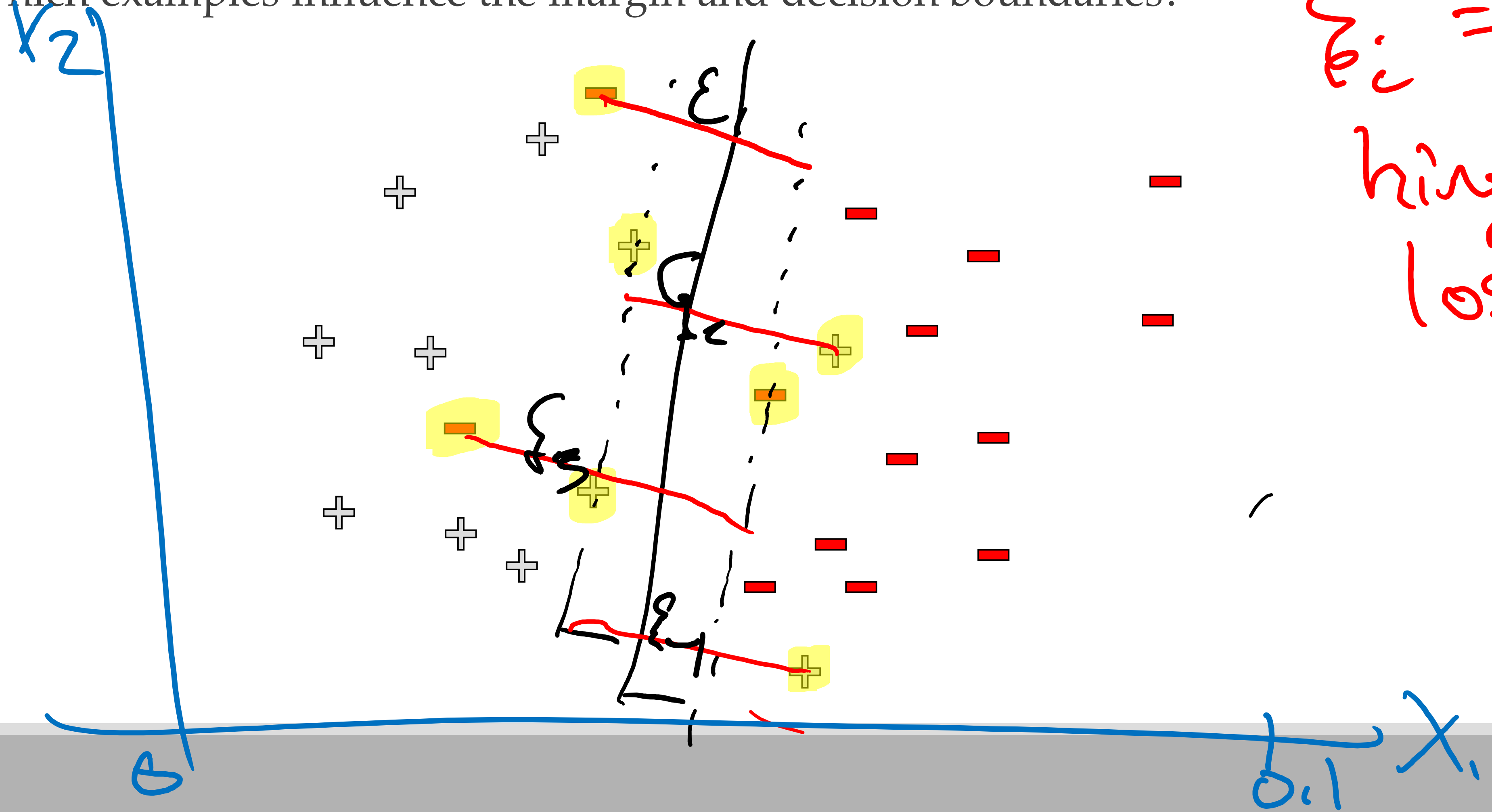
CS 412

FEB 20TH – NEURAL NETWORKS

HTF – CHAPTER 11

Support vectors of SVMs

Which examples influence the margin and decision boundaries?



ξ_i = hinge loss

Hinge Loss

st. |hyp| ≥ 1

Soft Margin

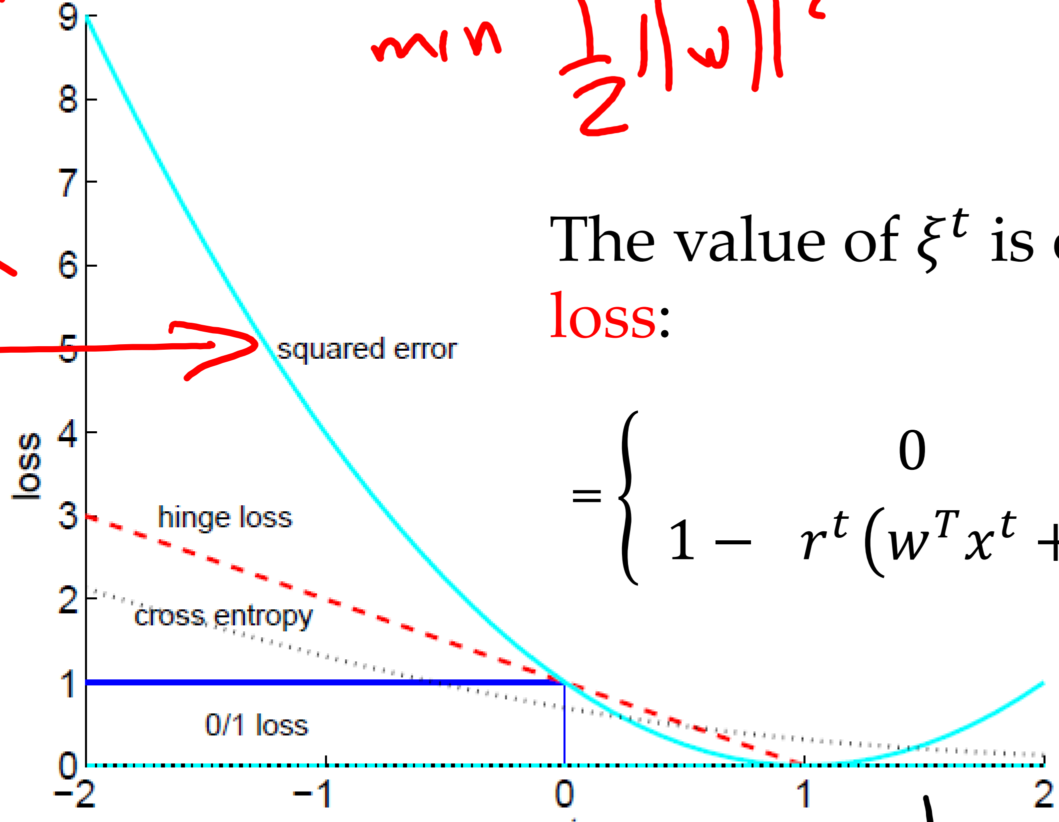
$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_t \xi^t \quad \text{subject to } r^t(\mathbf{w}^T x^t + w_0) \geq 1 - \xi^t$$

