Introduction to Neural Nets

Columbia Water Center Neural Network Working Group

James Doss-Gollin Tristan Eisenhart

September 12, 2017

Columbia University

Table of contents

Disclaimer

- I am not an expert in deep learning and none of the following content is original. Mostly it is poached from lecture notes by Peter Belhumeur for his deep learning for computer vision class, but there is also content taken from Friedman et al. [3], Goodfellow et al. [4], and James et al. [5].
- I highly recommend checking out the above sources
- This is not intended to be distributed outside the Columbia Water Center.

What Can Deep Learning Do?

What Is Machine Learning

Difficult to define!

- Vocab: machine learning, statistics, quantitative modeling, deep learning
- ML: more focus on making out-of-sample predictions which minimize some loss function
- ML: In general less focus on inference
- Scale to large data sets (large N) and high dimensions (large p)
- Deep learning is state-of-art ML algorithm for some types of problem

Some Cool Demos

- Dropbox Document Detection
- Text to handwriting
- Image Classification/Captioning
- Bird Identification

All About Classifiers

Linear Regression

We've all seen linear regression: predict value y given data $\mathbf{x}$ with

$$\hat{y} = \theta^T \mathbf{x} \tag{1}$$

Note the notation: θ typically used to refer to parameters. $\mathbf{x}$ includes column of 1's for the intercept.

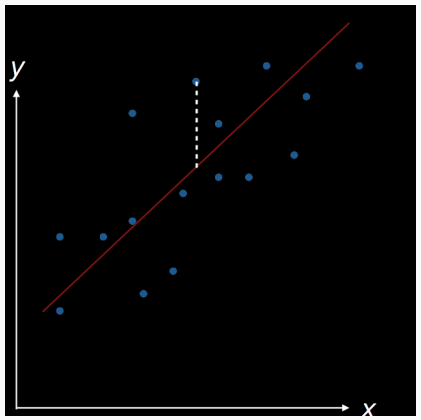


Figure 1: Linear Regression

Classification

Take data and predict 1+ categories

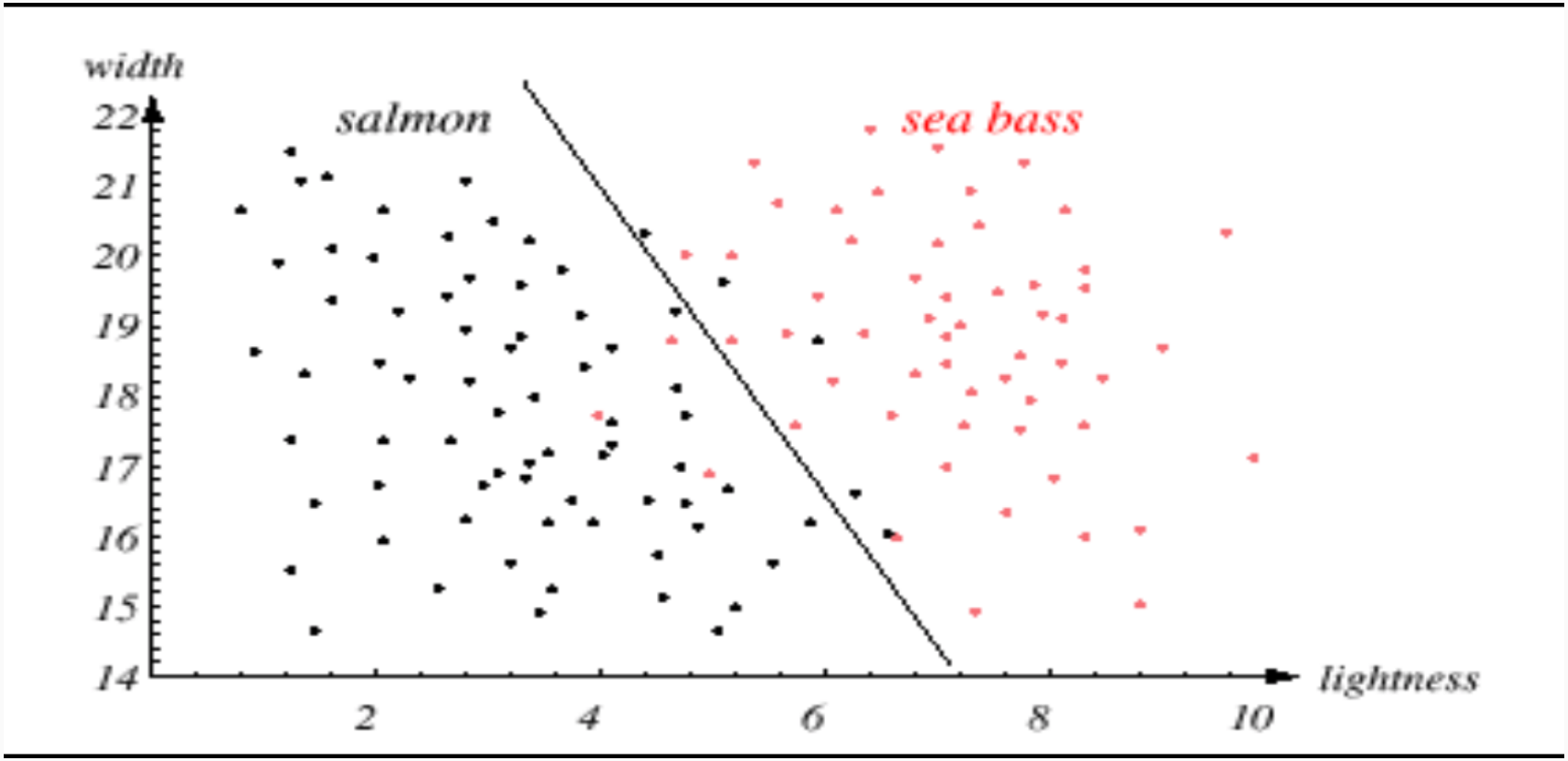


Figure 2: A Scatter Plot

Conditional Probability

$$p(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(A)P(B|A)}{P(B)} \tag{2}$$

