Computer Vision: from Recognition to Geometry

Lecture 5: Image Representation for Visual Classification

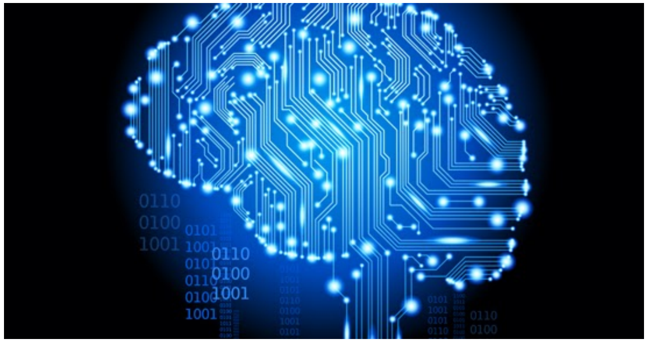
Yu-Chiang Frank Wang 王鈺強

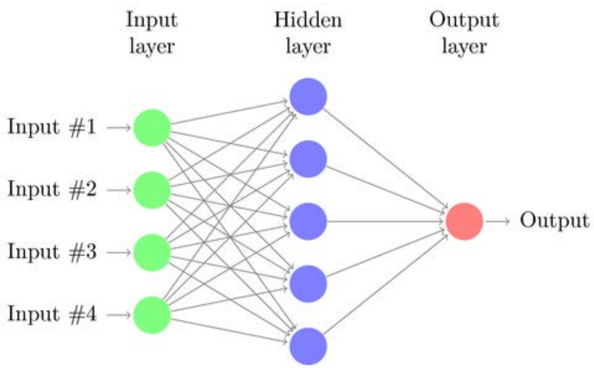
Dept. Electrical Engineering, National Taiwan University

2019/10/2

What’s to Be Covered Today…

- Unsupervised vs. Supervised Learning
  - Clustering
  - Unsup. vs. Sup. Dimension Reduction
  - Training, testing, & validation
- Image Representation
  - Bag-of-Words Representation
  - Linear Classification
  - Intro to Neural Networks





Eigenanalysis & PCA (cont'd)

- A d x d covariance matrix contains a maximum of d eigenvector/eigenvalue pairs.
  - Assuming you have N images of size M x M pixels, we have dimension d = M^2.
  - With the rank of ∑ as , we have at most non-zero eigenvalues.
  - How dimension reduction is realized? how to reconstruct the input data?

- Expanding a signal via eigenvectors as bases
  - With symmetric matrices (e.g., covariance matrix), eigenvectors are orthogonal.
  - They can be regarded as unit basis vectors to span any instance in the d-dim space.

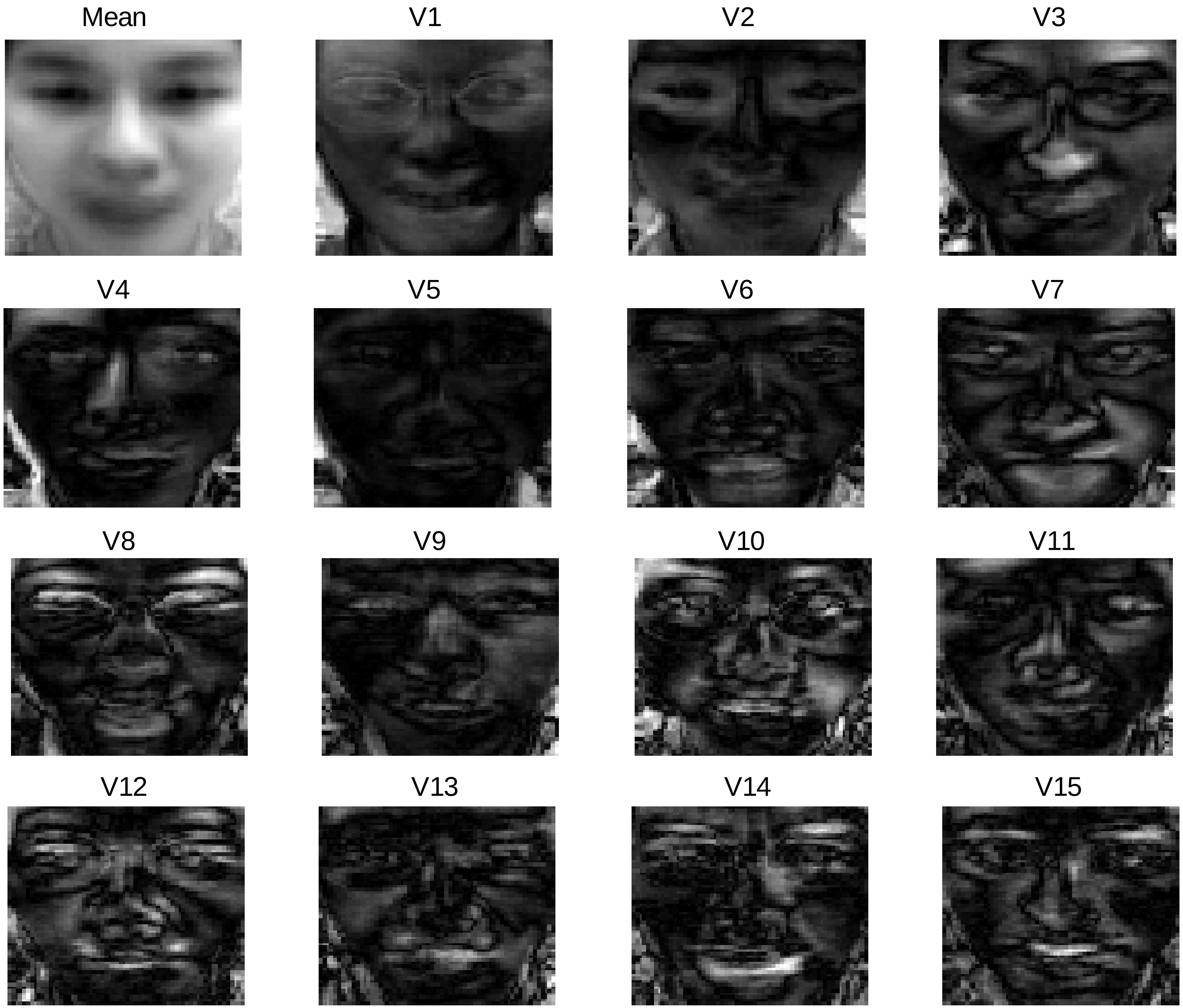
Practical Issues in PCA

- Assume we have N = 100 images of size 200 x 200 pixels (i.e., d = 40000).
- What is the size of the covariance matrix? What's its rank?
- What can we do? Gram Matrix Trick!

Let's See an Example (CMU AMP Face Database)

- Let's take 5 face images x 13 people = 65 images, each is of size 64 x 64 = 4096 pixels.
- # of eigenvectors are expected to use for perfectly reconstructing the input = 64.
- Let's check it out!

What Do the Eigenvectors/Eigenfaces Look Like?



All 64 Eigenvectors, do we need them all?