$$\xi^t \geq 0$$

min $\frac{1}{2}\|w\|^2$

$r_t - y_t$

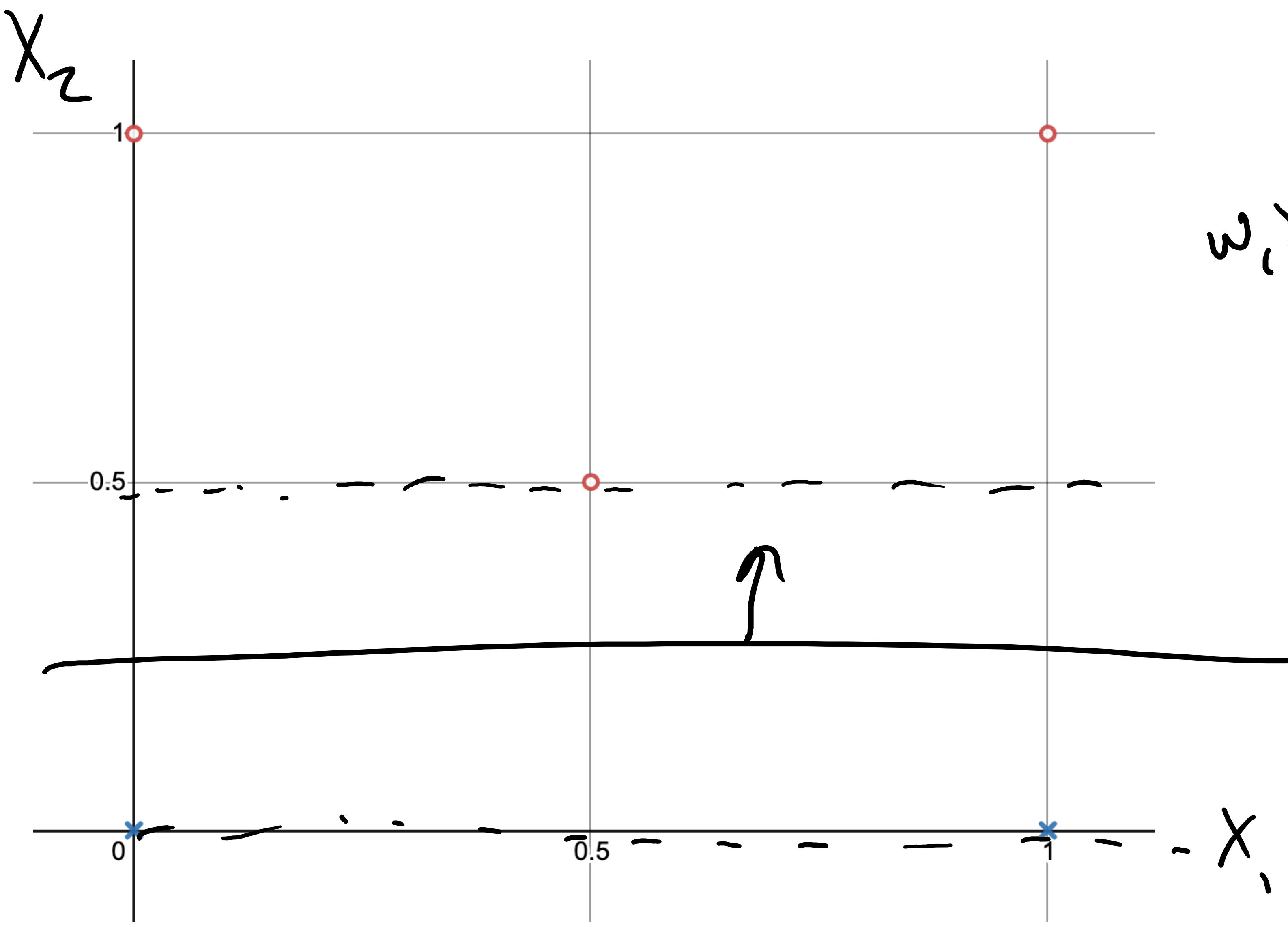
linear regression



$r^t(w^T x^t + w_0)$ ← hyperplane

The value of ξ^t is called hinge loss:

$$= \begin{cases} 0 & \text{if } r^t(w^T x^t + w_0) \geq 1 \\ 1 - r^t(w^T x^t + w_0) & \text{otherwise} \end{cases}$$



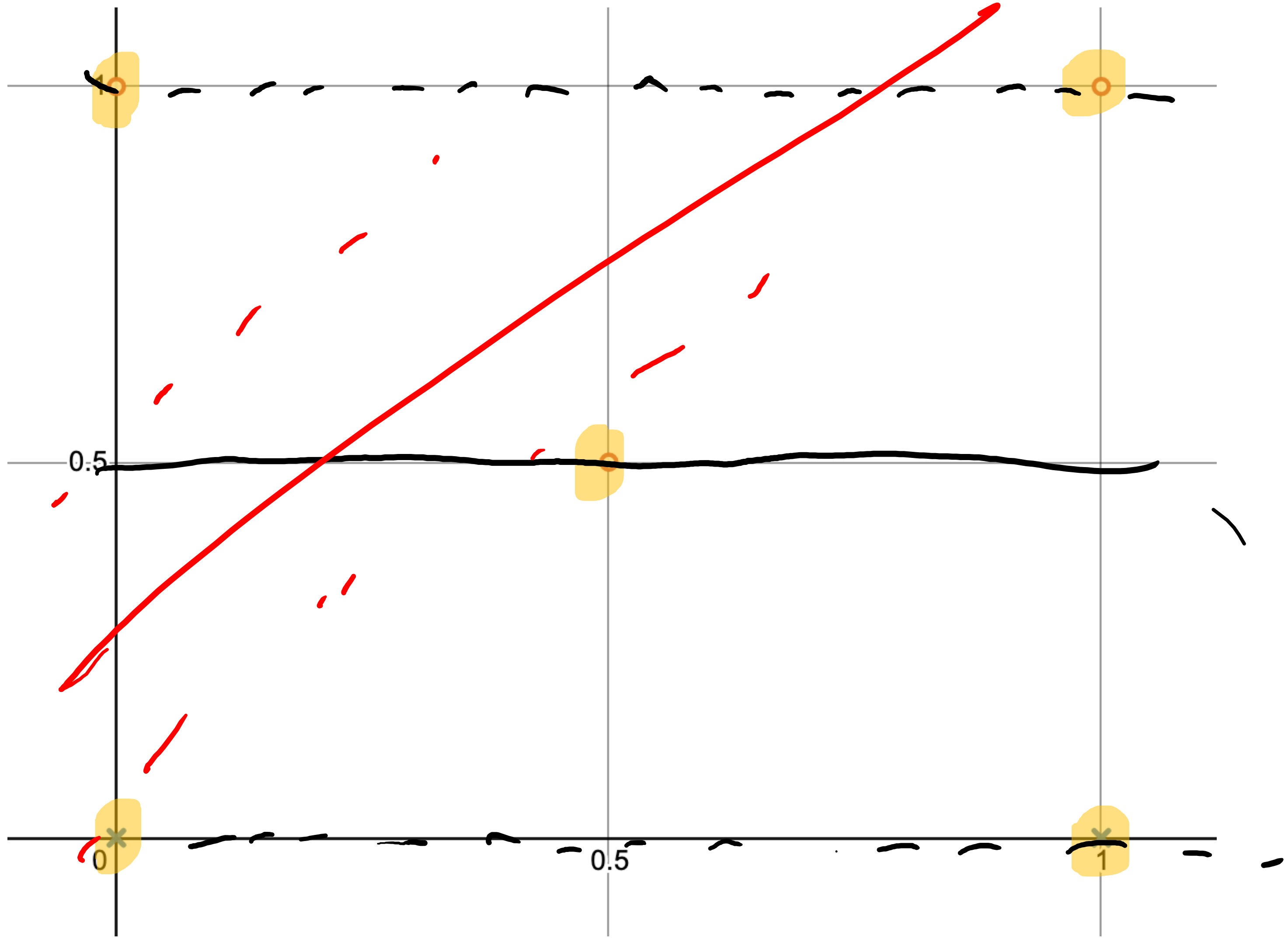
$w_1 = 0$

$w_1 x_1 + w_2 x_2 + w_0$

What is w for the hard margin separator?

$\sum_i \varepsilon_i = 0$

high C (>6?)



not possible b/c no support vectors

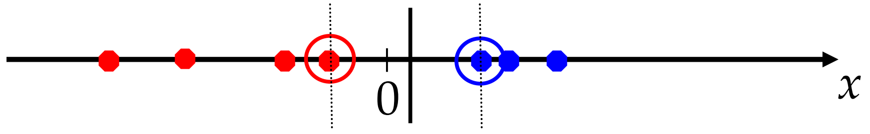
$$\sum_i \varepsilon_i = 1$$

What is w for the soft margin separator?

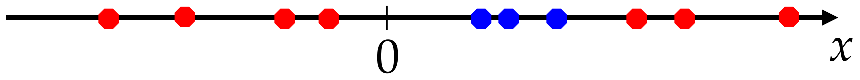
low $c \leq 6$

Non-linear SVMs

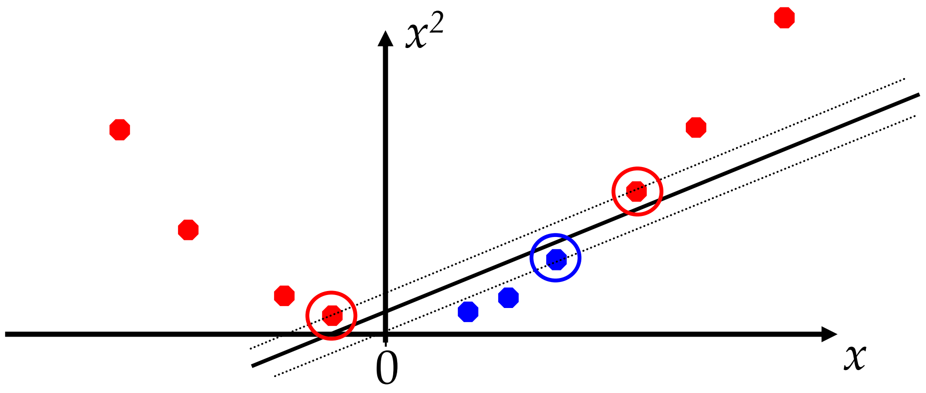
Datasets that are linearly separable with some noise work out great:



But what are we going to do if the dataset is just too hard?