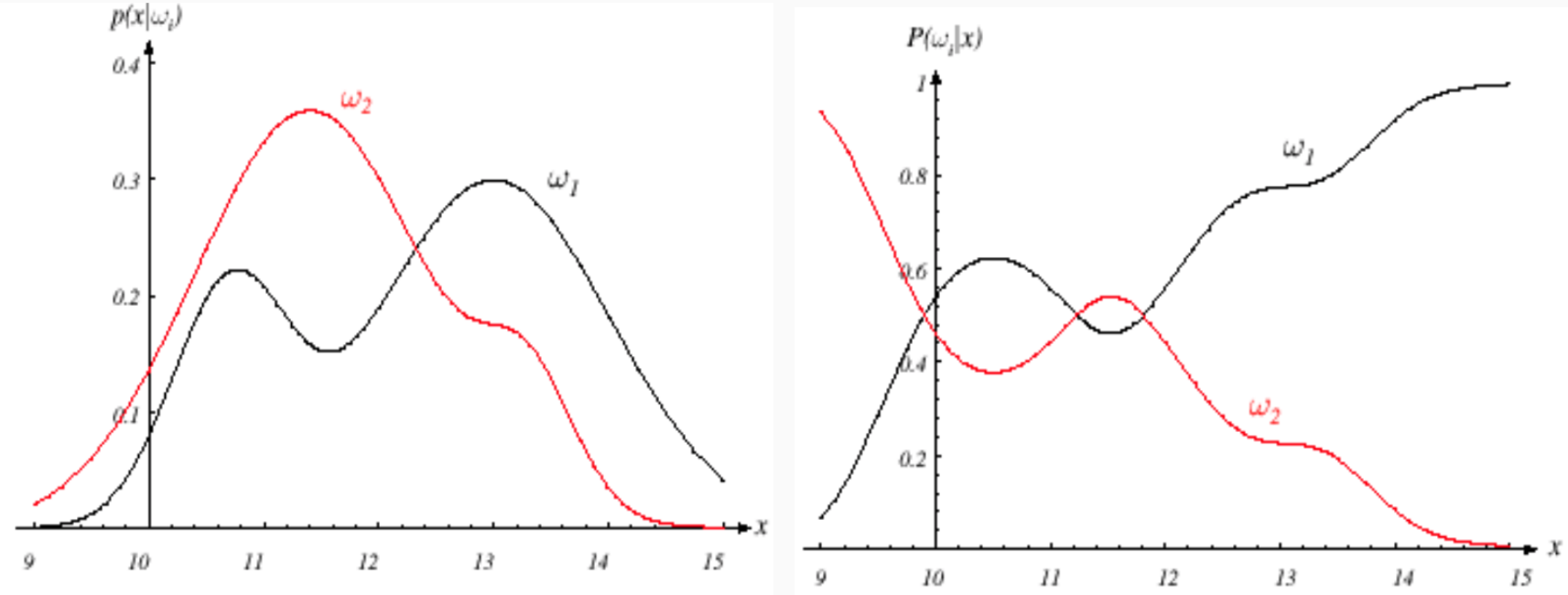


Figure 3: A naive fish classifier: (L) PDF (R) Posterior

Mathematical Framing

Let's think of a classifier as set of scalar functions $g_i(\mathbf{x})$ one for each class i that assigns a score to the vector of feature values $\mathbf{x}$ and then choses the class i with the highest score. In other words:

Choose class i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \forall j, i \neq j \tag{3}$$

This requires a good estimate of

$$P\left(\omega_i \middle| \mathbf{x}\right)$$

Curse of Dimensionality

- If we have 3 dimensional feature space $\mathbf{x} \in [0, 1]^3$
- We have 10000 training samples
- If we try to estimate our probability using a 3D histogram with 100 bins in each direction
- Each bin has (on average) 0.01 samples!

Thus we never have enough samples to sample high-dimensional feature space; Focus on finding decision surface which separates categories

Many Possible Classifiers

With a classifier, what is our goal?