Use only 1 eigenvector, MSE = 1233

Use 2 eigenvectors, MSE = 1027

MSE=1027.63

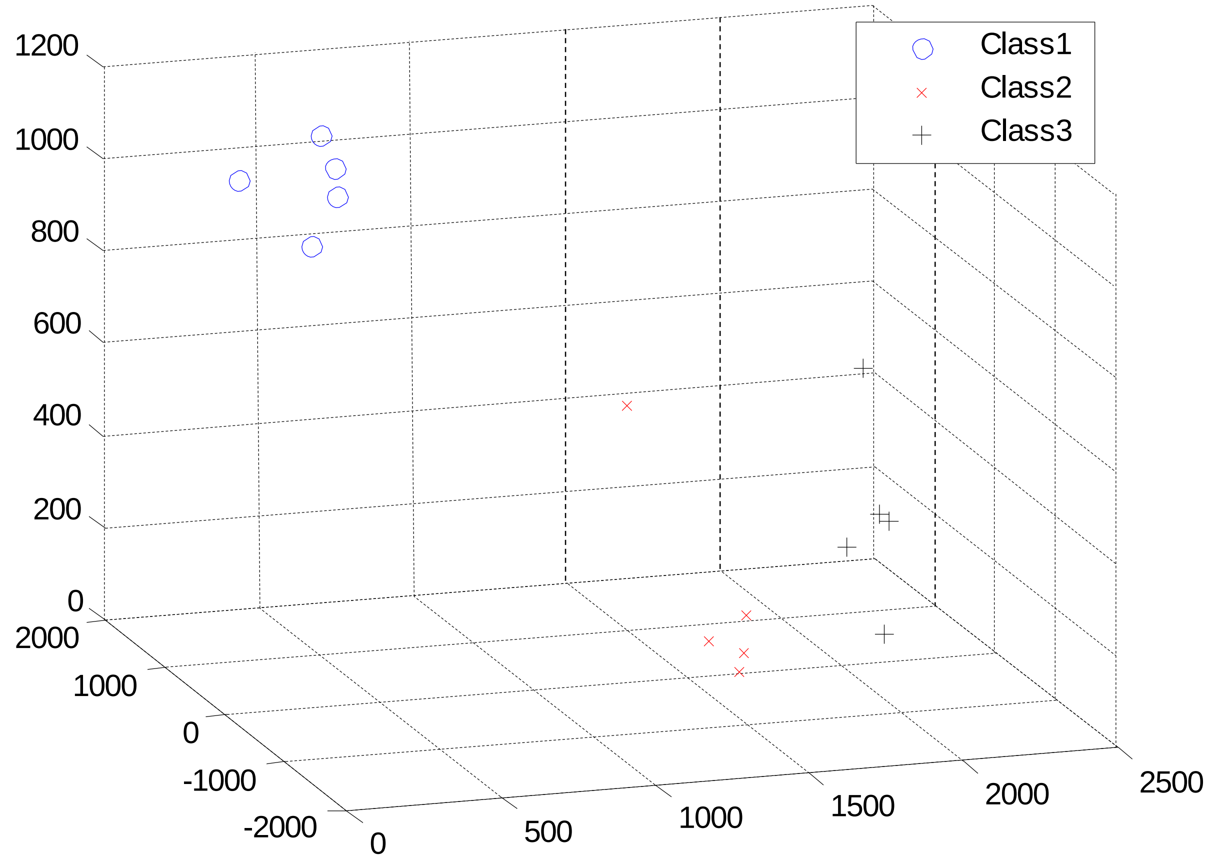
Use 3 eigenvectors, MSE = 758

All 64 eigenvectors, MSE = 0

MSE=0.00

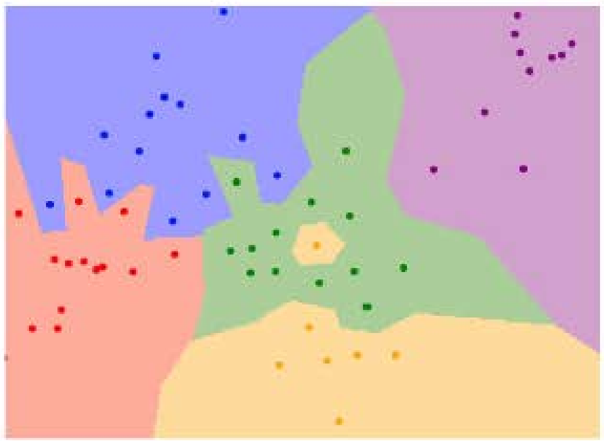
Final Remarks

- Linear & unsupervised dimension reduction
- PCA can be applied as a feature extraction/preprocessing technique.
  - E.g,, Use the top 3 eigenvectors to project data into a 3D space for classification.

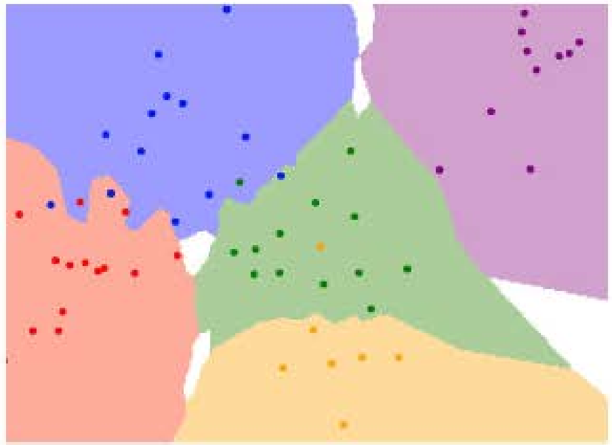


Final Remarks (cont'd)

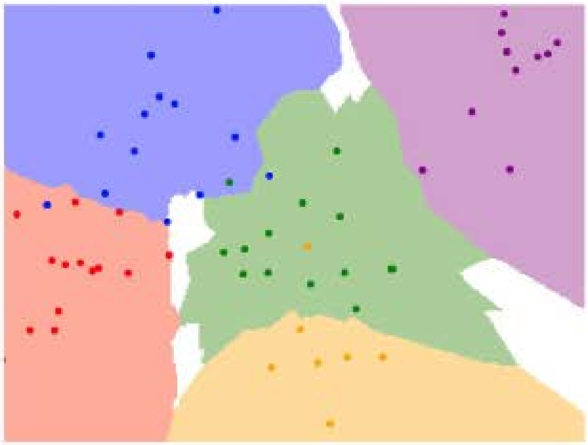
- How do we classify? For example…
  - Given a test face input, project into the same 3D space (by the same 3 eigenvectors).
  - The resulting vector in the 3D space is the feature for this test input.
  - We can do a simple Nearest Neighbor (NN) classification with Euclidean distance, which calculates the distance to all the projected training data in this space.
  - If NN, then the label of the closest training instance determines the classification output.
  - If k-nearest neighbors (k-NN), then k-nearest neighbors need to vote for the decision.



k = 1



k = 3



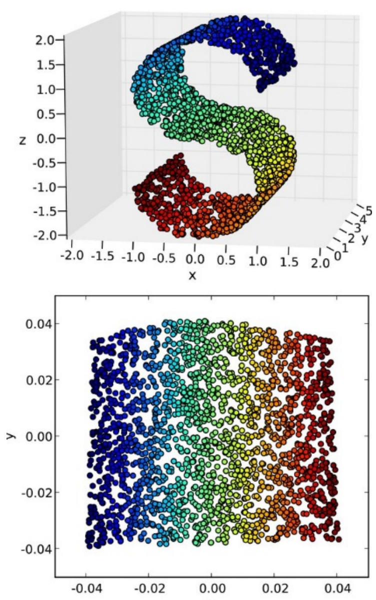
k = 5

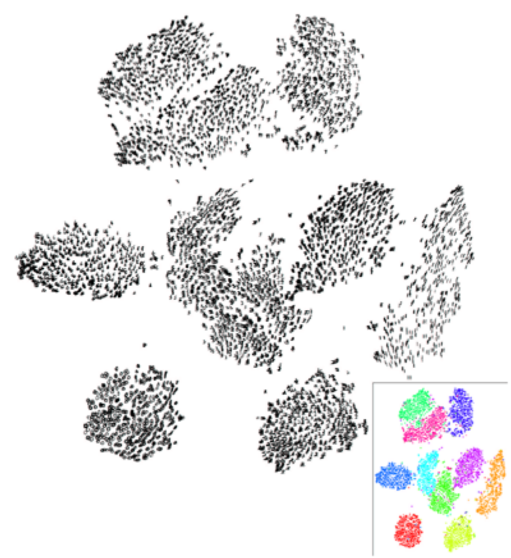
Demo available at http://vision.stanford.edu/teaching/cs231n-demos/knn/

Image credit: Stanford CS231n

Final Remarks (cont'd)

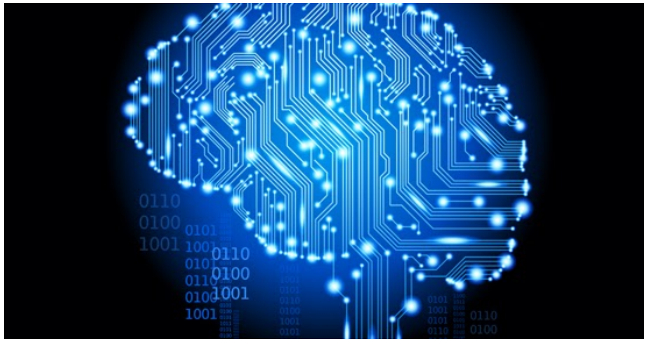
- If labels for each data is provided → Linear Discriminant Analysis (LDA)
  - LDA is also known as Fisher's discriminant analysis.
  - Eigenface vs. Fisherface (IEEE Trans. PAMI 1997)
- If linear DR is not sufficient, and non-linear DR is of interest…
  - Isomap, locally linear embedding (LLE), etc.
  - **t-distributed stochastic neighbor embedding (t-SNE)** (by G. Hinton & L. van der Maaten)

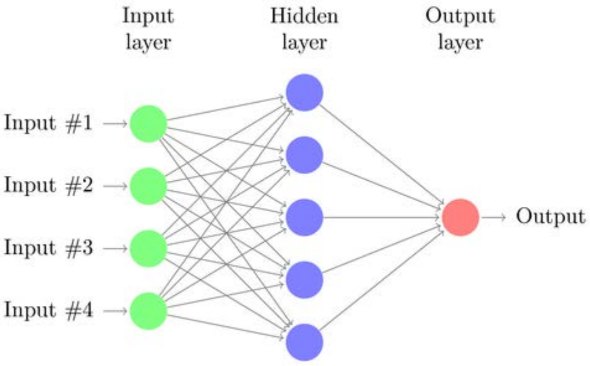




What’s to Be Covered Today…

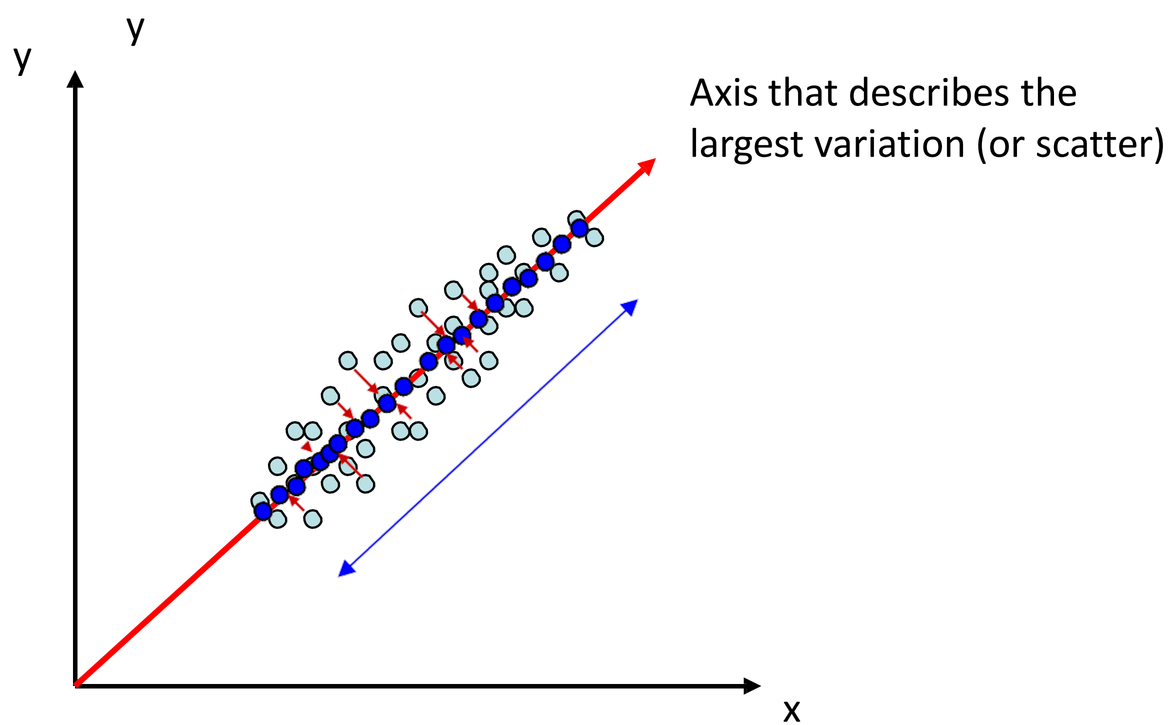
- Unsupervised vs. Supervised Learning
  - Clustering
  - Unsup. vs. Sup. Dimension Reduction
  - Training, testing, & validation
- Image Representation
  - Bag-of-Words Representation
  - Linear Classification
  - Intro to Neural Networks