How about… mapping data to a higher-dimensional space:



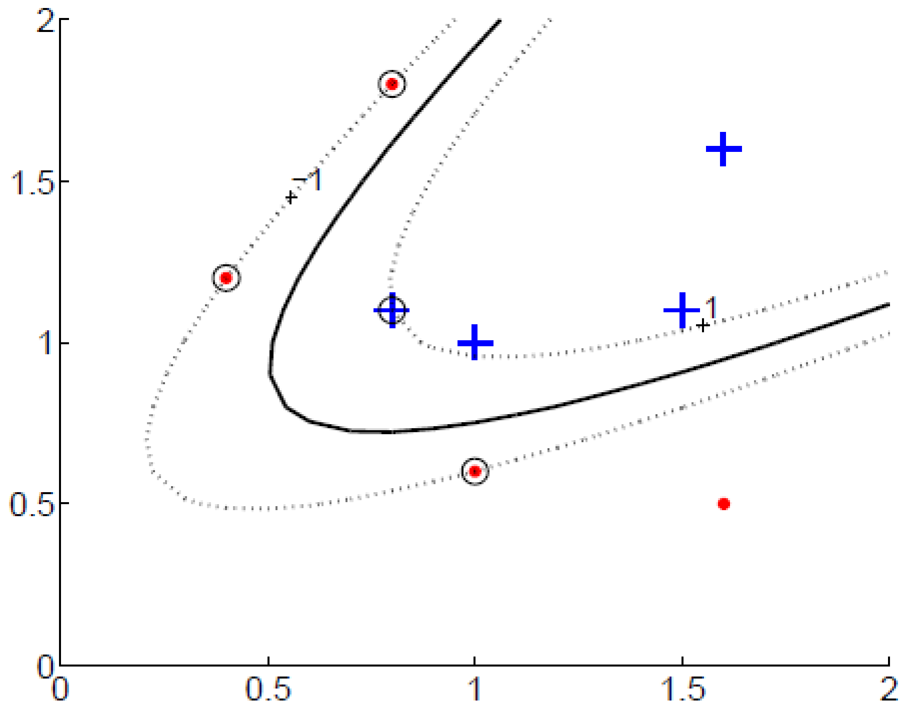
Vectorial Kernels

- Polynomials of degree q:

$$K\left(\mathbf{x}^t,\mathbf{x}\right)=\left(\mathbf{x}^T\mathbf{x}^t+1\right)^q$$

discrete value

$$K(\mathbf{x},\mathbf{y})=\left(\mathbf{x}^T\mathbf{y}+1\right)^2$$
$$=\left(x_1y_1+x_2y_2+1\right)^2$$
$$=1+2x_1y_1+2x_2y_2+2x_1x_2y_1y_2+x_1^2y_1^2+$$
$$\phi(\mathbf{x})=\left[1,\sqrt{2}x_1,\sqrt{2}x_2,\sqrt{2}x_1x_2,x_1^2,x_2^2\right]^T$$



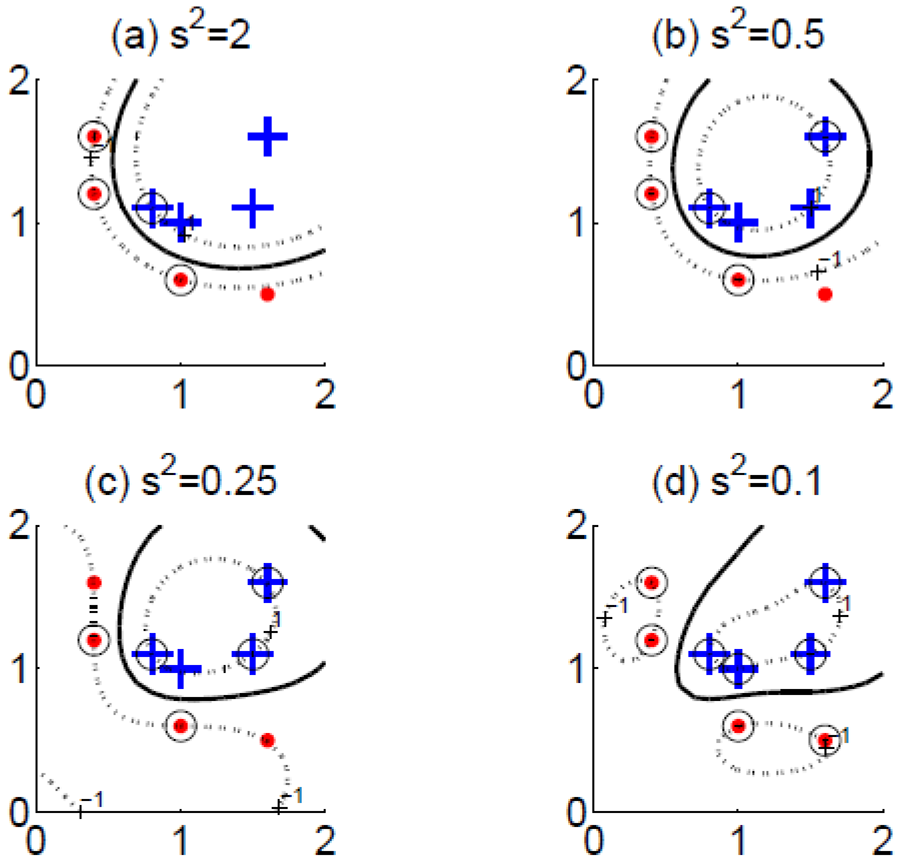
Vectorial Kernels

- Radial-basis functions:

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

"rbf"

γ

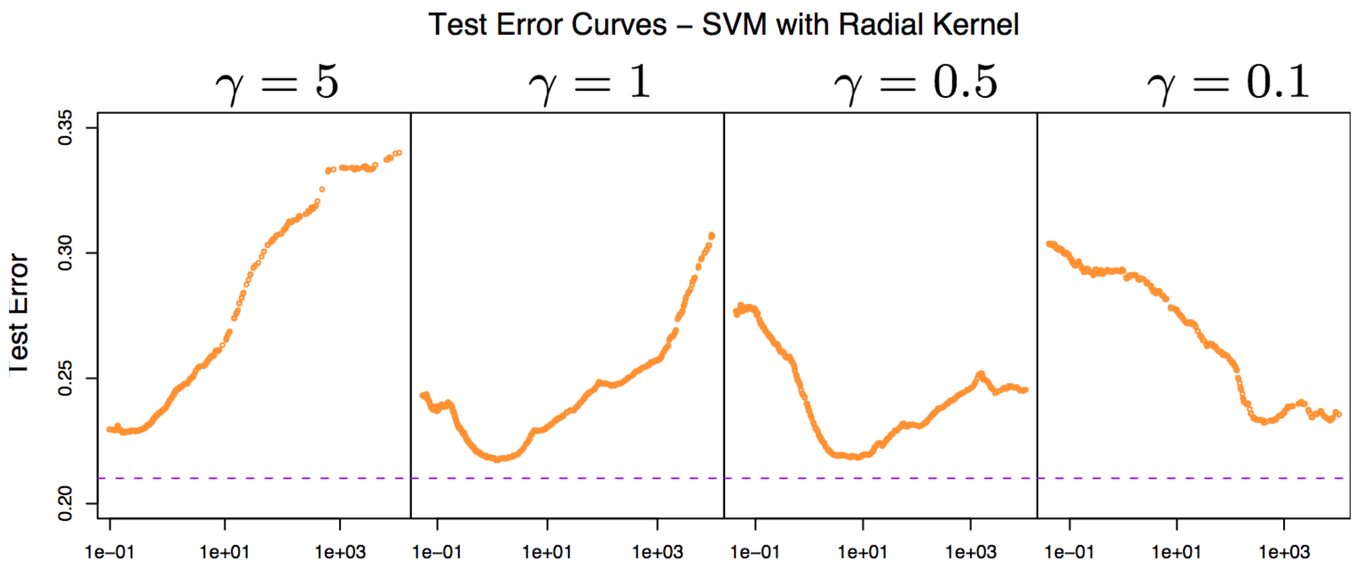


Sklearn SVC classifier

Overfitting

Because of the high dimensionality of the kernel model, there is a stronger need to adjust for overfitting

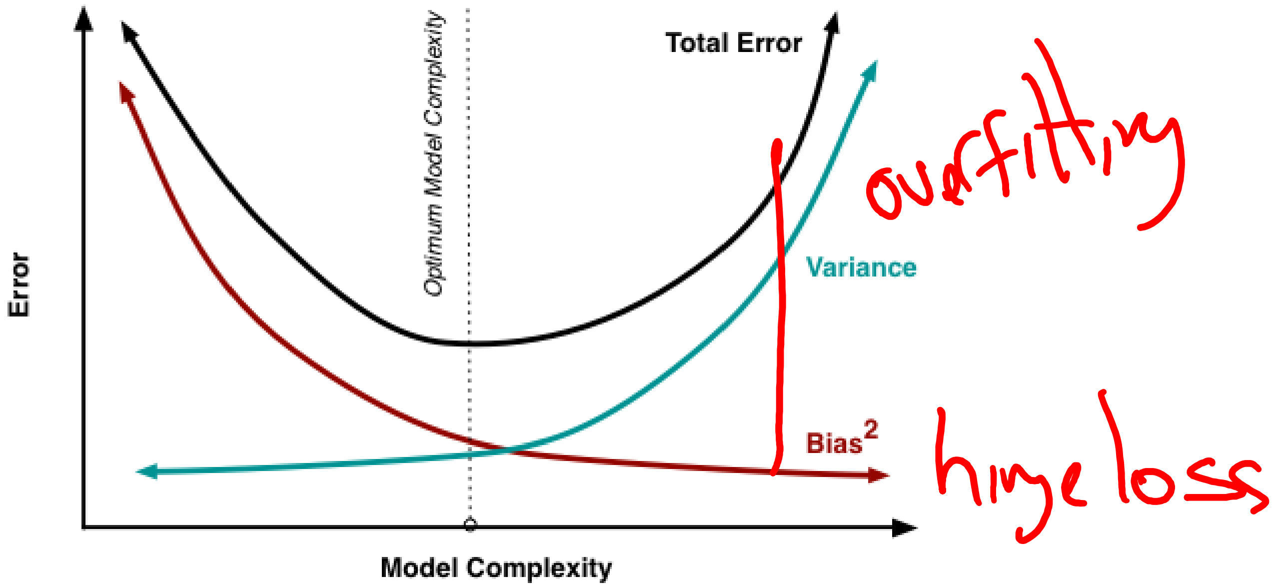
- Here *c (x-axis)* is the regularization paramater and
- λ is the "scale parameter" for the model which indicates its allowed complexity.