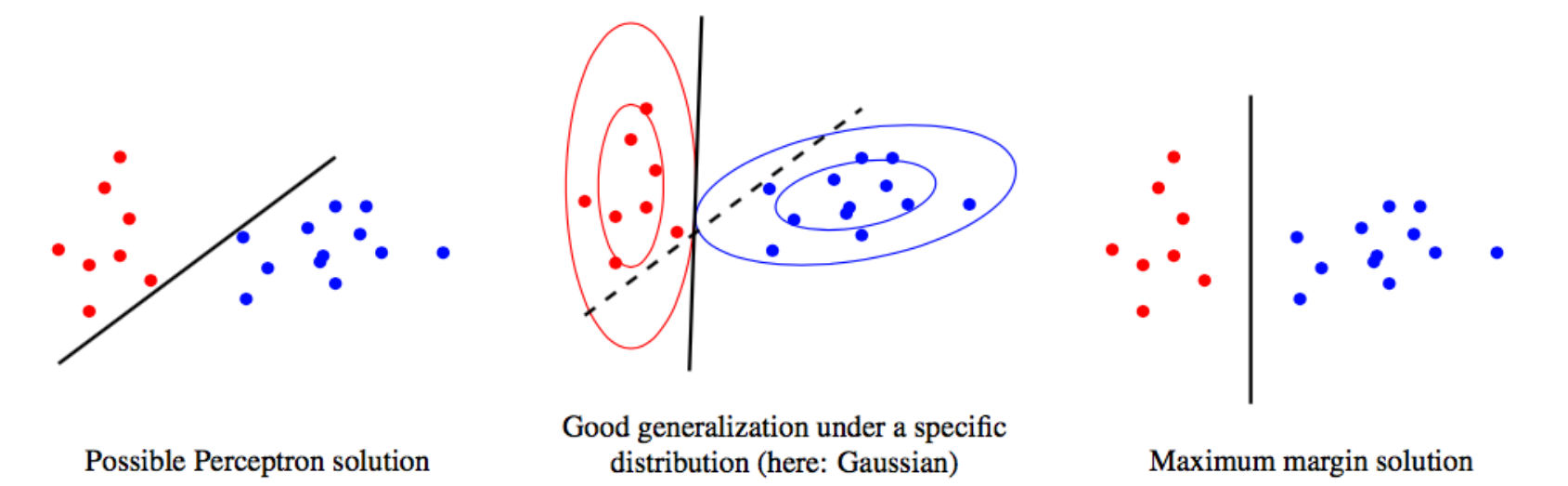


Figure 4: Maximum Margin tends to do well

Support vector machine implements maximum-margin classifier – skipping details for concision

Linear Classifier

Standard framing: training data $(\mathbf{x}_i, y_i)$ with $i = 1, \ldots, N$ and $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. Learn a linear classifier $f(\mathbf{x}_i)$:

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y = +1 \\ < 0 & y = -1 \end{cases} \tag{4}$$

and since we have a linear classifier, $f(\mathbf{x}) = \theta^T \mathbf{x}$

Logistic (Sigmoid) Classifier

- Extend linear classifier $f(\mathbf{x}) = \sigma^T \mathbf{x}$ to $\sigma(f(\mathbf{x})) = \frac{1}{1+e^{-f(x)}}$
- Predict $+1$ if $\sigma \geq 0.5$, -1 if $\sigma < 0.5$

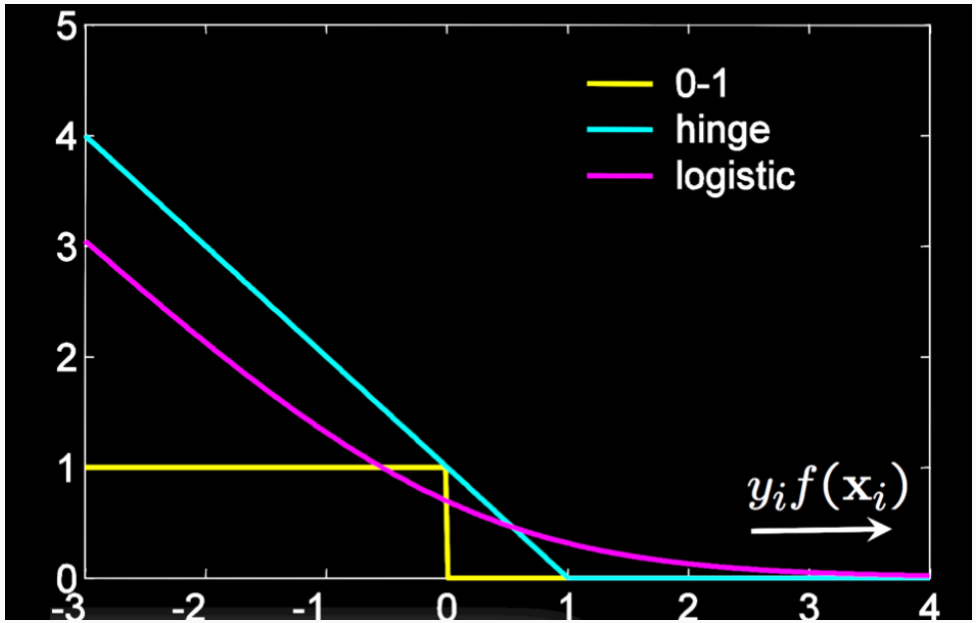


Figure 5: Comparison of loss functions

Maximum Likelihood Estimation

Maximizing the likelihood is the same as minimizing negative log-likelihood

$$P\left(y\middle|\mathbf{x},\theta\right)=\prod_{i=1}^{N}\frac{1}{1+\exp\left(-y_i f\left(\mathbf{x_i}\right)\right)} \tag{5}$$

$$L\left(y\middle|\mathbf{x},\theta\right)=\sum_{i=1}^{N}\log\left[1+\exp\left(-y_i f\left(\mathbf{x_i}\right)\right)\right] \tag{6}$$

Computational Graph

For this figure we have renamed θ to be w and added a regularization term $\frac{\lambda}{2}||\theta||^2$ to the cost function

