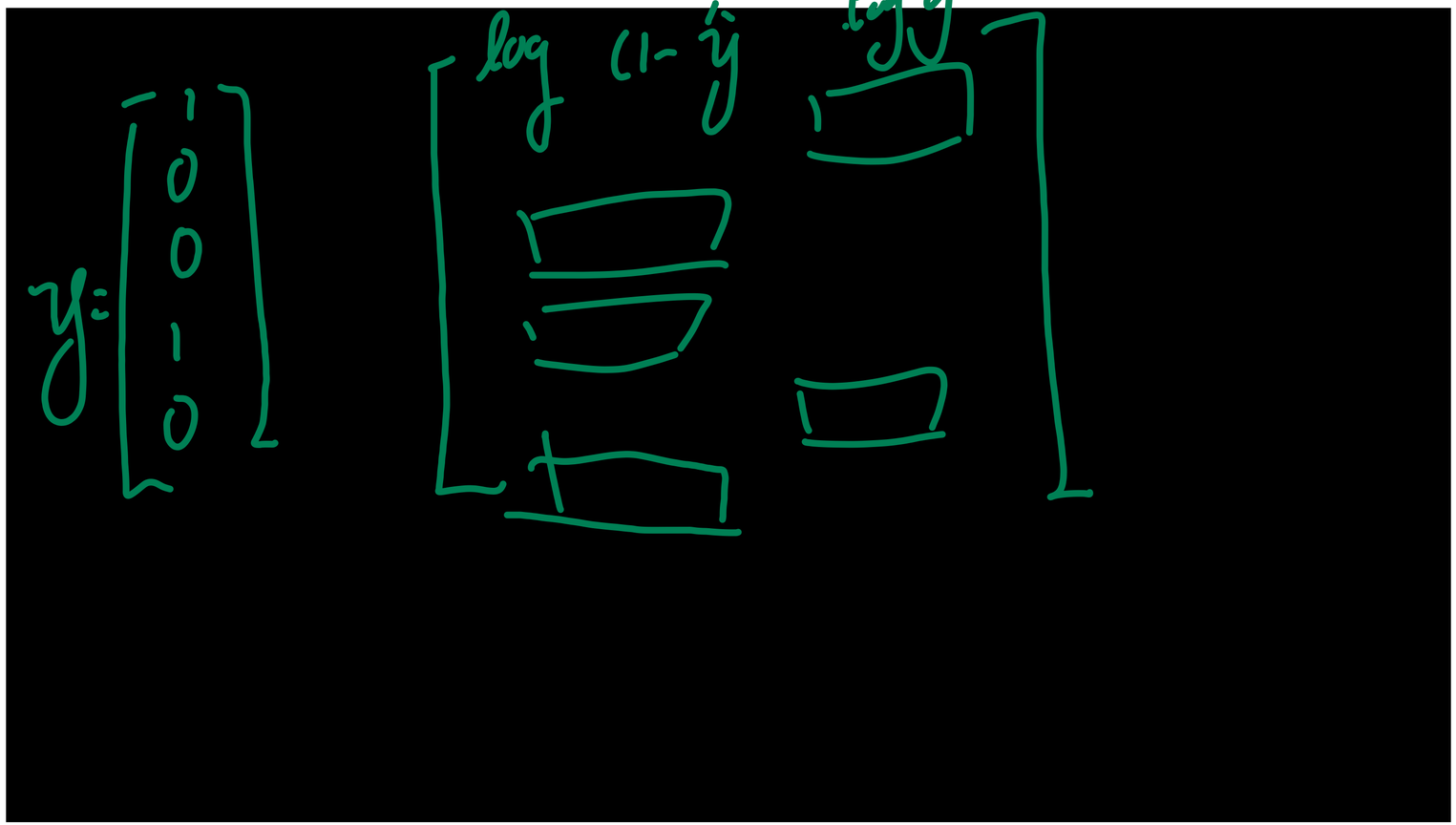
to find max

$$\theta = \theta + \alpha \left(y^{(i)} - h_\theta(\mathbf{x}^{(i)}) \right) \mathbf{x}^{(i)}$$