Over/underfitting

We also now have a concept of “training error”

- What is the training error for the SVM?



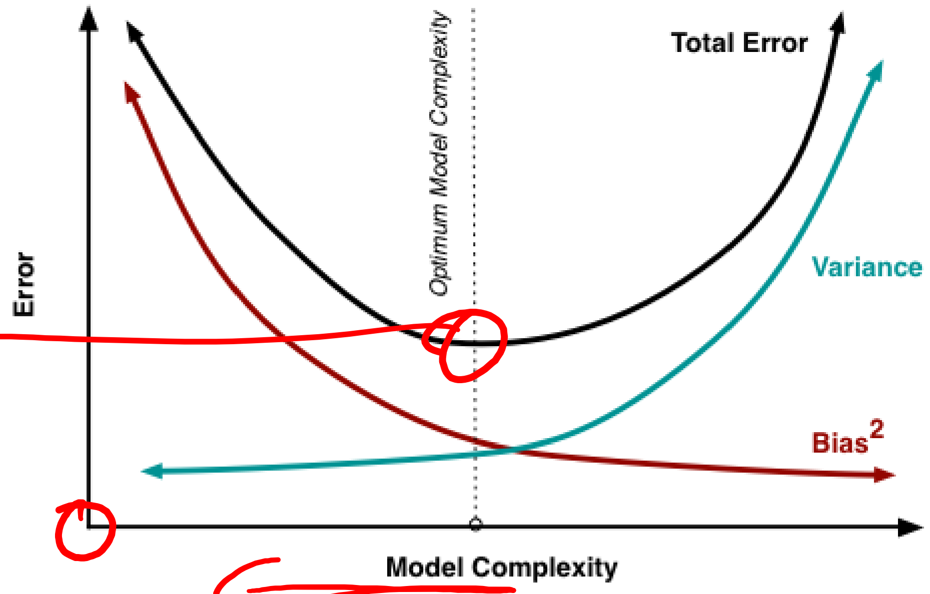
Over/underfitting

$C>0$

We also now have a concept of "training error"

- What is the training error for the SVM?
- As complexity increases, should the error go up or down?

E_{CV}

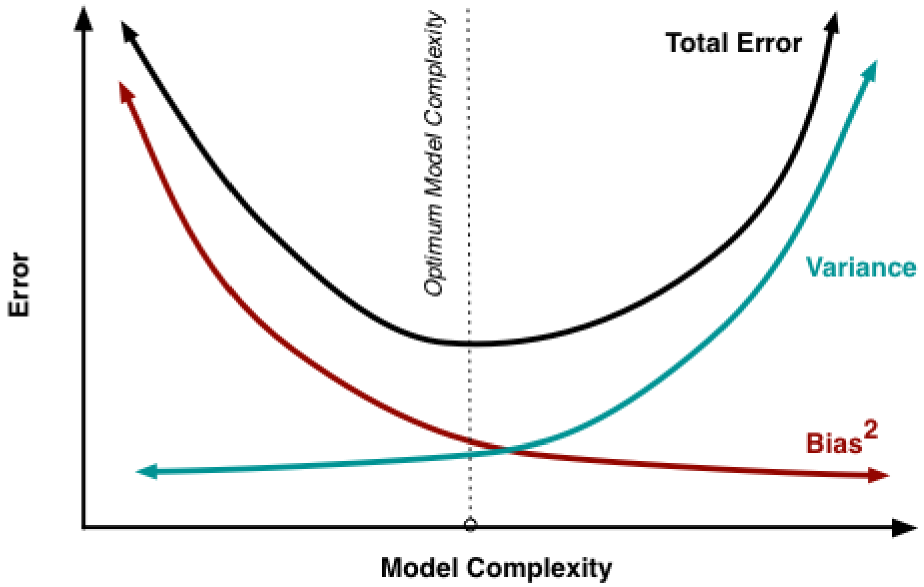


γ

$C=\infty$

Over/underfitting

We also now have a concept of "training error"
- What is the training error for the SVM?
- As complexity increases, should the error go up or down?
- Do we have a variance for this error?



Over/underfitting

We're now also considering a much larger set of models

- kNN, we only did 25 models
- SVM, we could do thousands of them

We pay a penalty when we select the "best" model from a large set of hypothesis models, but we may want to try several in order to be certain

Next, we'll start Neural Networks and this will be the focus of the next couple weeks

- Even more complexity
- Even less interpretability
- Very good at predicting, however

Neural Networks

Networks of processing units (neurons) with connections (synapses) between them

Large number of neurons: 10^{10}

Large connectitivity: 10^5

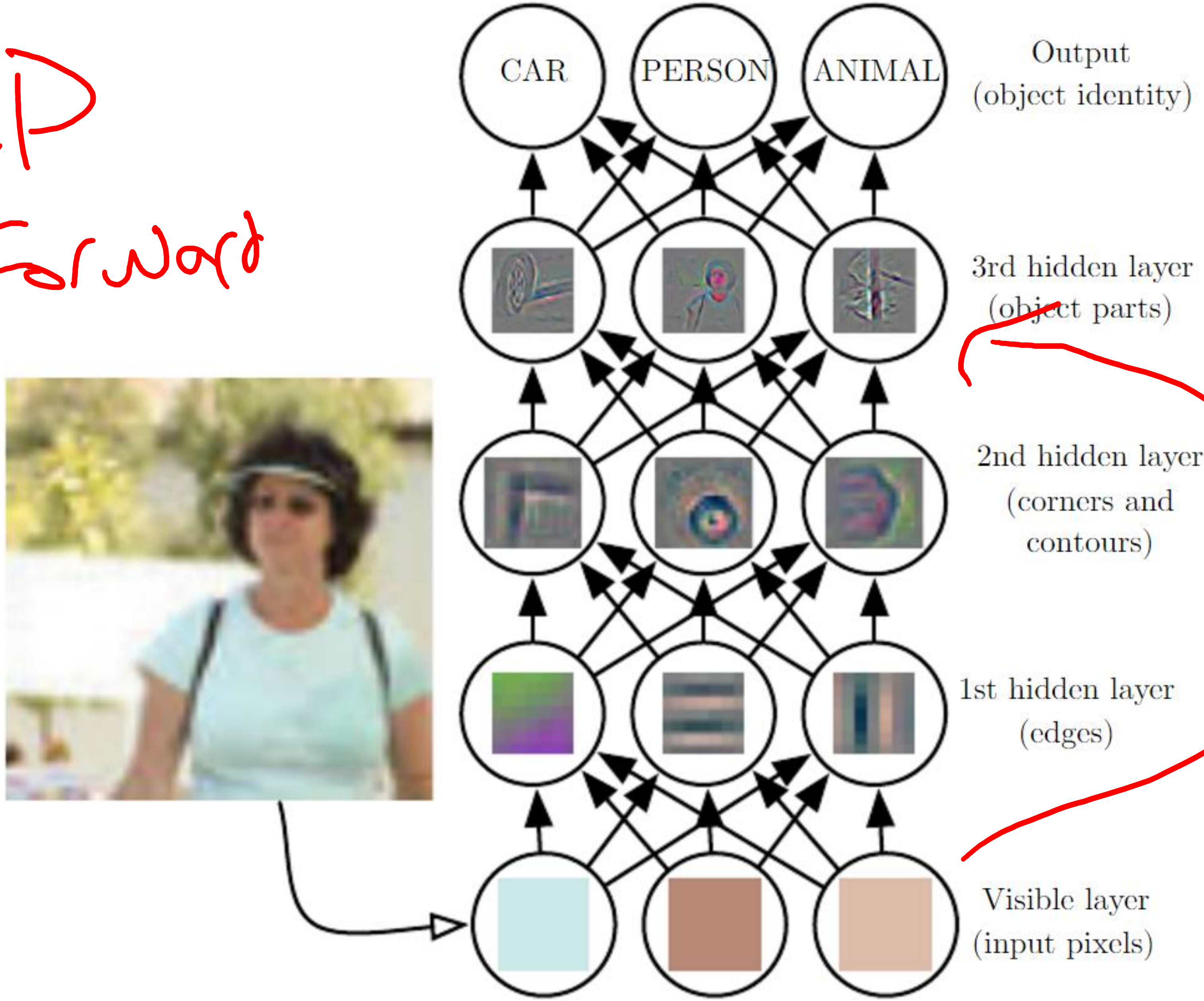
Parallel processing

Distributed computation/memory

Robust to noise, failures

MLP

Feed forward



not fully connected

Understanding the Brain

Levels of analysis for an information processing system such as sorting (Marr, 1982)

1. Computational theory: goal of computation and abstract definition of the task
2. Representation and algorithm: how to represent input and output, and how to transform from input to output
3. Hardware implementation