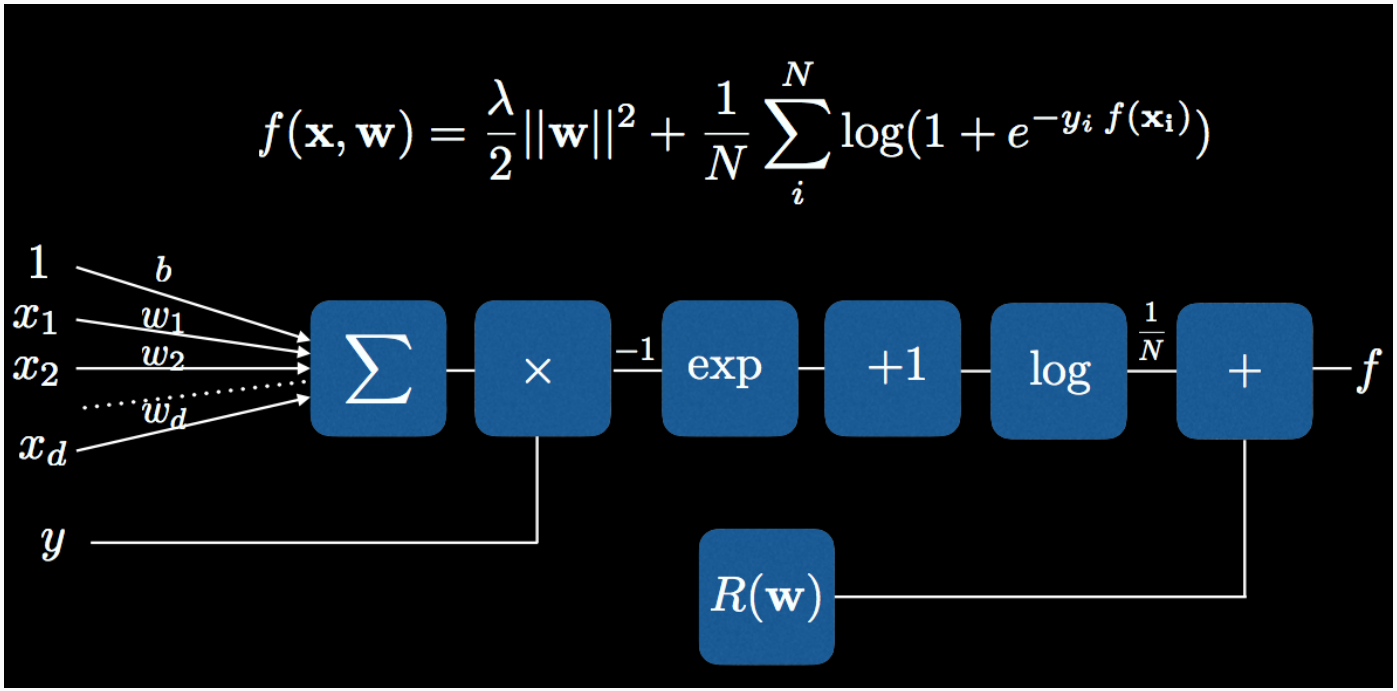


Figure 6: Computational graph for logistic regression

Chain Rule

Given a training sample **x** lets compute the gradient of f with respect to the weights θ.

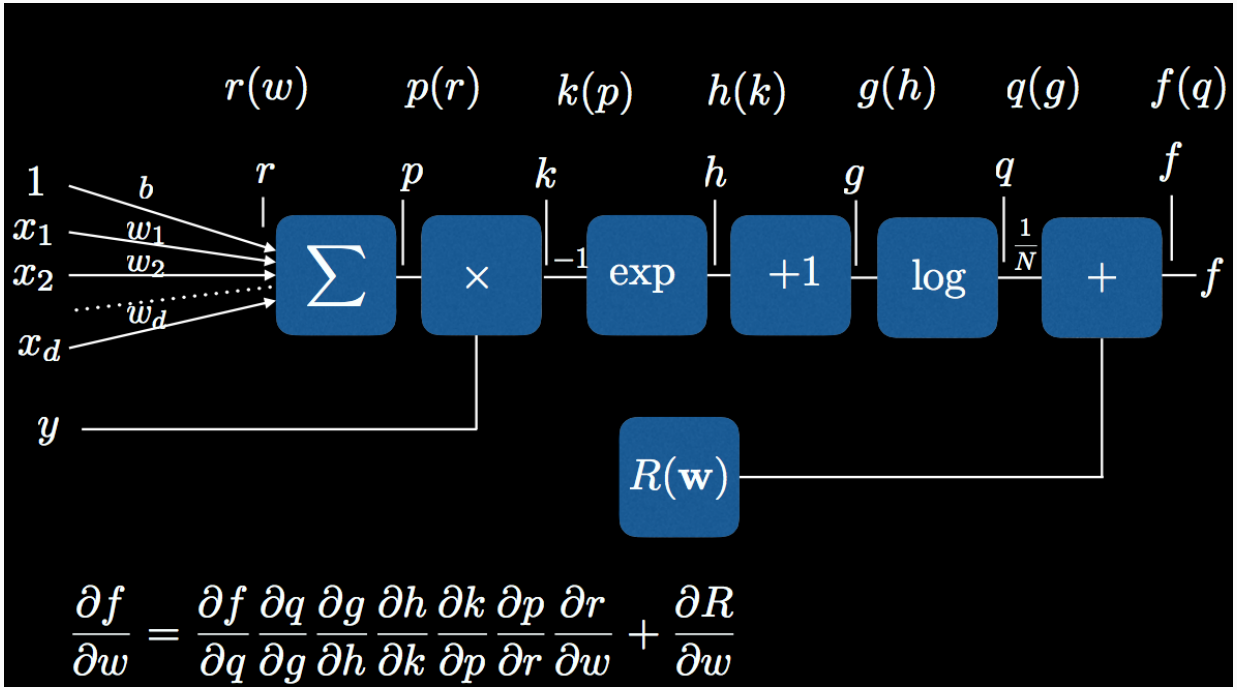


Figure 7: Chain rule for logistic regression example

Gradient Descent

(There are more complicated ways to do this). Take our initial weights, choose a learning rate $\eta = 0.1$ say:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} f(\theta_t) \tag{7}$$

Neural Networks

Really?

James, are we going to show us some grainy pictures to illustrate the chain rule, and then claim we understand neural networks?

Really?

James, are we going to show us some grainy pictures to illustrate the chain rule, and then claim we understand neural networks?

yep (and not just because I'm writing this at midnight and want to sleep!)

Feedforward Network

We just need to generalize what we already learned.

- Let $y = f^*(\mathbf{x})$ be some function we don't know but want to approximate given data – could be a classifier boundary!
- Try to learn parameters θ to approximate this

A feedforward network has *no* feedback – layers are composed of functions ("layers") so that (e.g.)

$$f(\mathbf{x}) = l^3(l^2(l^1(\mathbf{x}))) \tag{8}$$

The network is thus mapping x to a feature space $\phi(\mathbf{x})$ in which (ideally...) the samples are linearly separable!