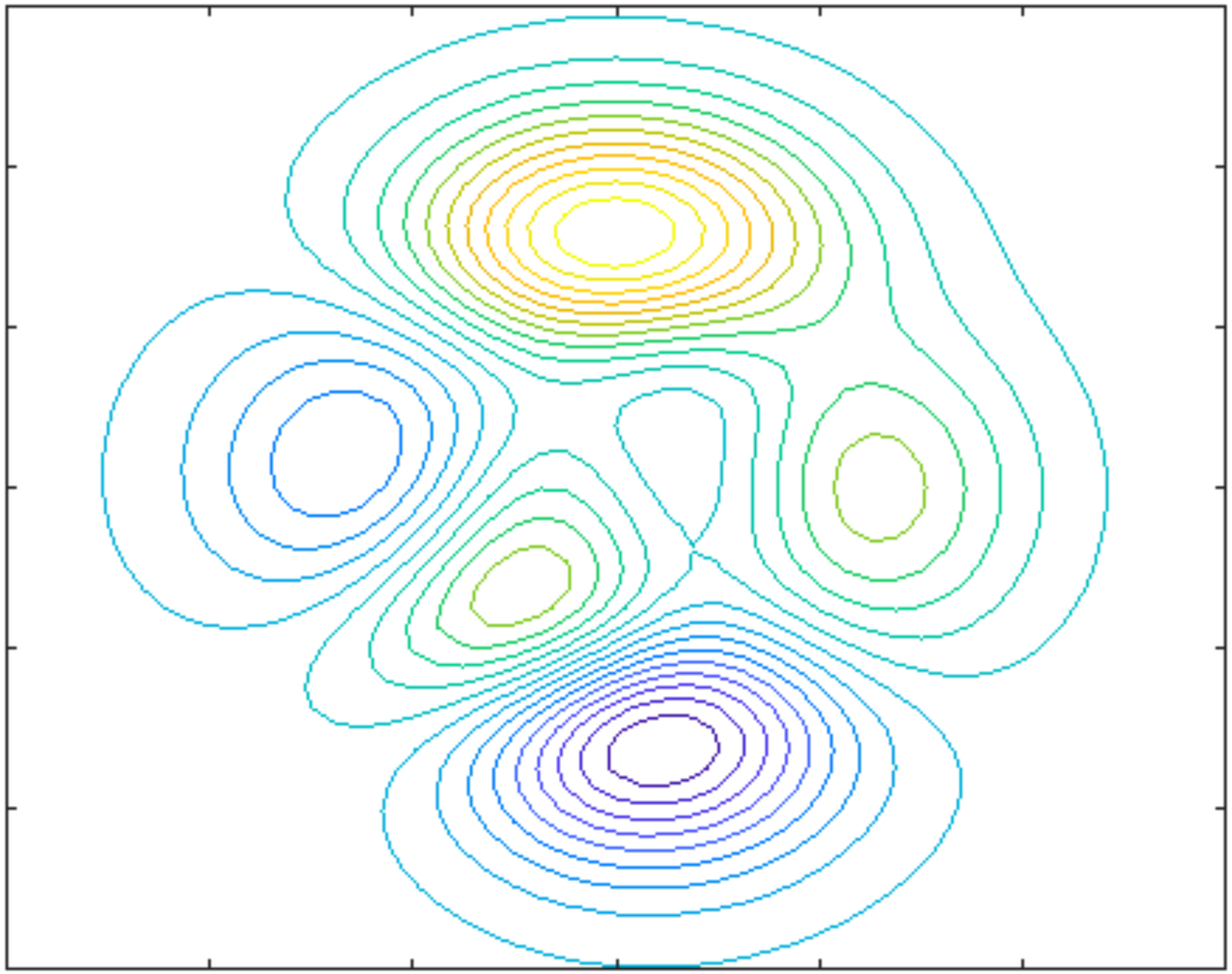
gradient ascend

Negative LogLikelihood

$$\log \mathbb{P}(\mathbf{D}|\theta) = \sum_i (y_i \log \hat{y}_{\theta,i} + (1 - y_i) \log(1 - \hat{y}_{\theta,i}))$$



Problems with GD (or SGD)