Reverse engineering: From hardware to theory

Parallel processing: SIMD vs MIMD

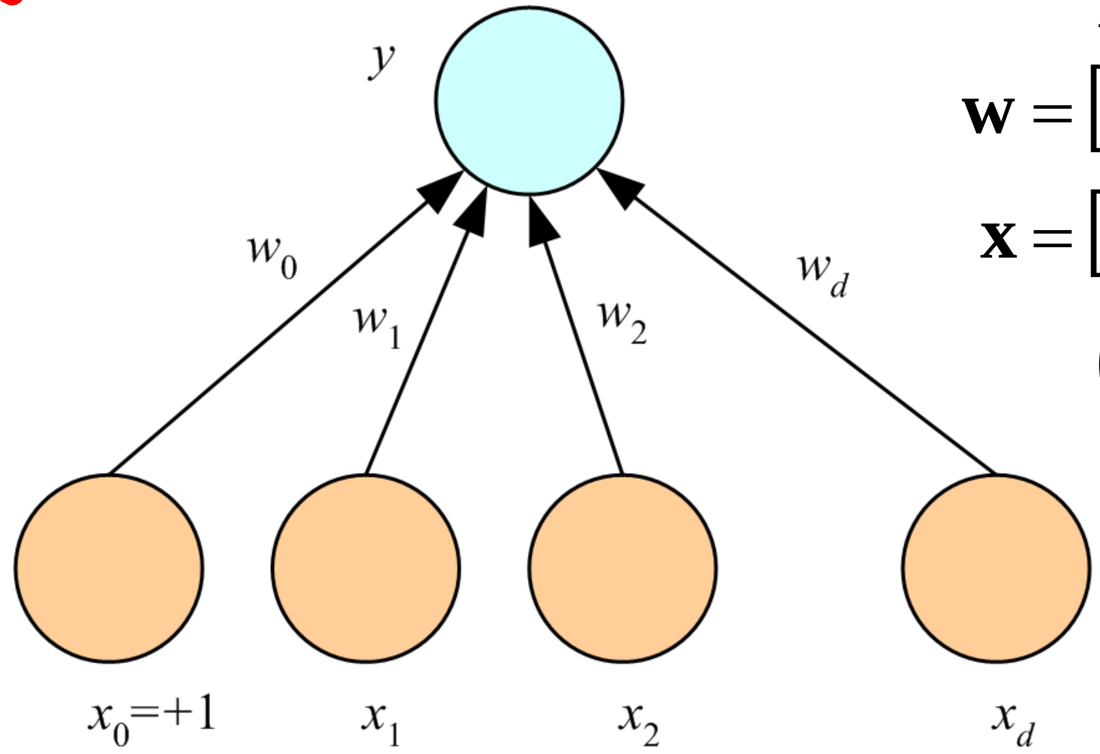
Neural net: SIMD with modifiable local memory

Learning: Update by training/experience

Single instruction
multiple data

Perceptron

Regression



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

$$\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$$

$$\mathbf{x} = [1, x_1, \ldots, x_d]^T$$

(Rosenblatt, 1962)

$y = 1 \cdot w_0 + w_1 x_1 + w_2 x_2 + \ldots w_d x_d$

Perceptron

What is the single-layer perceptron?

Perceptron

What is the single-layer perceptron?

- Just the linear ~~discriminator~~ regressor
- No support vector constraints

How do we train it?

Perceptron

What is the single-layer perceptron?

- Just the linear discriminator
- No support vector constraints

How do we train it?

- Stochastic gradient descent
  - Small changes based on the data

Perceptron

What is the single-layer perceptron?

- Just the linear discriminator
- No support vector constraints

How do we train it?

- Stochastic gradient descent
  - Small changes based on the data – minimizing loss (what loss should we minimize?)
  - Update = learning factor*(DesiredOutput – Actual Output) * Input

$$\Delta w_{ij}^{t} = \eta\left(r_i^t - y_i^t\right)x_j^t$$

  - Descent is moderated by our learning factor (eta)

t = iteraton of gd

stochastic

↳ random batch

Perceptron

What is the single-layer perceptron?

- Just the linear discriminator
- No support vector constraints

How do we train it?

- Stochastic gradient descent
  - Small changes based on the data
  - Update = learning factor*(DesiredOutput – Actual Output) * Input

$$\Delta w_{ij}^t = \eta\left(r_i^t - y_i^t\right)x_j^t$$

loss function is strict error

  - Is this for classification or regression?

Cross Entropy

$p = [0, 1]$

Rather than RSS, which we use to measure error for regression, we want to consider a new error – **cross-entropy** which is commonly used for classification problems

$$H(p, q) = -\sum_{x} p(x) \log q(x)$$

What are p,q here?

x = data points

p = prediction for x

q = label for x

Cross Entropy

Rather than RSS, which we use to measure error for regression, we want to consider a new error – **cross-entropy** which is commonly used for classification problems

$$H(p, q) = -\sum_x p(x)\, \log q(x)$$

What are p,q here? The two things we want to compare: predicted vs actual

Cross Entropy

Rather than RSS, which we use to measure error for regression, we want to consider a new error – **cross-entropy** which is commonly used for classification problems

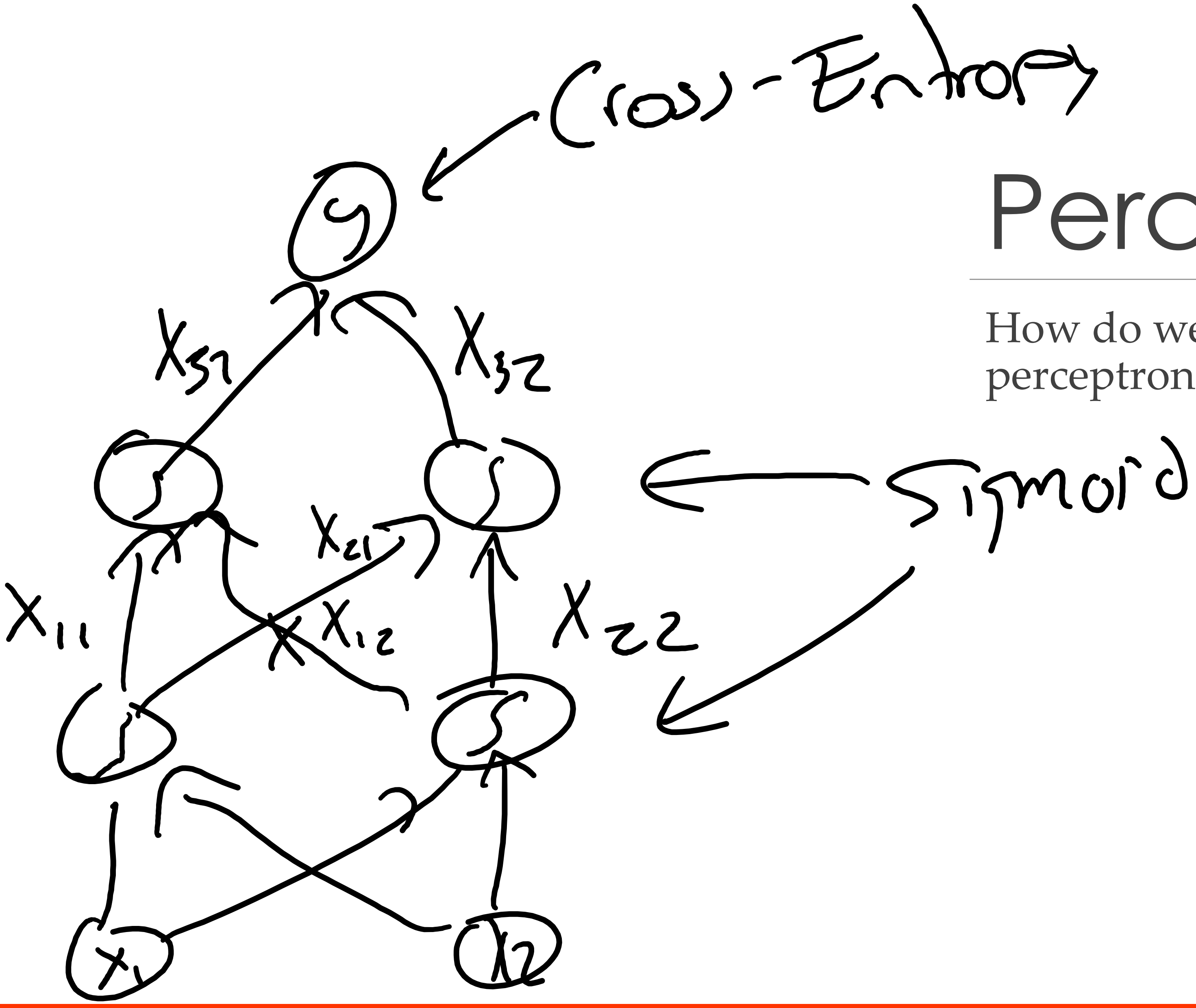
$$H(p, q) = -\sum_{x} p(x)\,\log q(x)$$

What are p,q here? The two things we want to compare: predicted vs actual

We will talk about this when we talk about model validation, but for now, I wanted to introduce the concept. Luckily for us, this is a *similar* approach to using the sigmoid as our activation function