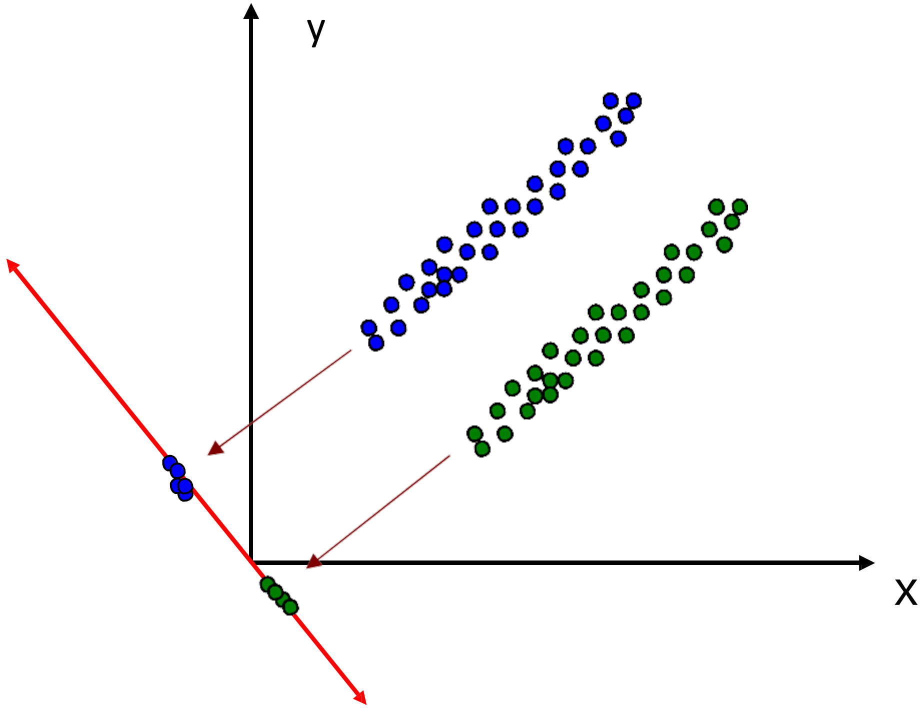
What is PCA?
What are we trying to do?

- We want to find projections of data (i.e., direction vectors that we can project the data on to) that describe the maximum variation.



What is LDA?
What are we trying to do?

- We want to find projections that separate the classes with the assumption of unimodal Gaussian modes.
- That is, to max. distance between two means while min. the variances
- =>will lead to minimize overall probability of error



Case 1: A simple 2-class problem

- We want to maximize the distance between the projected means:

  e.g., maximize $\left|\tilde{\mu}_1 - \tilde{\mu}_2\right|^2$

Between Class Scatter Matrix S_B

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T\mu_1 - w^T\mu_2)^2$$

$$= w^T(\mu_1 - \mu_2) \quad (\mu_1 - \mu_2)^T w$$

$$= w^T S_B w$$

We want to maximize $w^T S_B w$ where S_B is the between class scatter matrix defined as:

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

NOTE: S_B is rank 1. This will be useful later on to find closed form solution for 2-class LDA

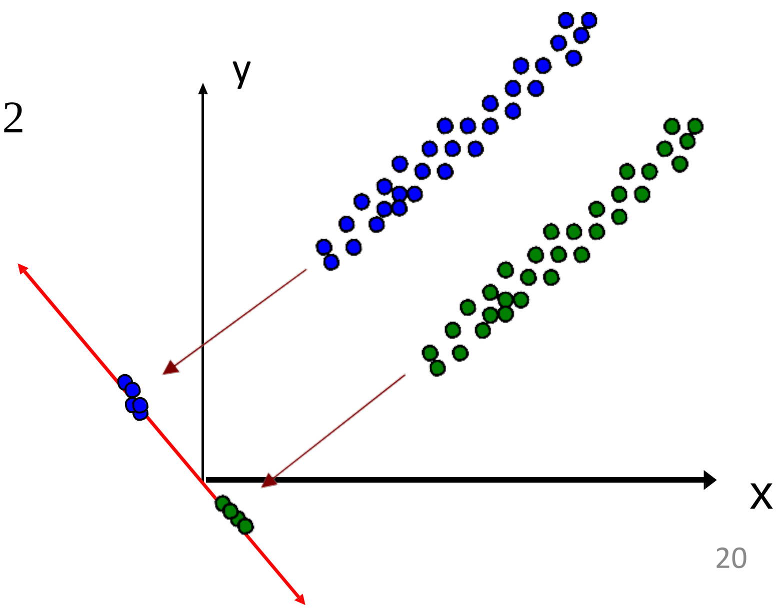
We also want to minimize….

- The variance or scatter of the projected samples from each class (i.e. we want to make each class more compact or closer to its mean). The scatter from class 1 defined as s_1 is given as

$$\tilde{s}_1^{\,2} = \sum_{i=1}^{N_1} (\tilde{x}_i - \tilde{\mu}_1)^2$$

- Thus we want to minimize the scatter of class 1 and class 2 in projected space, i.e.

minimize the total scatter $\tilde{s}_1^{\,2} + \tilde{s}_2^{\,2}$



Fisher Linear Discriminant Criterion Function

- Objective #1: We want to ***maximize*** the between class scatter:

$$\left|(\tilde{\mu}_1 - \tilde{\mu}_2)\right|^2$$

- Objective #2: We want to ***minimize*** the within-class scatter.

$$\tilde{s}_1^{\,2} + \tilde{s}_2^{\,2}$$

- Thus we define our objective function J(w) as the following ratio that we want to ***maximize*** in order to achieve the above objectives:

LDA

- Thus we want to find the vector **w** that maximizes J(**w**).
- Let's expand on scatter s_1 & s_2.

$$\tilde{s}_1{}^2 = \sum_{i=1}^{N_1} (\tilde{x}_i - \tilde{\mu}_1)^2 \qquad \tilde{s}_2{}^2 = \sum_{i=1}^{N_2} (\tilde{x}_i - \tilde{\mu}_2)^2$$

$$= \sum_{i=1}^{N_1} (w^T x_i - w^T \mu_1)^2 \qquad = \sum_{i=1}^{N_2} (w^T x_i - w^T \mu_2)^2$$

$$= \sum_{i=1}^{N_1} w^T (x_i - \mu_1)(x_i - \mu_1)^T w \qquad = \sum_{i=1}^{N_2} w^T (x_i - \mu_2)(x_i - \mu_2)^T w$$

$$= w^T S_1 w \qquad = w^T S_2 w$$

Total Within-Class Scatter Matrix

- We want to minimize total within-class scatter. i.e.

$$\tilde{s}_1^{\,2} + \tilde{s}_2^{\,2}$$

- This is equivalent to minimize $\mathbf{w}^T\mathbf{S}_w\mathbf{w}$

Solving LDA

- Maximize $$J(\mathrm{w}) = \frac{\mathrm{w}^{\mathrm{T}} \mathrm{S}_{\mathrm{B}} \mathrm{w}}{\mathrm{w}^{\mathrm{T}} \mathrm{S}_{\mathrm{W}} \mathrm{w}}$$
- We need to find the optimal **w** which will maximize the above ratio.
- What do we do now?

Some calculus….

LDA derivation

$$\mathrm{S_B w} - J(\mathrm{w})\mathrm{S_W w} = 0$$

$$\mathrm{S_B w} - \lambda \mathrm{S_W w} = 0$$

$$\mathrm{S_B w} = \lambda \mathrm{S_W w}$$

Generalized Eigenvalue problem

We want to maximize J(**w**). This is equivalent to the derivation of the eigenvector **w** with the largest eigenvalue. Why?

$$\mathrm{S_W}^{-1}\mathrm{S_B w} = \lambda \mathrm{w}$$

If S_w is non-singular and invertible.

Special Case LDA Solution for 2-Class Problems

- Lets replace what S_B is for two classes and see how we can simplify to get a closed form solution.

  (i.e., we would like to get a solution of the vector **w** for the 2-class case.)

- We know that in two class case, there is only 1 **w** vector.
  Lets use this knowledge cleverly…

S_B is rank 1

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T = mm^T$$

$$S_B = mm^T = \begin{bmatrix} | & | & | \\ m(1)m & m(2)m & m(N)m \\ | & | & | \end{bmatrix}$$

S_B has only 1 linearly independent colum vector => Rank 1 matrix

2-class LDA

$$S_W^{-1} S_B \, w = \lambda w$$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{Ni} (x_j - \mu_i)(x_j - \mu_i)^T$$

- Basically in this generalized eigenvalue/eigenvector problem, the number of valid eigenvectors with non- zero eigenvalue is determined by the ***minimum*** rank of matrices S_B and S_W.