XOR 1

Let's see how this works. Take a problem where a linear classifier is useless:

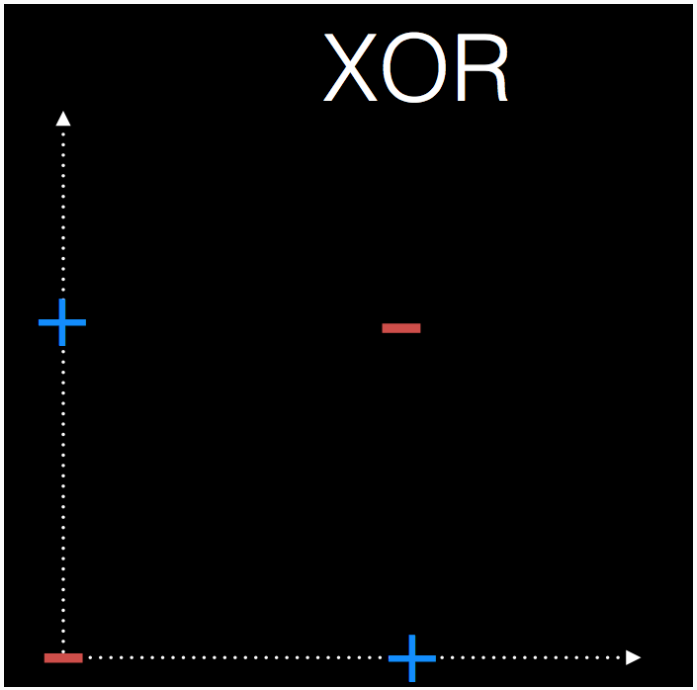


Figure 8: The XOR Classification Problem

XOR 2

Now let's add a single "hidden" layer:

$$\mathbf{x} \rightarrow^{w_1} \mathbf{h} \rightarrow^{w_2} y \tag{9}$$

$$h = g(w_1^T \mathbf{x} + \mathbf{c}) \tag{10}$$

$$y = w_2^T \mathbf{h} + \mathbf{b} \tag{11}$$

$$= w_2^T g(w_1^T \mathbf{x} + \mathbf{v}) + b \tag{12}$$

Now let's try setting $g(z) = \max(0, z)$ "ReLU Function"

XOR 2

Now let's add a single "hidden" layer:

$$\mathbf{x} \rightarrow^{w_1} \mathbf{h} \rightarrow^{w_2} y \tag{9}$$

$$h = g(w_1^T \mathbf{x} + \mathbf{c}) \tag{10}$$

$$y = w_2^T \mathbf{h} + \mathbf{b} \tag{11}$$

$$= w_2^T g(w_1^T \mathbf{x} + \mathbf{v}) + b \tag{12}$$

XOR 2

Now let's add a single "hidden" layer:

$$\mathbf{x} \rightarrow^{w_1} \mathbf{h} \rightarrow^{w_2} y \tag{9}$$

$$h = g(w_1^T \mathbf{x} + \mathbf{c}) \tag{10}$$

$$y = w_2^T \mathbf{h} + \mathbf{b} \tag{11}$$

$$= w_2^T g(w_1^T \mathbf{x} + \mathbf{v}) + b \tag{12}$$

Now let's try setting $g(z) = \max(0, z)$ "ReLU Function"

XOR 3

Now let's go back to the XOR problem! Let's let $w_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $w_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, $b = 0$ and our samples $X = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T$

XOR 4

$$w_1^T \mathbf{x} + \mathbf{c} = \begin{bmatrix} 0 & 1 & 1 & 2 \\ -1 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$h = \max\left(0, w_1^T \mathbf{x} + \mathbf{c}\right) \tag{14}$$

$$= \begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$y = w_2^T h + b \tag{16}$$

$$= [0110] \tag{17}$$

XOR Finally Done!