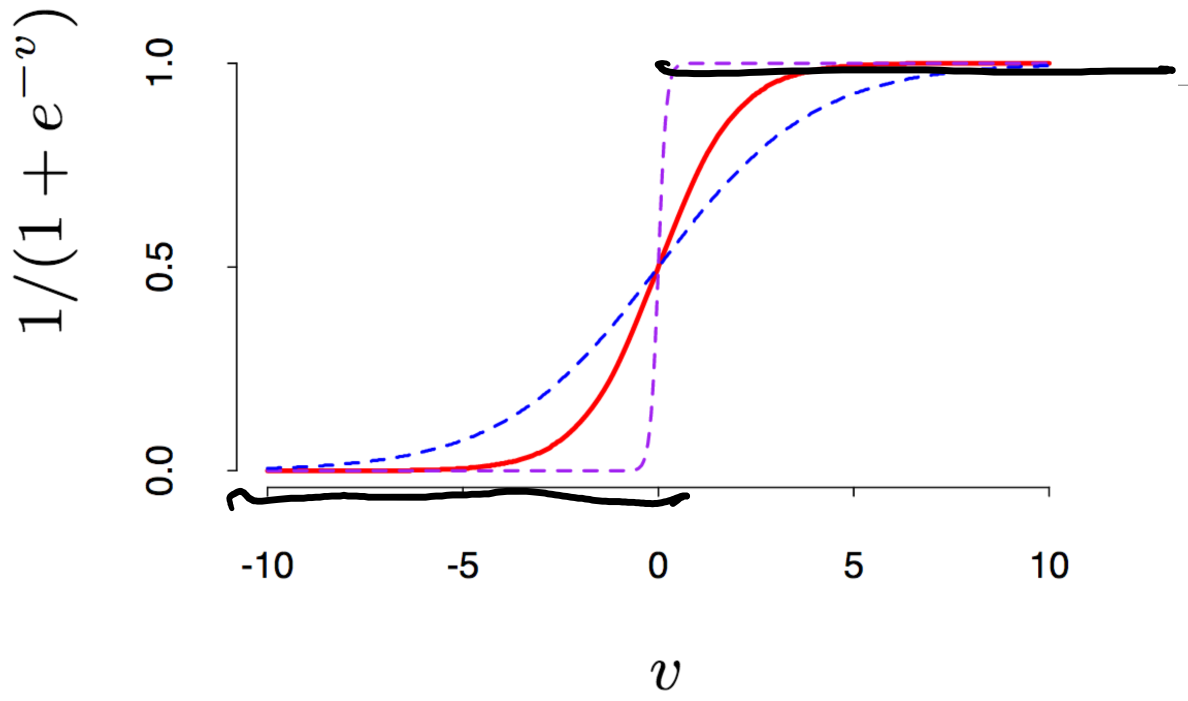
Perceptron

How do we combine outputs of multiple perceptrons together? Is it relevant?



Perceptron

How do we transform a perceptron for regression into a classifier?



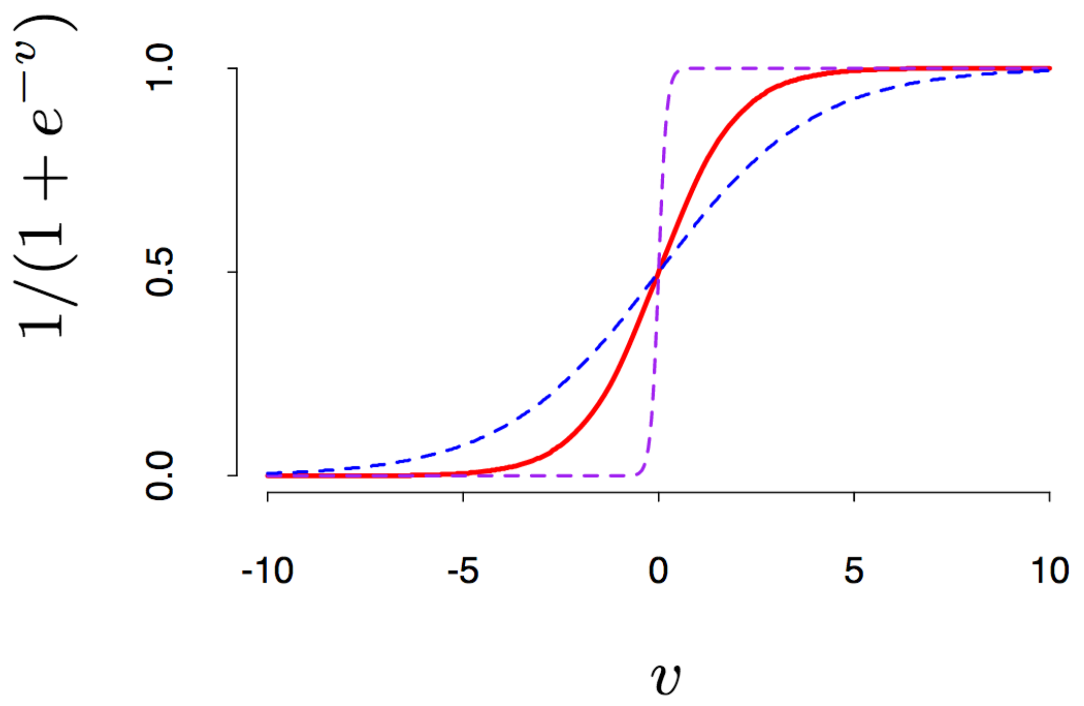
Perceptron

How do we make a regression model into a classification model?

- Activation function (here: sigmoid)
- Like logistic regression, there is no unique solution, so we also have to consider the rate at which the sigmoid transitions, this is the **activation rate**, s (here: ½,1,10)

Why do we prefer this to the sign function?

Points of highest uncertainty have largest slope. $\rightarrow \Delta w$



Perceptron

How do we make a regression model into a classification model?

- Activation function (here: sigmoid)
- Like logistic regression, there is no unique solution, so we also have to consider the rate at which the sigmoid transitions, this is the **activation rate**, s (here: ½,1,10)

Why do we prefer this to the sign function?

- They are differentiable and non-linear

Perceptron

What is the problem with the simple perceptron?

Perceptron

What is the problem with the simple perceptron?

- It can't model non-linear data

How do we fix this?

Perceptron

What is the problem with the simple perceptron?

- It can't model non-linear data

How do we fix this?

- SVM fixed this by using the kernel methods
- Can the perceptron?

Perceptron

What is the problem with the simple perceptron?

- It can't model non-linear data

How do we fix this?

- SVM fixed this by using the kernel methods
- Can the perceptron? **Yes**, but that's not what the neural network does

Perceptron

What is the problem with the simple perceptron?

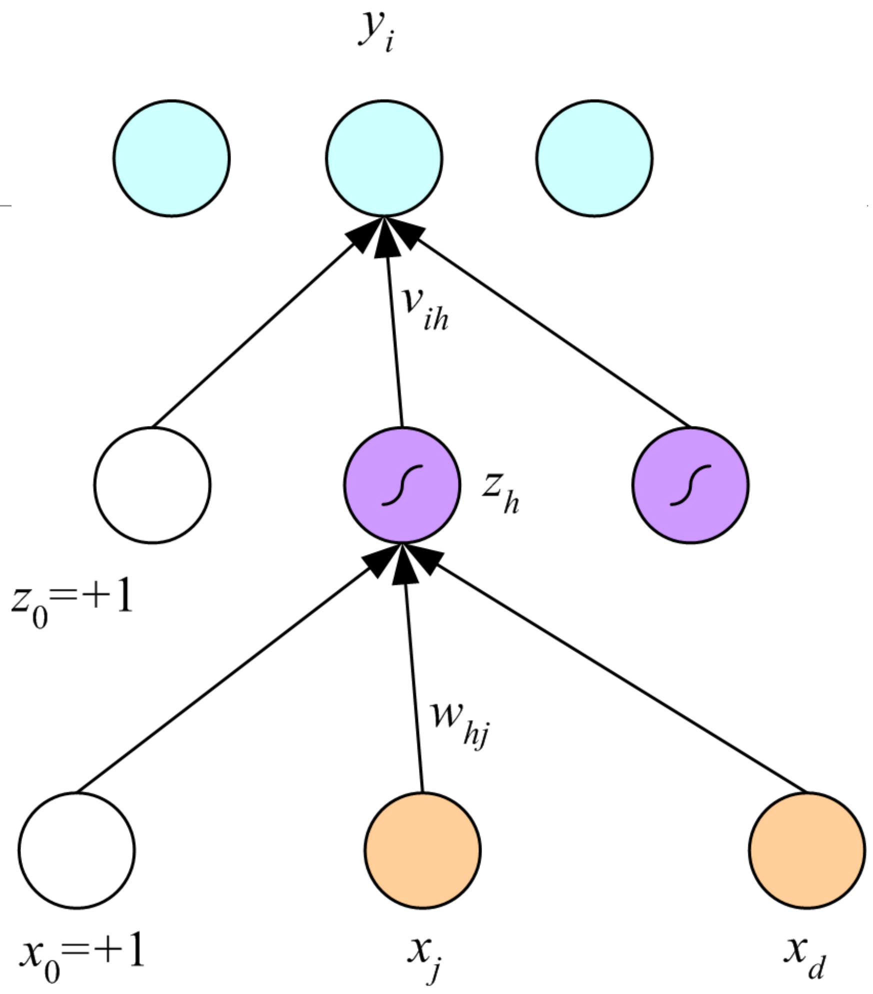
- It can't model non-linear data

How do we fix this?

- SVM fixed this by using the kernel methods
- Can the perceptron? **Yes**, but that's not what the neural network does

Let's add multiple layers to the perceptron

- At each level we have a **regression** model defined by the activation function and **always** a constant w_0



Perceptron

What is the problem with the simple perceptron?