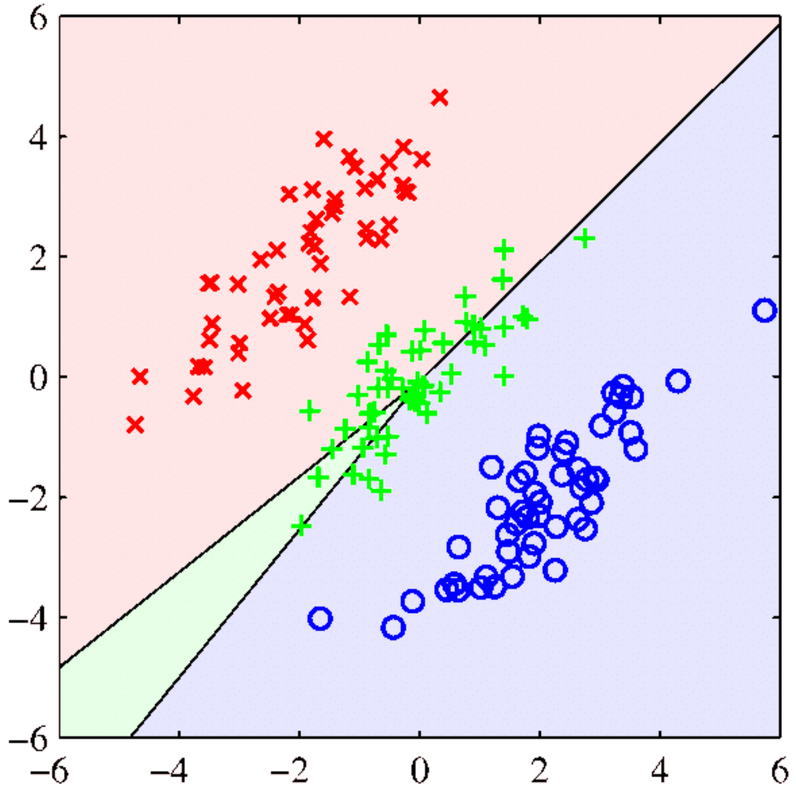
$$v_{t+1} = \rho v_t + \nabla f(w_t)$$

$$w_{t+1} = w_t - \alpha v_{t+1}$$

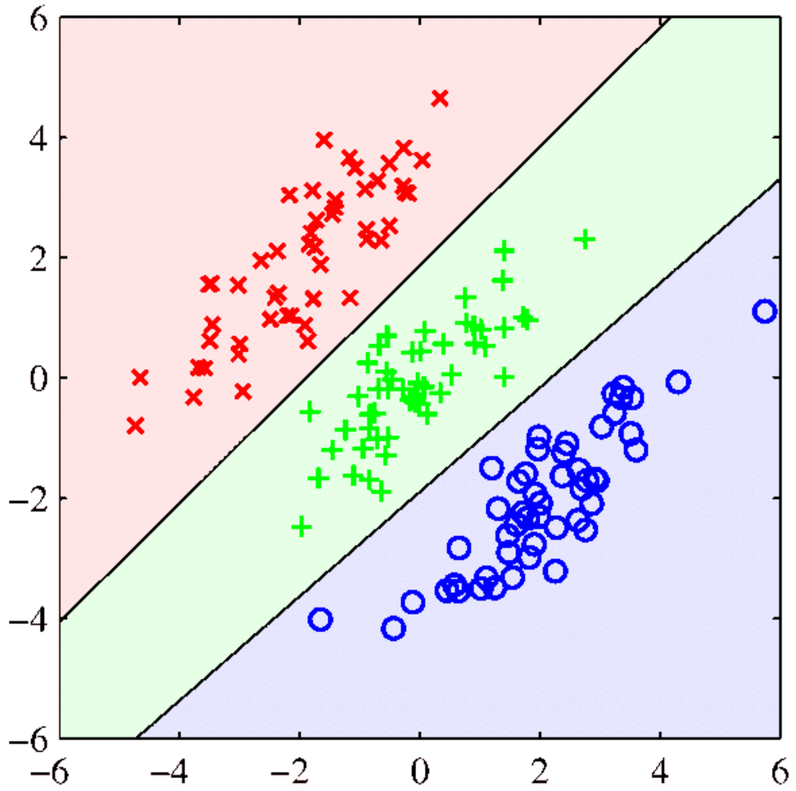
$$= w_t - \alpha\rho v_t - \alpha\nabla f(w_t)$$