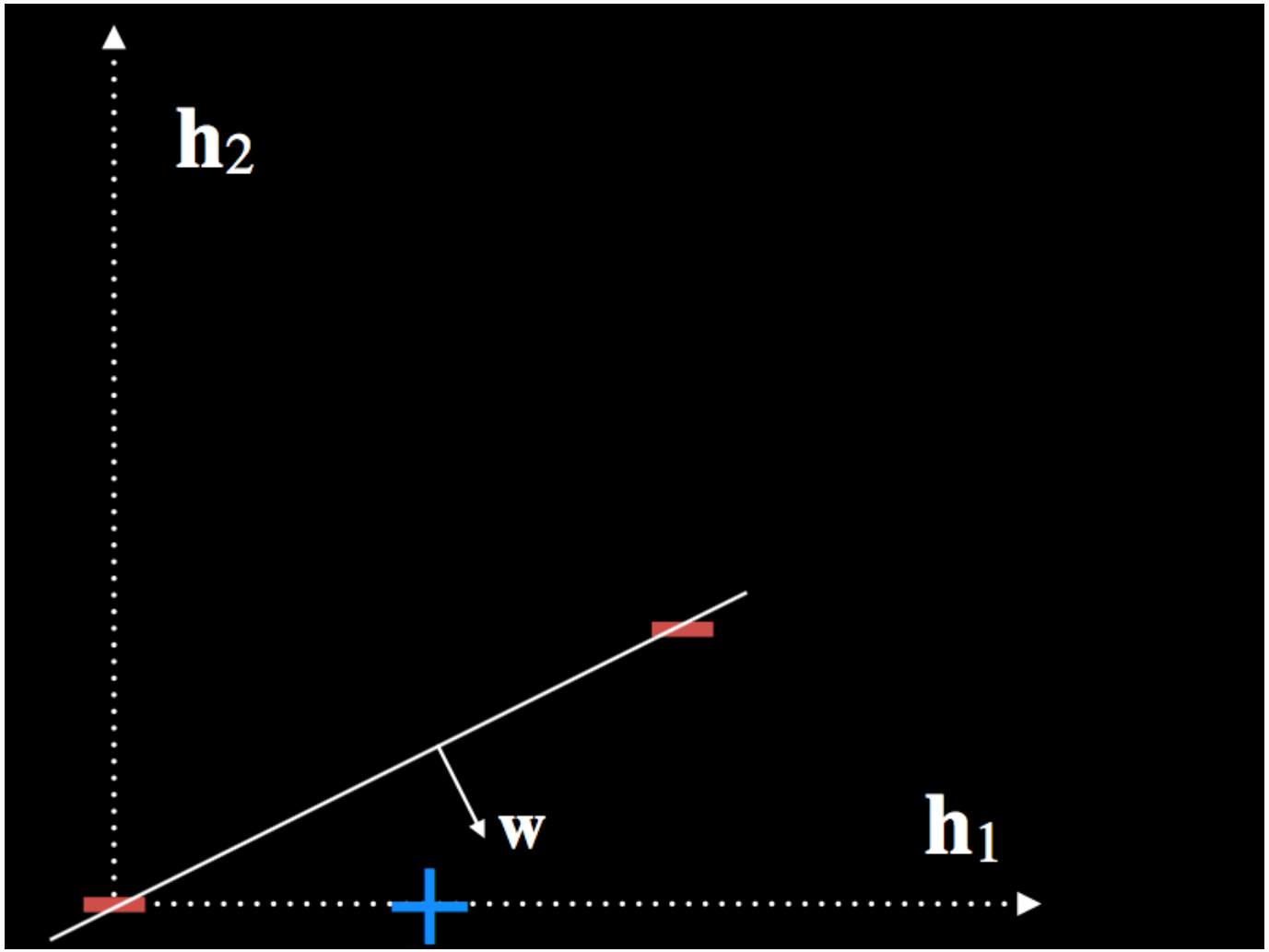


Figure 9: So it worked! $h = \phi(\mathbf{x})$

Training with Back-Propagation

1. Create noisy XOR data
2. Choose loss function for binary classifier (sigmoid is good – or Softplus)

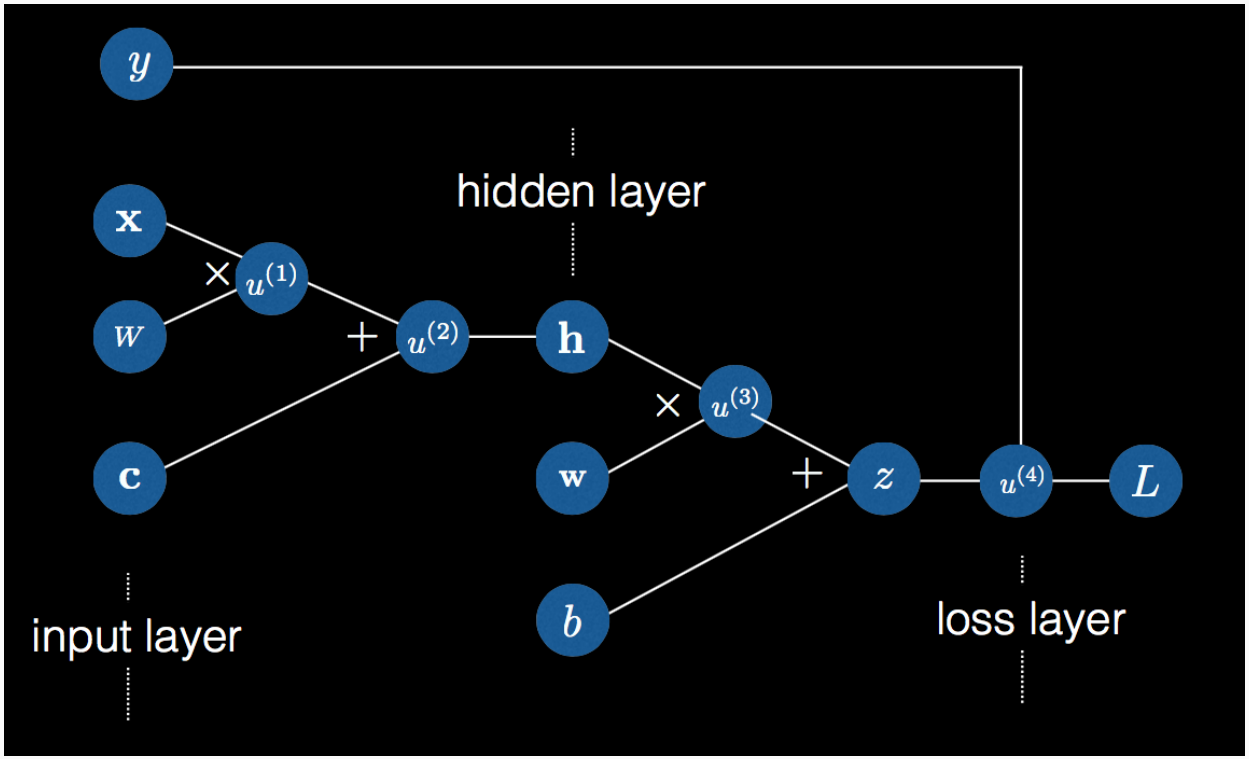