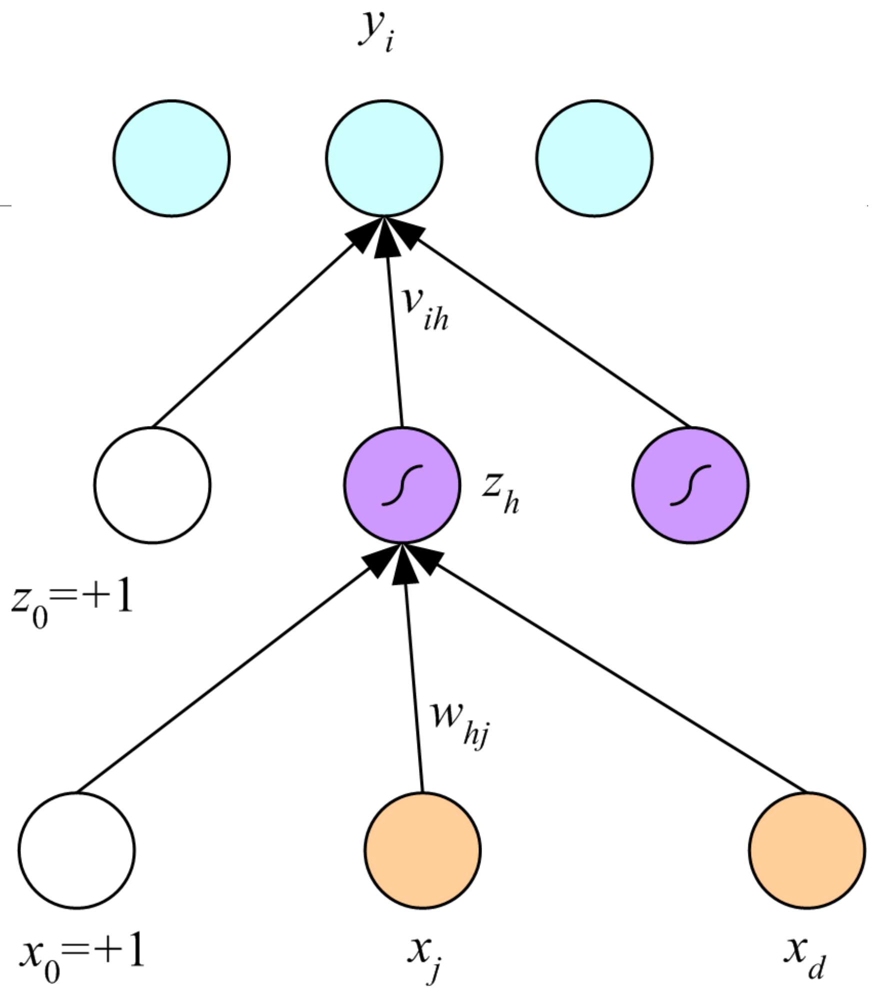
- It can't model non-linear data

How do we fix this?

- SVM fixed this by using the kernel methods
- Can the perceptron? **Yes**, but that's not what the neural network does

Let's add multiple layers to the perceptron

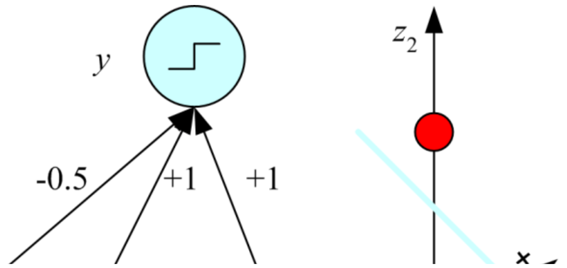
- At each level we have a **regression** model defined by the activation function and **always** a constant w_0

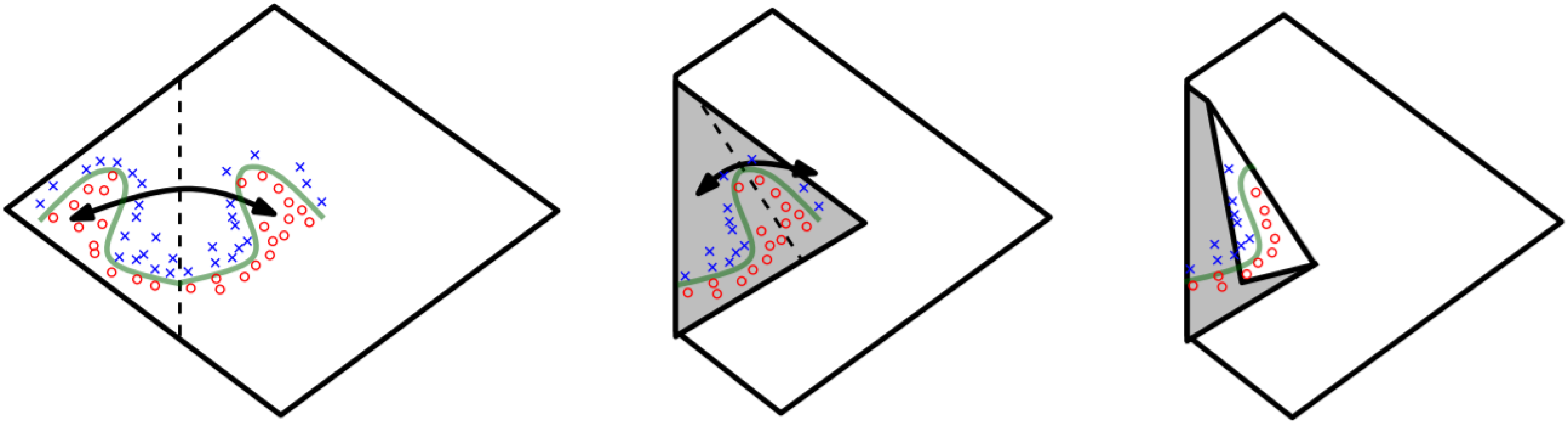


Perceptron

How do we use this to solve the XOR problem?

- What is happening here?





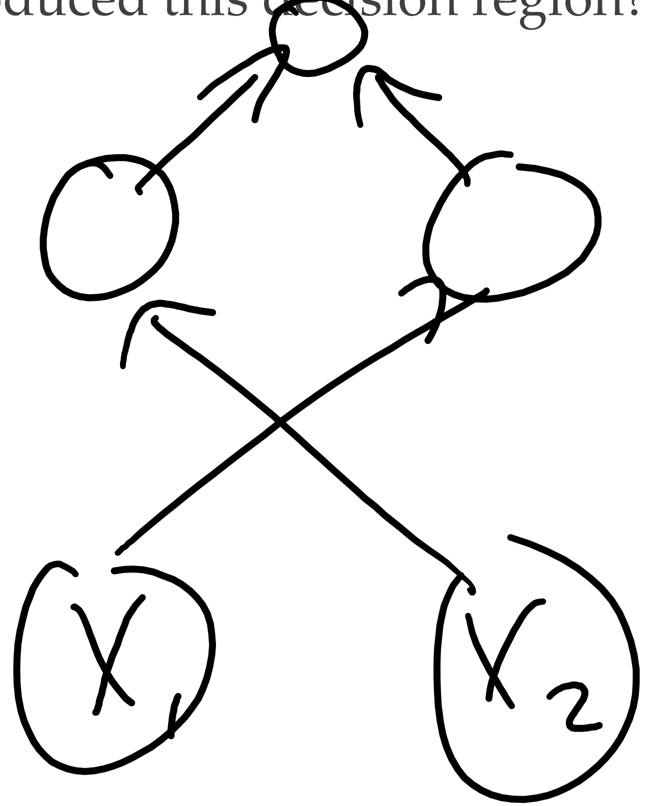
How to train?

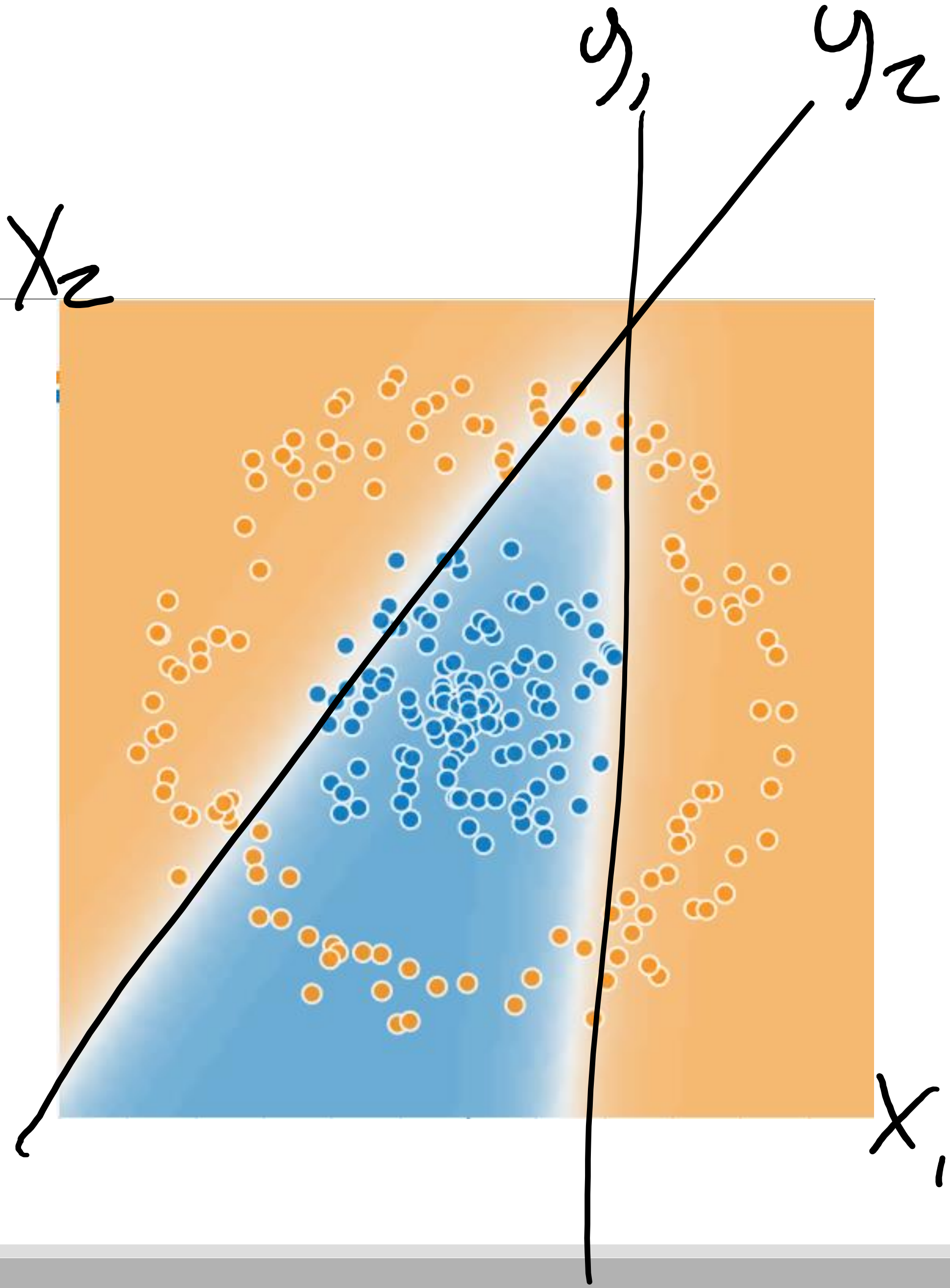
We (hopefully) have some idea of how a particular set of weights causes the neural network to make a decision