- *In this case*, there is only 1 valid eigenvector with a non-zero eigenvalue! (i.e., there is only one valid w vector solution.)

2-class LDA (cont'd)

- Lets see and simplify the 2 class case:

$$(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w = \lambda S_W w$$

$$(\mu_1 - \mu_2)^T w = scalar = \beta$$

which gives $(\mu_1 - \mu_2)\beta = \lambda\, S_W w$

2-Class LDA Closed Form Solution

$$(\mu_1 - \mu_2)\beta = \lambda S_W w$$

Multi-Class LDA

- What if we have more that 2 classes…what then?
- We need more than one **w** projection vector to provide separapability.
- Let’s look at our math derivations to see what changes.

Multi-Class LDA (cont'd)

- Maximize $$J(\mathrm{w}) = \frac{\mathrm{w}^{\mathrm{T}} \mathrm{S}_{\mathrm{B}} \mathrm{w}}{\mathrm{w}^{\mathrm{T}} \mathrm{S}_{\mathrm{W}} \mathrm{w}}$$

- Lets start with the Between-Class Scatter matrix for 2 class.

$$\mathrm{S}_{\mathrm{B}} = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^{\mathrm{T}}$$

- However, S_B now is the between class scatter matrix for *many* classes. We need to make all the class means furthest from each other. One way is to push them as far away from their global mean

$$S_B = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T$$

Multi-Class LDA (cont'd)

- LDA solution: $$S_B w = \lambda S_w w$$

Generalized Eigenvalue problem, the number of valid eigenvectors are bound by the MINIMUM rank of matrix (S_B,S_W). In this case S_B is typically lowest rank which is sum of C outer-product matrices. (Since they subtract the global mean, the rank is C-1.

$$S_w^{-1} S_B w = \lambda w$$

If S_w is non-singular and invertible.

For C classes we have at most C-1 **w** vectors where we can project on to. Why?

When Would LDA Fail?

- What happens when we deal with high-dimensional data.
- If more dimensions *d* than sample # N, then we run into more problems.
- S_w is singular. It will still have at most N-C non-zero eigenvalues.

N is the total number of samples from all classes, C is the number of classes.

$$S_w^{-1} S_B w = \lambda w$$

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{N_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

$$S_B = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T$$

Fisherfaces

- Solution? Fisherfaces…..
- First do PCA and keep N-C eigenvectors. Project your data on to these N-C eigenvectors. (S_w will now be full rank = N-C not d.)
- Do LDA and compute the c-1 projections in this N-C dimensional subspace.
- PCA + LDA = Fisherfaces!

(read the famous PAMI paper of ‘Fisherfaces vs Eigenfaces’)

$$S_w^{-1} S_B w = \lambda w$$

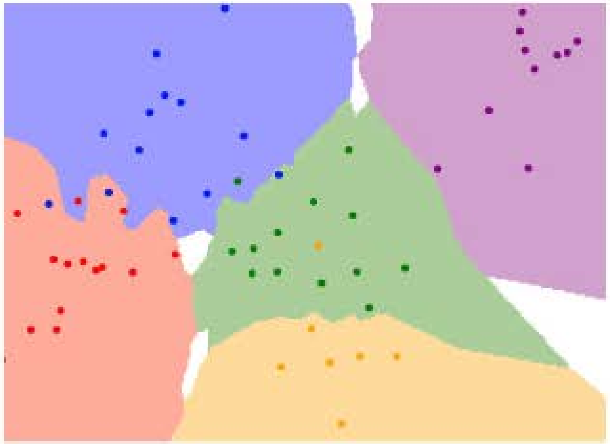
$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{N_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

$$S_B = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T$$

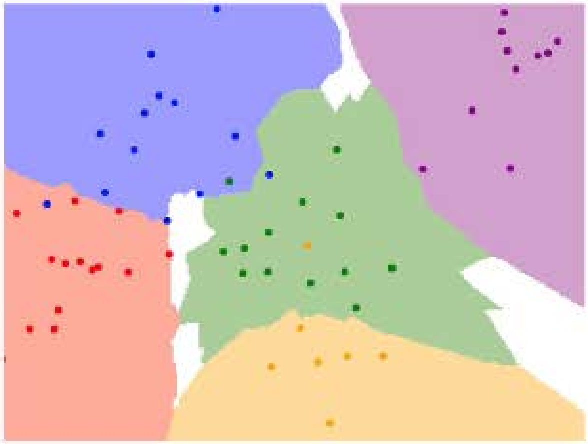
Hyperparameters in ML

- Recall that for k-NN, we need to determine the k value in advance.
  - What is the best k value?
  - And, what is the best distance/similarity metric?
  - Similarly, take PCA for example, what is the best reduced dimension number?
- **Hyperparameters**: choices about the learning model/algorithm of interest
  - We need to determine such hyperparameters instead of learn them.
  - Let's see what we can do and cannot do…

k = 1



k = 3

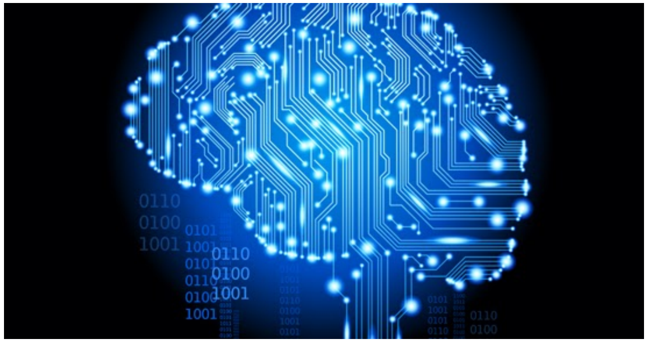


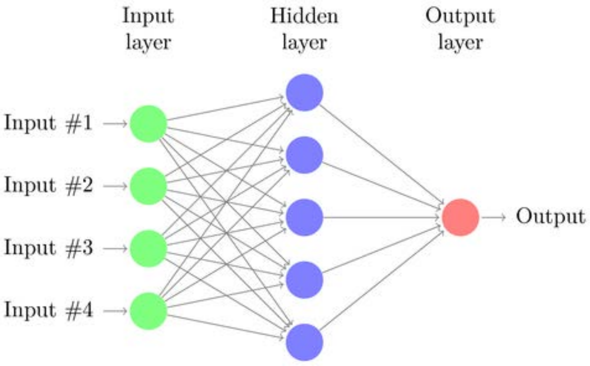
k = 5

Image credit: Stanford CS231n

What's to Be Covered Today…

- Unsupervised vs. Supervised Learning
  - Clustering
  - Unsup. vs. Sup. Dimension Reduction
  - Training, testing, & validation
- Image Representation
  - Bag-of-Words Representation
  - Linear Classification
  - Intro to Neural Networks





How to Determine Hyperparameters?

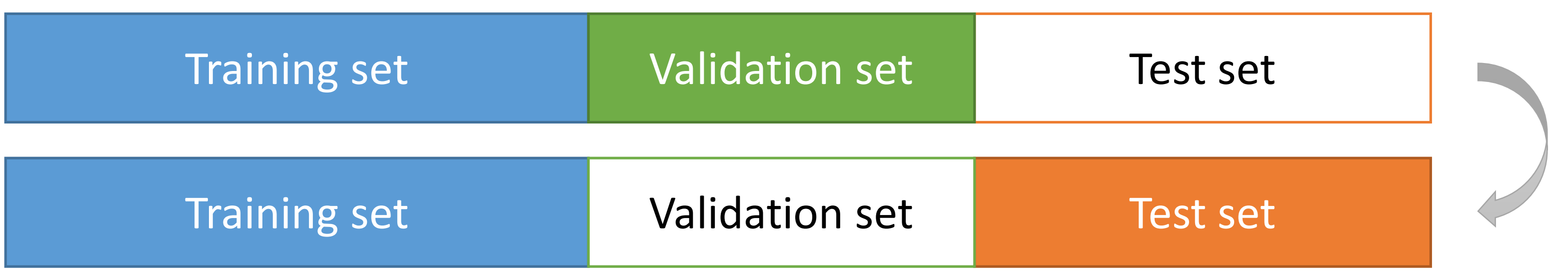
- Idea #1
  - Let’s say you are working on face recognition.
  - You come up with your very own feature extraction/learning algorithm.
  - You take a dataset to train your model, and select your hyperparameters based on the resulting performance.
  - 

How to Determine Hyperparameters? (cont’d)

- Idea #2
  - Let’s say you are working on face recognition.
  - You come up with your very own feature extraction/learning algorithm.
  - For a dataset of interest, you split it into training and test sets.
  - You train your model with possible hyperparameter choices, and select those work best on test set data.
  - 

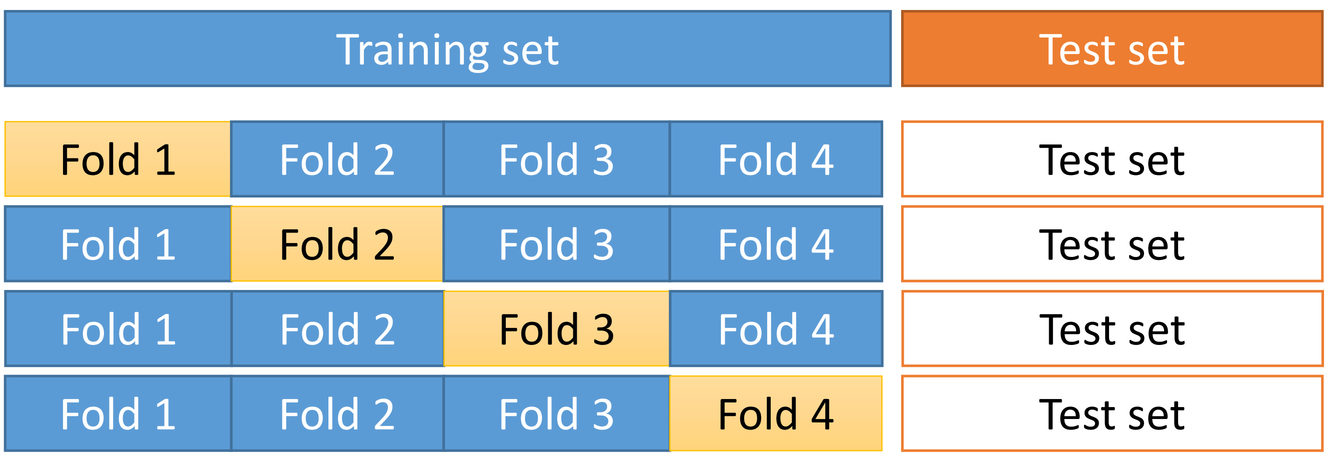
How to Determine Hyperparameters? (cont'd)