Figure 10: Computational graph with a hidden layer

Challenges in Machine Learning

Over-fitting

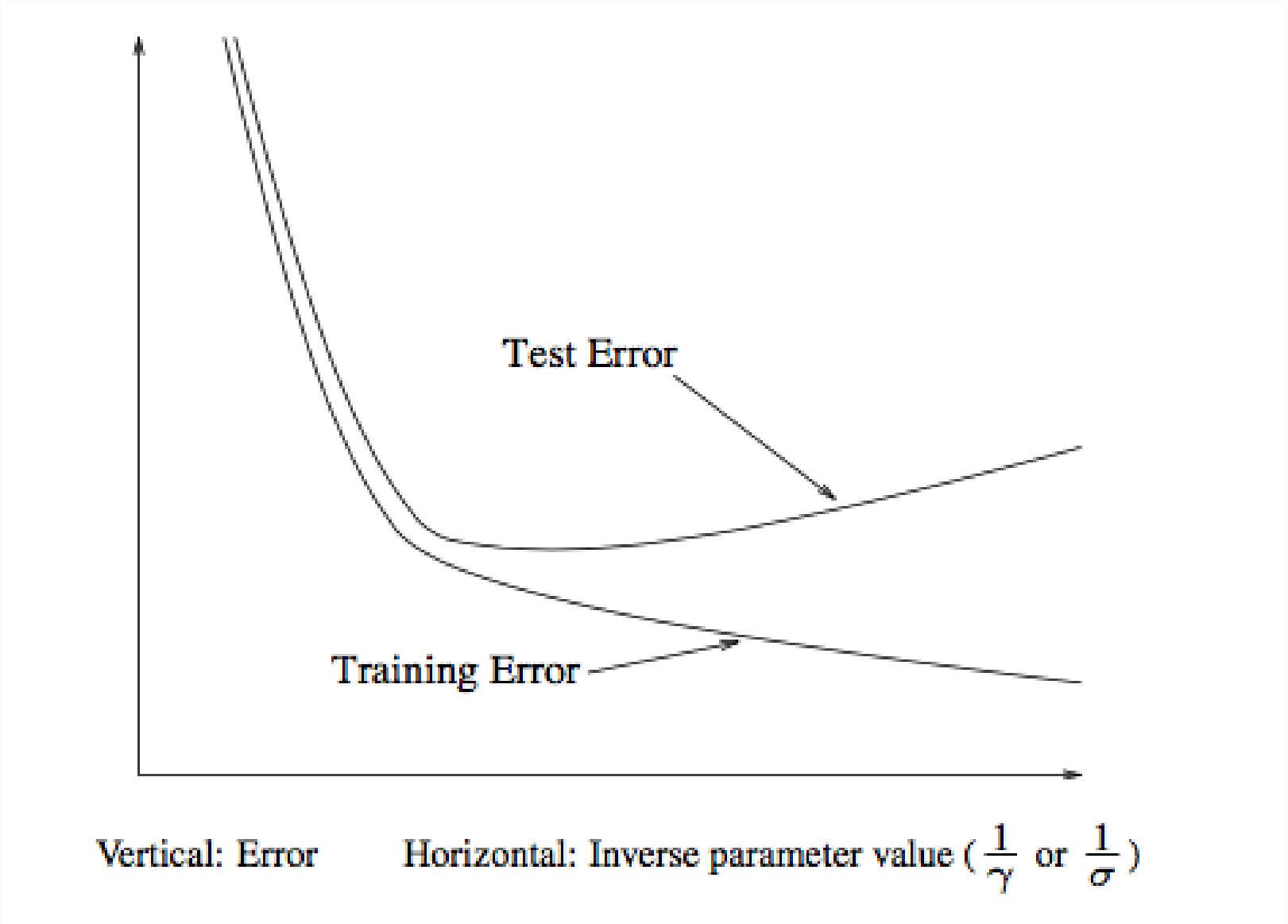
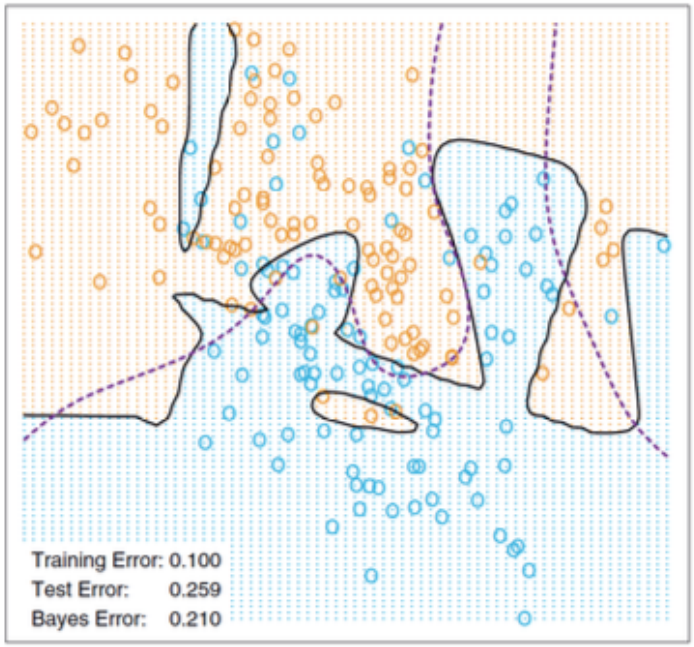


Figure 11: Over-fitting can be a huge challenge!

Over-fitting Example

Overfitting is best illustrated with a *nonlinear* classifier.