- We have some vector of inputs X_d that are all fed as parameters to some number of nodes
- Each of these nodes outputs a sigmoid function to the next hidden layer
- This process eventually leads to the final layer, which makes the final prediction

How to train?

What is the structure of the neural network that produced this decision region?

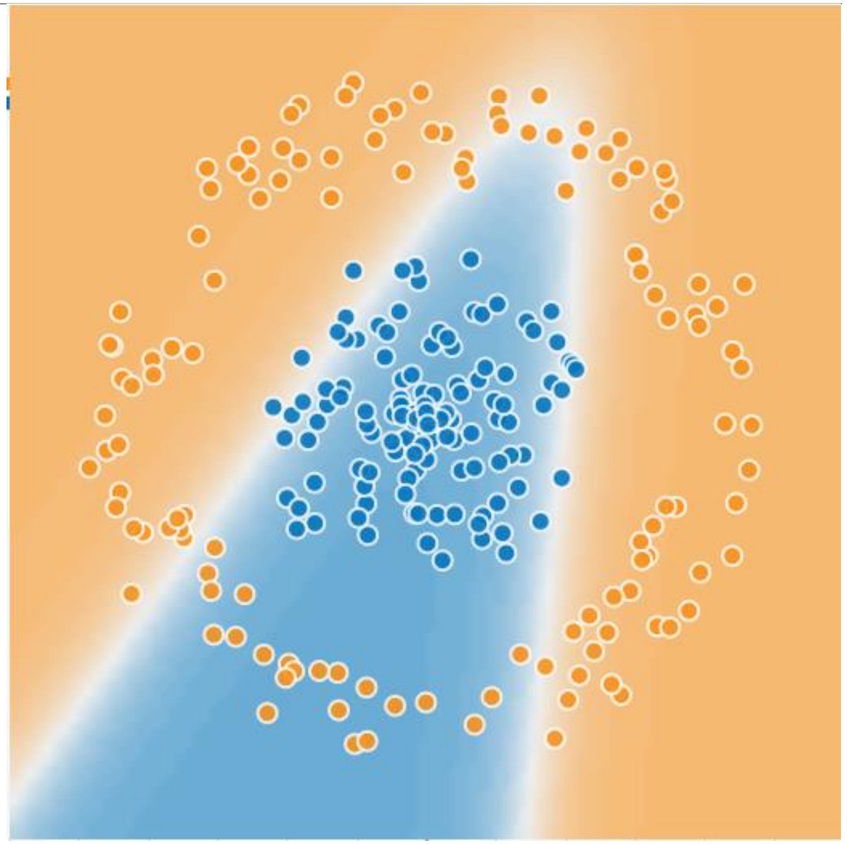




How to train?

What is the structure of the neural network that produced this decision region?

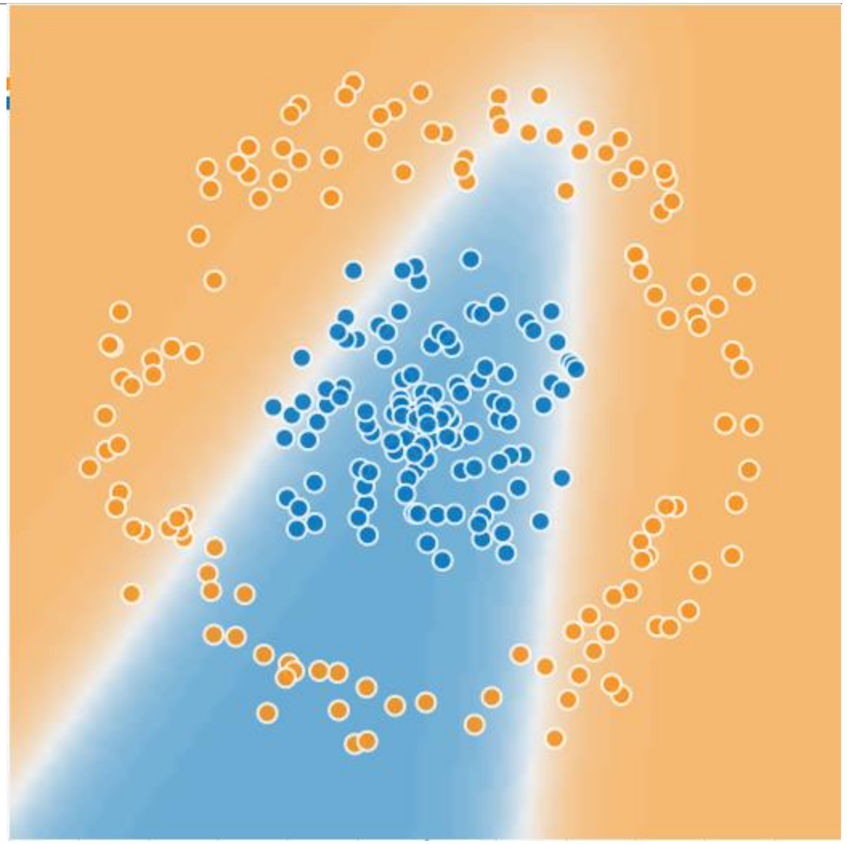
- Blue is positive, orange is negative
- What do the white regions represent?



How to train?

What is the structure of the neural network that produced this decision region?

- Blue is positive, orange is negative
- What do the white regions represent?

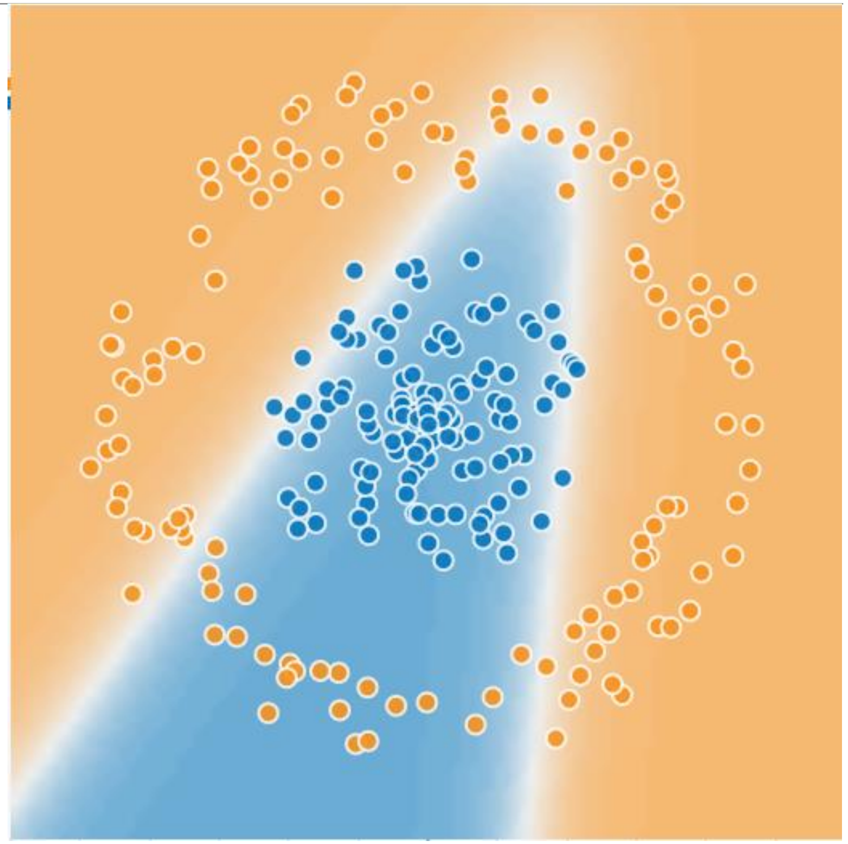


Two lines from two middle “hidden nodes” with sigmoid behavior

What might the weights look like for each of these nodes?

How to train?

Which would help more? Increasing the number of nodes in our hidden layer or increasing the number of hidden layers?



How to train?

With three internal nodes, we can now generate three linear models to separate the data.