Logistic vs Linear Regression, n>2 classes

Linear regression

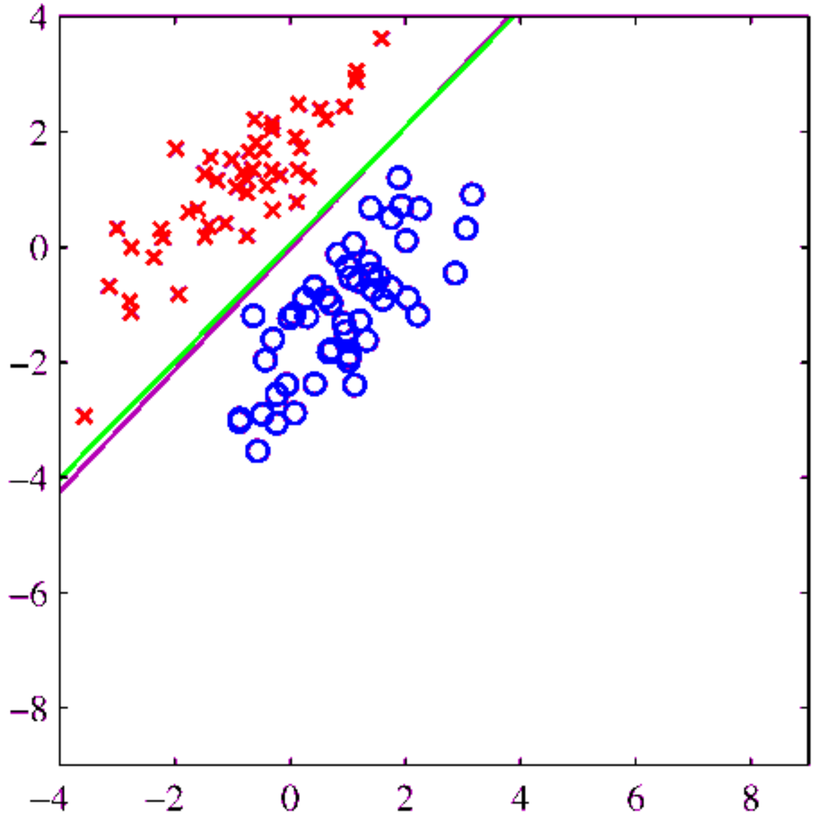


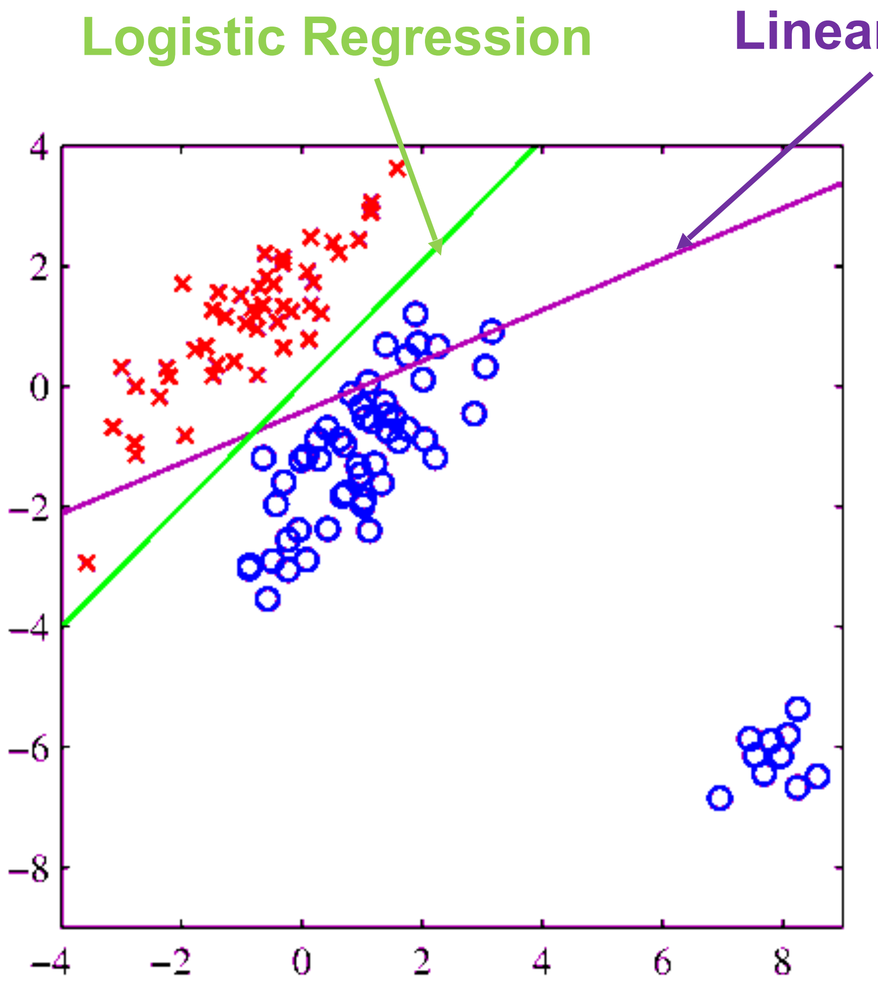
Logistic regression



Logistic regression does not exhibit the masking problem

Logistic vs Linear Regression





Multiple Classes & Linear Regression

C classes: one-of-c coding (or one-hot encoding)

4 classes, i-th sample is in 3rd class:

$$\mathbf{y}^i = (0, 0, 1, 0)$$

Matrix notation:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}^1 \\ \hline \vdots \\ \hline \mathbf{y}^N \end{bmatrix} = \left[\begin{array}{c|c|c} \mathbf{y}_1 & \dots & \mathbf{y}_C \end{array} \right] \quad \text{where} \quad \mathbf{y}_c = \begin{bmatrix} y_c^1 \\ \vdots \\ y_c^N \end{bmatrix}$$

$$\mathbf{W} = \left[\begin{array}{c|c|c} \mathbf{w}_1 & \dots & \mathbf{w}_C \end{array} \right]$$

Loss function:

$$L(\mathbf{W}) = \sum_{c=1}^{C} (\mathbf{y}_c - \mathbf{X}\mathbf{w}_c)^T (\mathbf{y}_c - \mathbf{X}\mathbf{w}_c)$$

Least squares fit (decouples per class):

$$\mathbf{w}_c^* = (\mathbf{X}^T\mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}_c$$

Multiple Classes via SoftMax