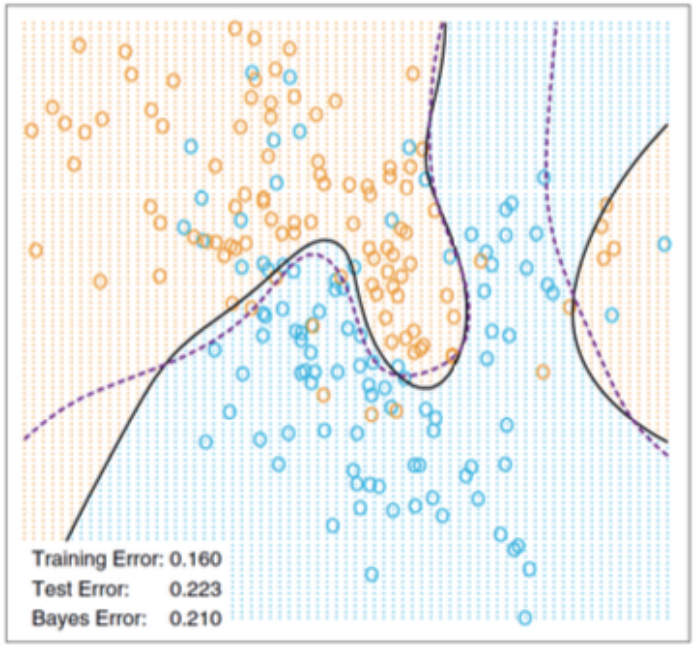


Figure 12: An example of overfit data

Over-fitting Example2

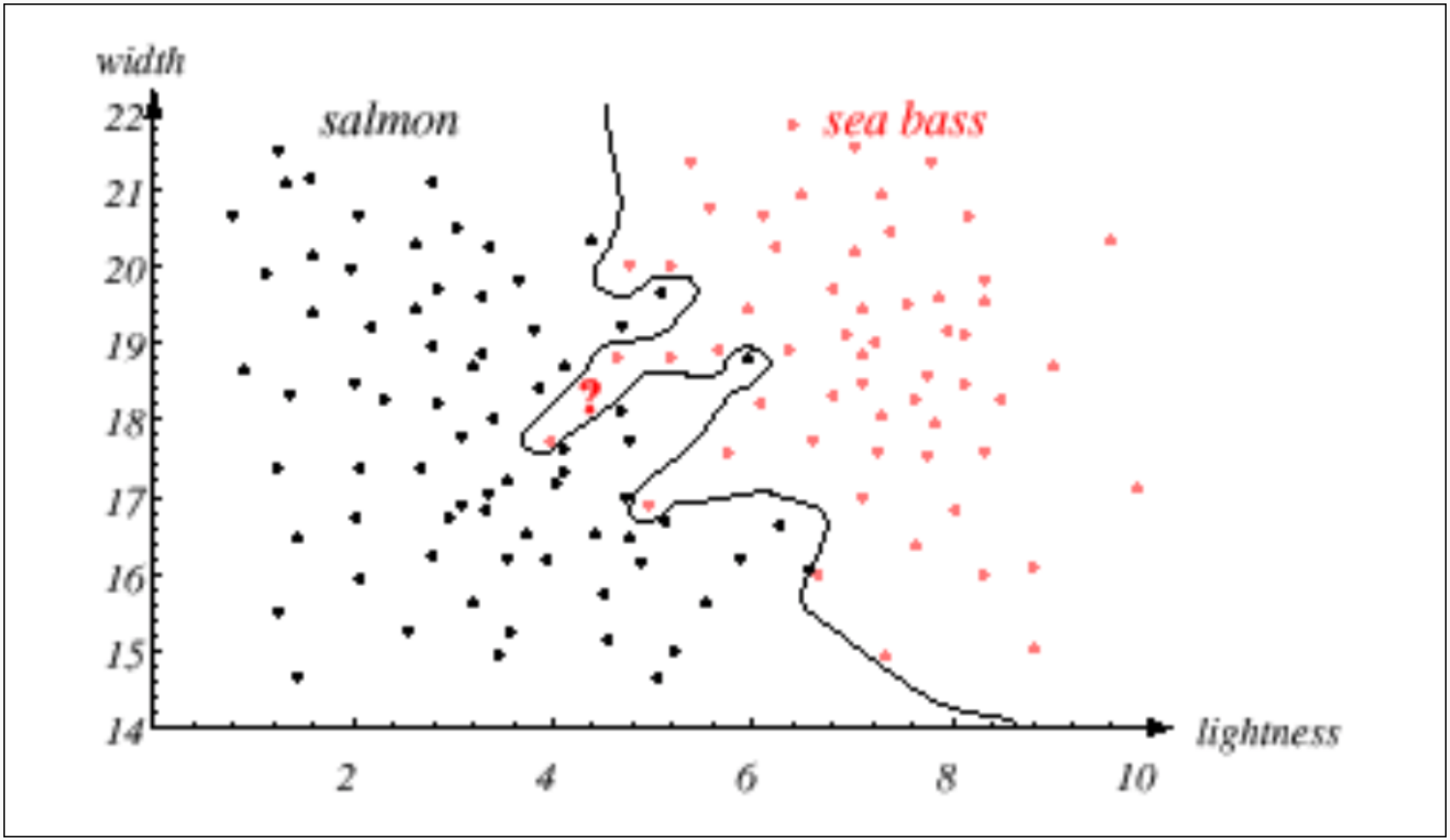


Figure 13: Another example of overfit data

Also there are

- Geometric invariants for rigid transformations of 3-D objects viewed under perspective projective projection do not exist[1]
- Illumination invariants for 3D ojbects to not exist [2]

Regularization

The goal of regularization is to prevent overfitting the training data with the hope that this improves generalization, i.e., the ability to correctly handle data that the network has not trained on.

Go to Peter Belhumeur's slides on regularization (begin p.365 of his course notes!)

References

[1] J B Burns et al. "View variation of point-set and line-segment features". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.1 (1993).

[2] H F Chen et al. "In search of illumination invariants". *IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000.* IEEE Comput. Soc.

[3] Jerome Friedman et al. *The Elements of Statistical Learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[4] Ian Goodfellow et al. *Deep Learning*. MIT Press, 2016.

[5] Gareth James et al. *An Introduction to Statistical Learning*. Vol. 103. Springer Texts in Statistics. New York, NY: Springer New York, 2013.

Some More Resources

As you check tutorials, be aware that there are subtle differences between python 2 and python 3 (i.e. `print`). There are big differences between different versions of `tensorflow`. Using a custom conda environment is a great way to manage versions – read on your own :). Some other useful resources:

- DLCV course slides will be posted to Slack
- DLCV course website
- Tensorflow tutorial
- TF tutorial by Karlijn Willems
- Stamford Deep Learning Class CS231 Blog post: why understanding back-propagation matters

Thanks!