- Idea #3
  - Let's say you are working on face recognition.
  - You come up with your very own feature extraction/learning algorithm.
  - For the dataset of interest, it is split it into training, validation, and test sets.
  - You train your model with possible hyperparameter choices, and select those work best on the validation set.
  - 



How to Determine Hyperparameters? (cont'd)

- Idea #3.5
  - What if only training and test sets are given, not the validation set?
  - Cross-validation (or *k-fold* cross validation)
    - Split the training set into k folds with a hyperparameter choice
    - Keep 1 fold as validation set and the remaining k-1 folds for training
    - After each of k folds is evaluated, report the average validation performance.
    - Choose the hyperparameter(s) which result in the highest average validation performance.
  - Take a 4-fold cross-validation as an example…



Minor Remarks on NN-based Methods

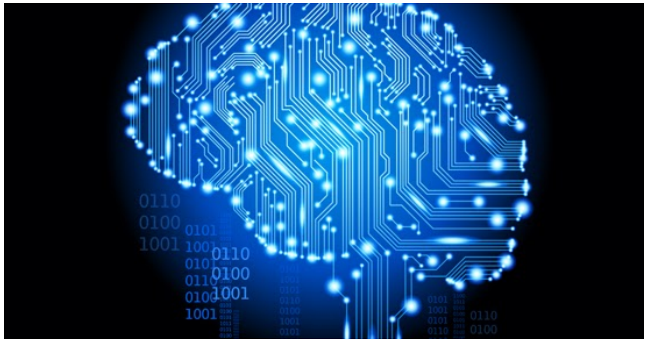
- In fact, k-NN (or even NN) is not of much interest in practice. Why?
  - Choice of distance metrics might be an issue. See example below.
  - Measuring distances in high-dimensional spaces might not be a good idea.
  - Moreover, NN-based methods require lots of and !
    (That is why NN-based methods are viewed as *data-driven* approaches.)

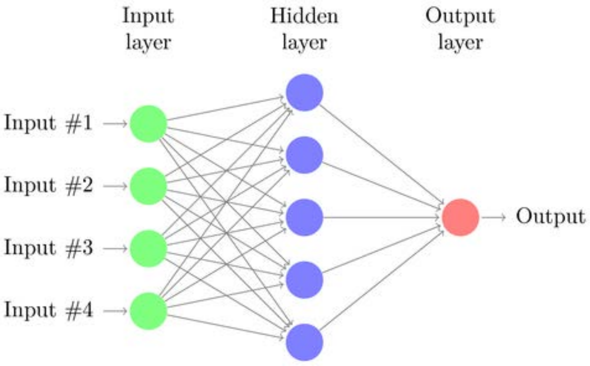
All three images have the same Euclidean distance to the original one.

Image credit: Stanford CS231n

What’s to Be Covered Today…

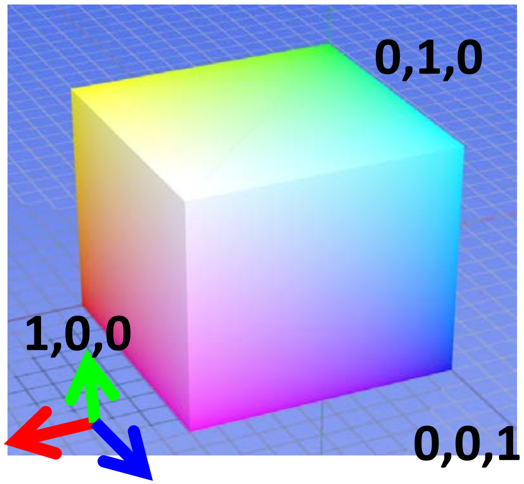
- Unsupervised vs. Supervised Learning
  - Clustering
  - Unsup. vs. Sup. Dimension Reduction
  - Training, testing, & validation
- Image Representation
  - Bag-of-Words Representation
  - Linear Classification
  - Intro to Neural Networks





Color as Image Representation

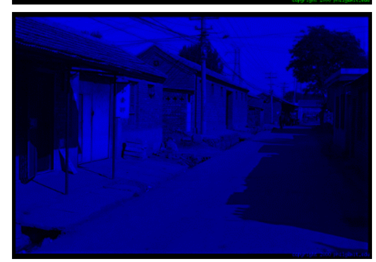
- Default Color Space



- Remarks
  - Easy for devices
  - But not perceptual
  - Where do the grays live?
  - Where is hue and saturation?

R
(G=0,B=0)

G
(R=0,B=0)



B
(R=0,G=0)

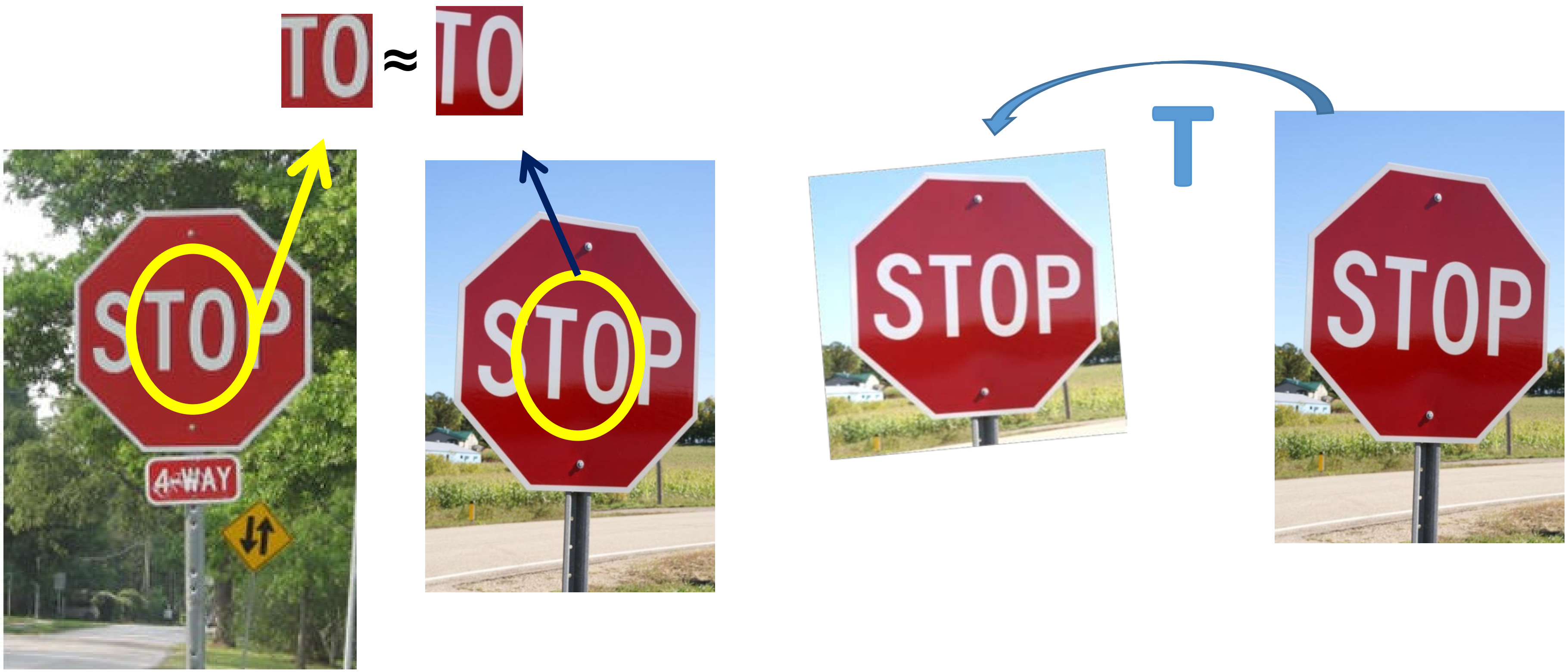
Image from: http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png

Interest Points as Image Representation

- Examples
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Object recognition
  - Robot navigation
  - Indexing and database retrieval

Recall that: Interest Points?

- Registration & Correspondence
  - Identifying corresponding points/patches/regions across images
  - Apps: matching, alignment, stitching, etc.



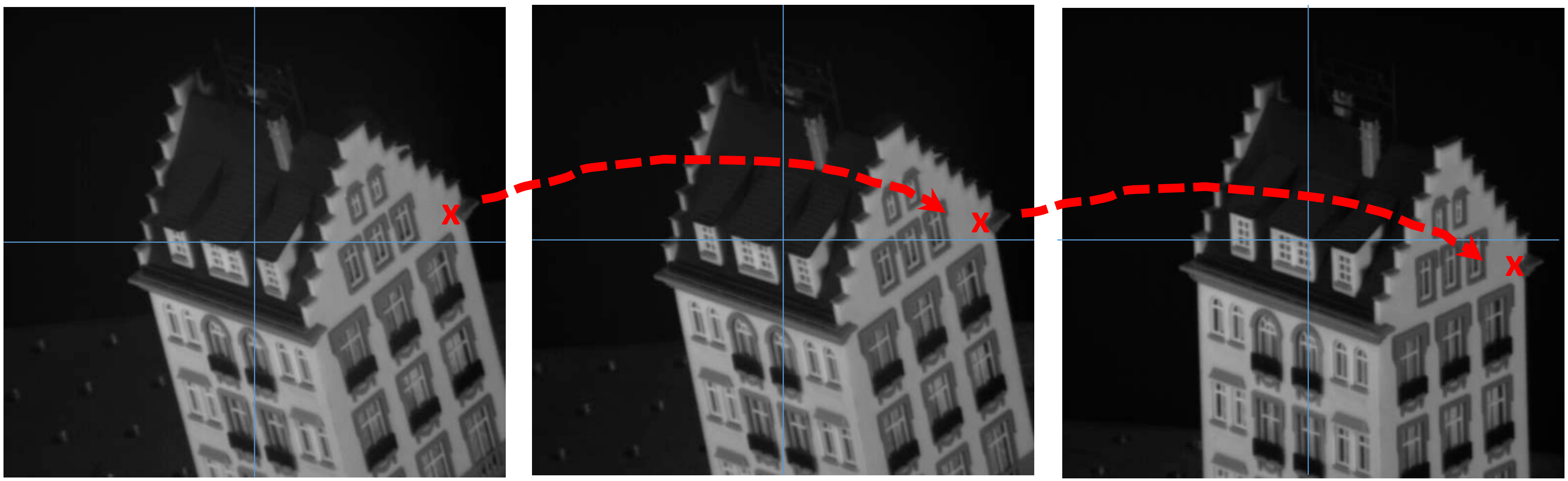
Why Interest Points? (cont'd)

- Example: panorama

Credit: Matt Brown

Why Interest Points? (cont’d)

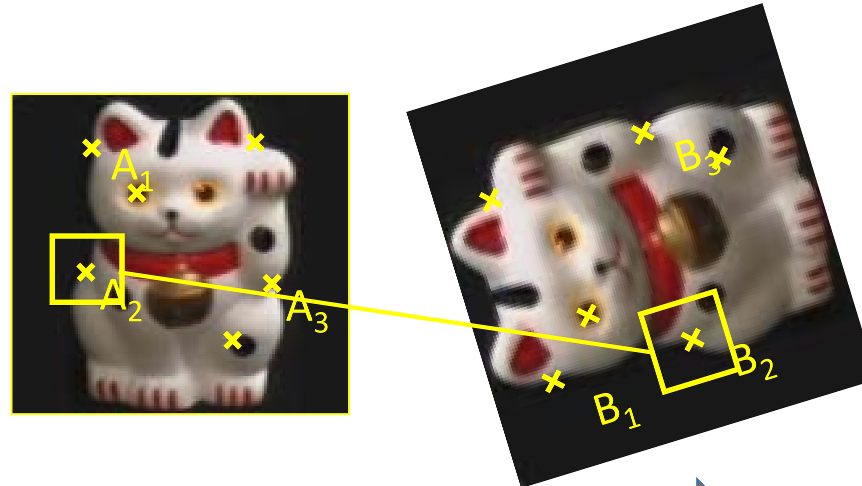
- Example: tracking



frame 0 frame 22 frame 49

About Interest Points

- Key Trade-offs



Detection

Few Points

More distinctive representation
Robust detection
Precise localization

More Points

Robust to occlusion
Works with less texture

Description

More Distinctive

Minimize wrong matches

More Flexible

Robust to expected variations
Maximize correct matches

Scale Invariant Feature Transform (SIFT)

- Key Ideas
  - Take a 4 x 4 (= 16 grids) square window around each detected keypoint
  - Compute edge orientation (angle of the gradient - 90°) for each pixel in it
  - Throw out weak edges (threshold gradient magnitude)
  - Create histogram of surviving edge orientations

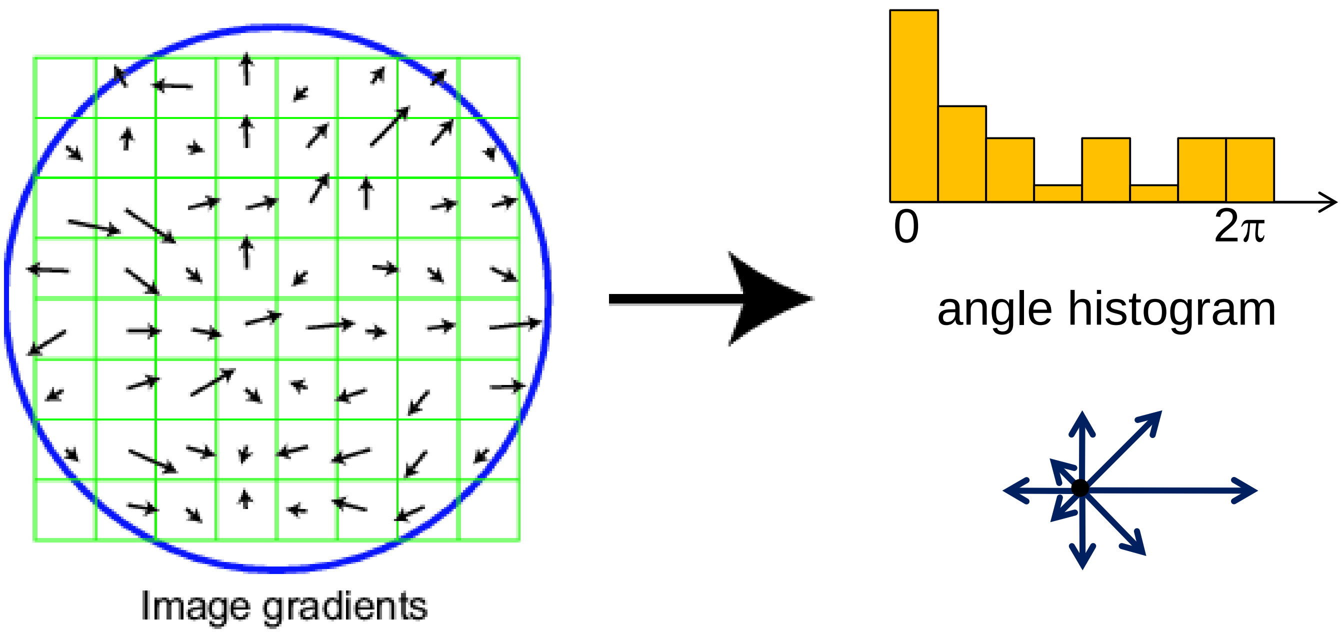
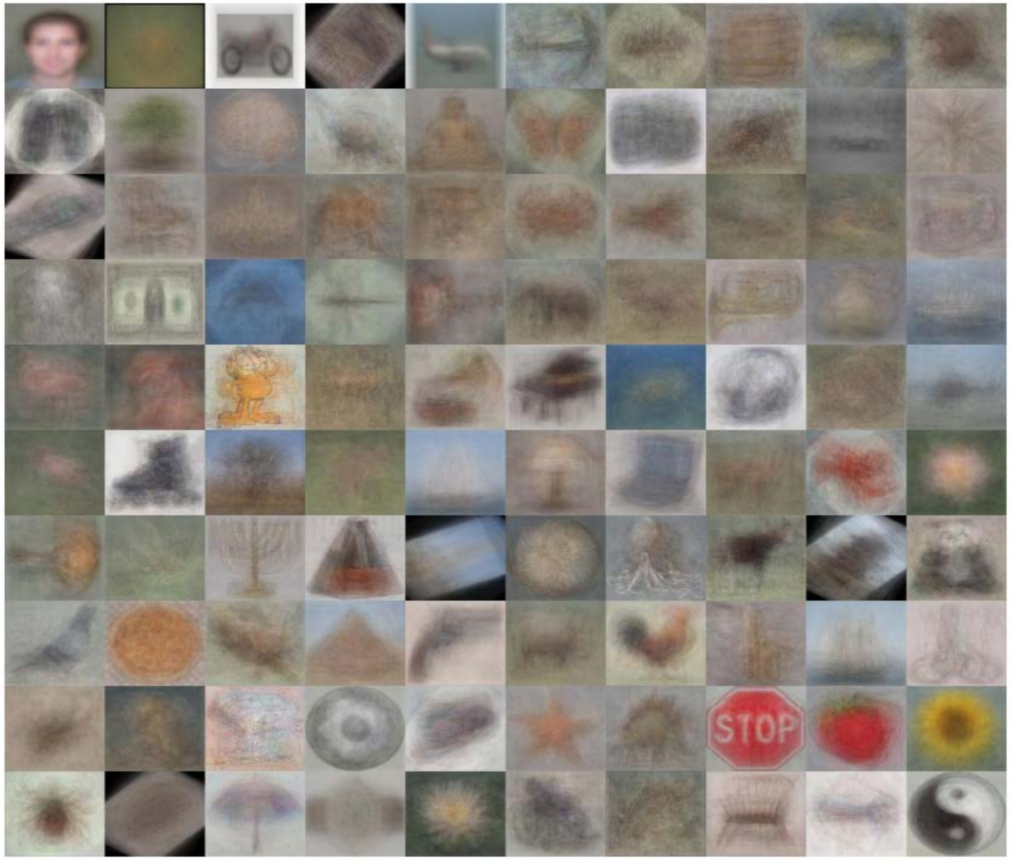


Image Categorization

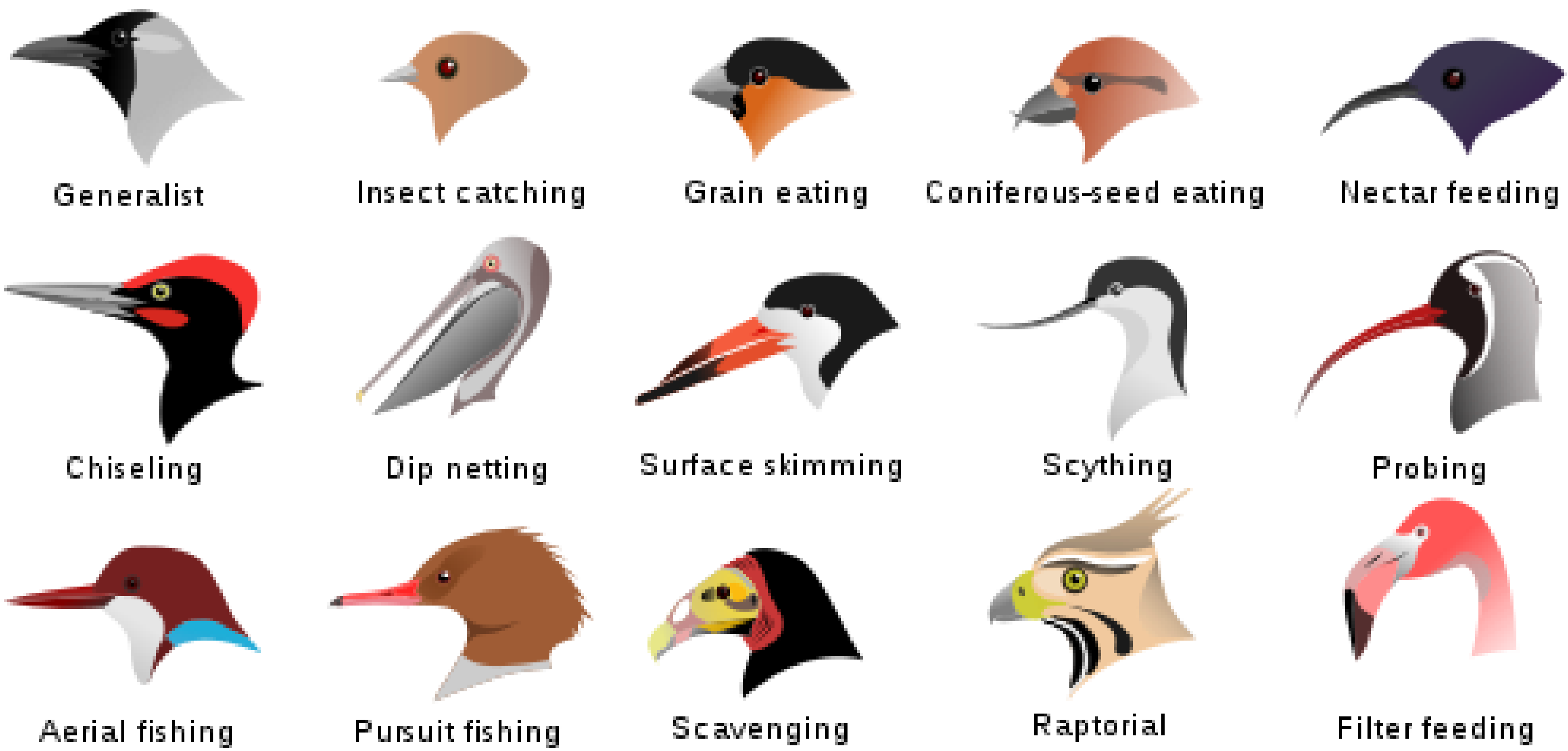
- Object Recognition



Average Object Images of Caltech 101

Image Categorization

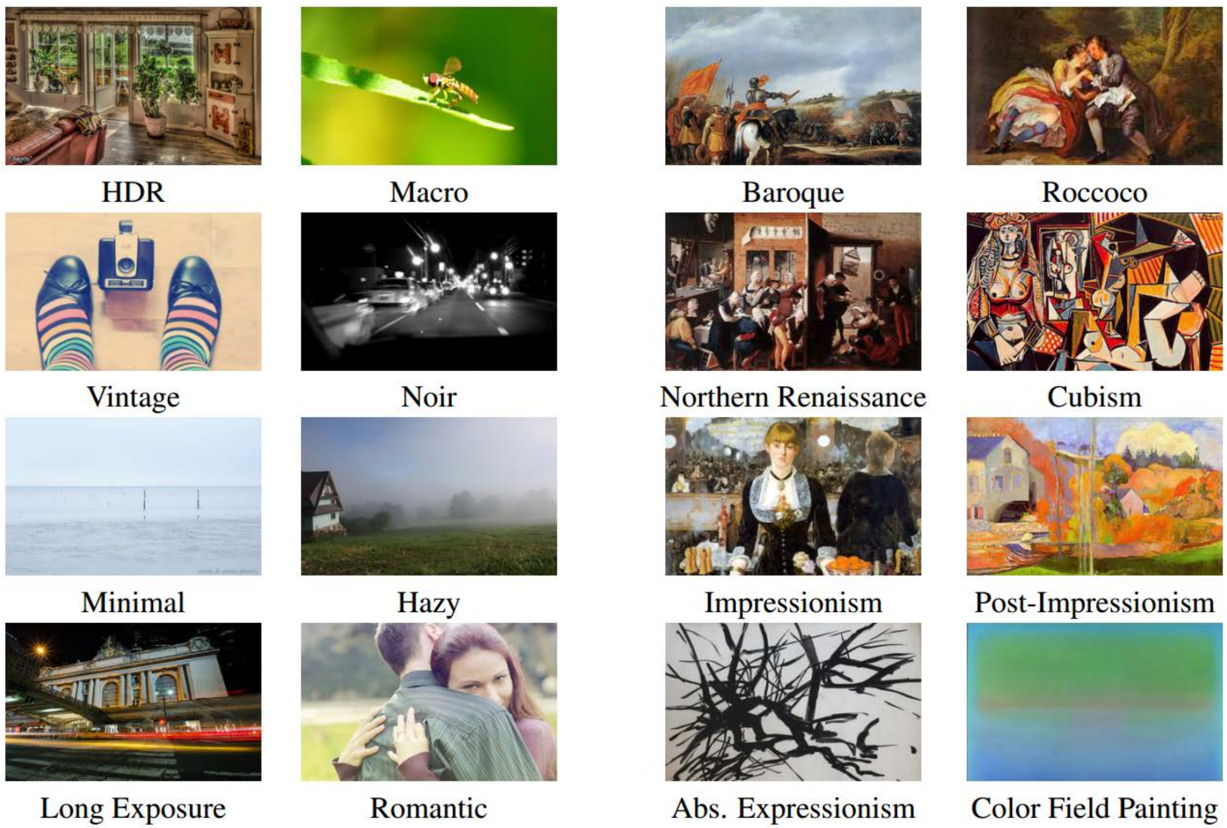
- Fine-Grained Recognition



[Visipedia Project]

Image Categorization

- Image style recognition



Flickr Style: 80K images covering 20 styles.

Wikipaintings: 85K images for 25 art genres.

[Karayev et al. BMVC 2014]

Image Categorization

- Dating historical photos

1940 1953 1966 1977

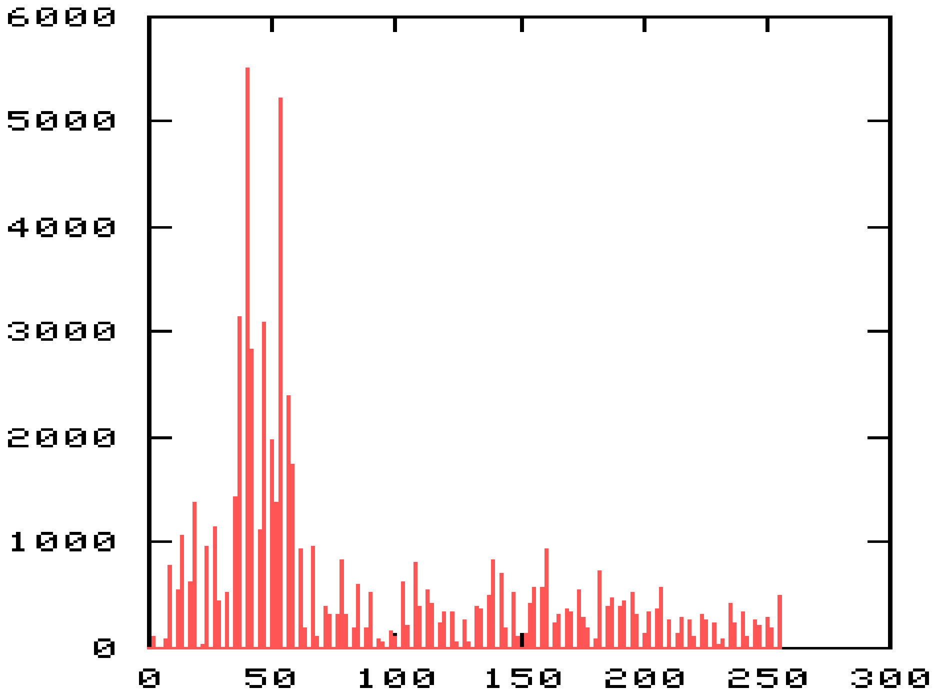
[Palermo et al. ECCV 2012]

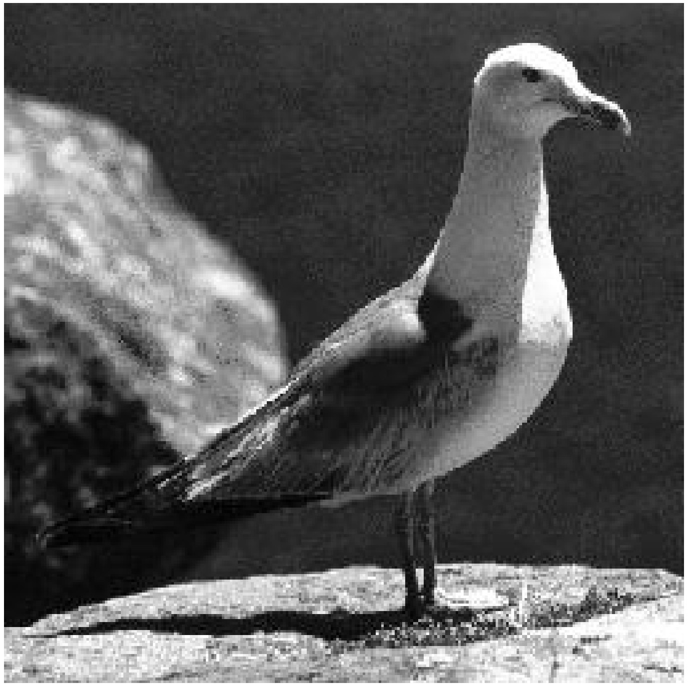
What Are the Right Features?
(When deep features are not applicable…)

- Depending on the task of interest!
- Possible choices
  - Object: shape
    - Local shape info, shading, shadows, texture
  - Scene : geometric layout
    - linear perspective, gradients, line segments
  - Material properties: albedo, feel, hardness
    - Color, texture
  - Action: motion
    - Optical flow, tracked points

Image Representation: Histograms

- Global histogram
  - Possible to describe color, texture, depth, or even interest points!

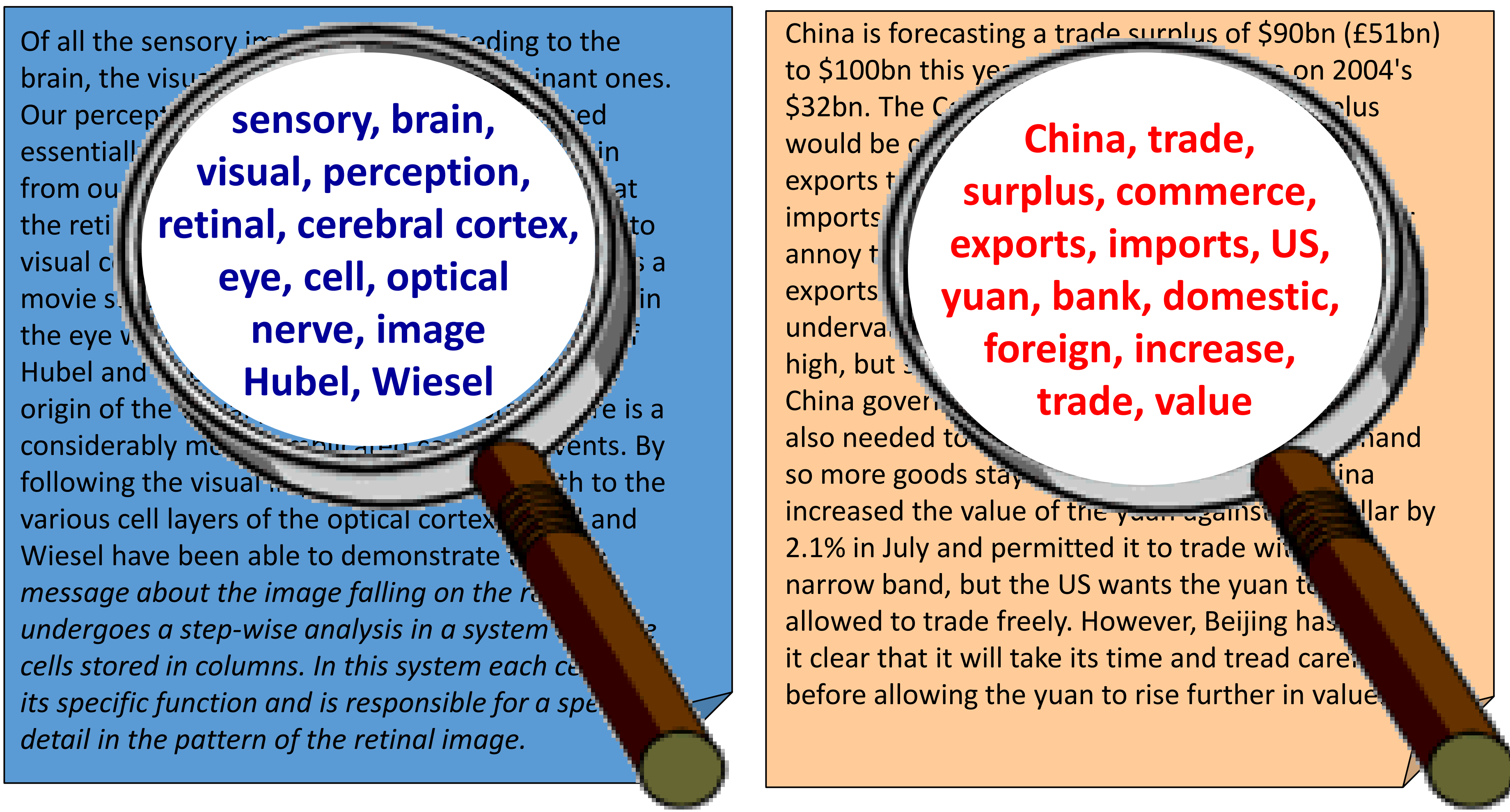




Images from Dave Kauchak

Bag-of-Words Models for Image Classification

- Analogy to document categorization



Bag of Words (or Visual Words)

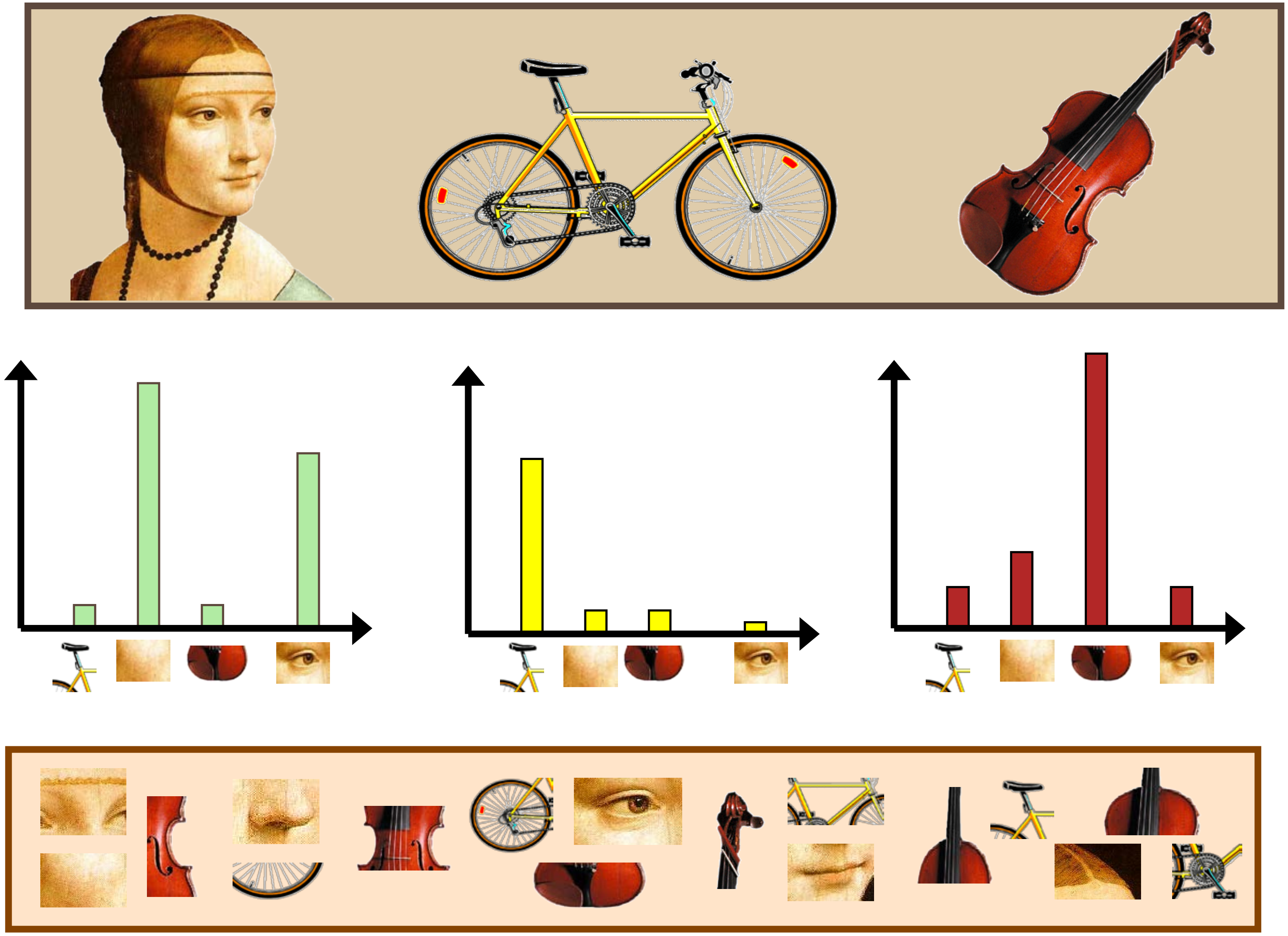
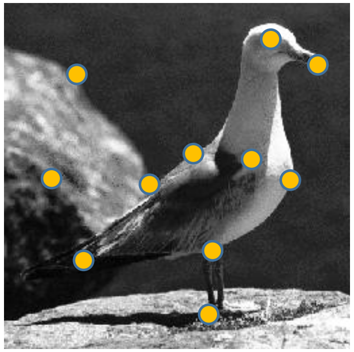


Image Representation: Histograms



- Take images with 2D features/descriptors as an example

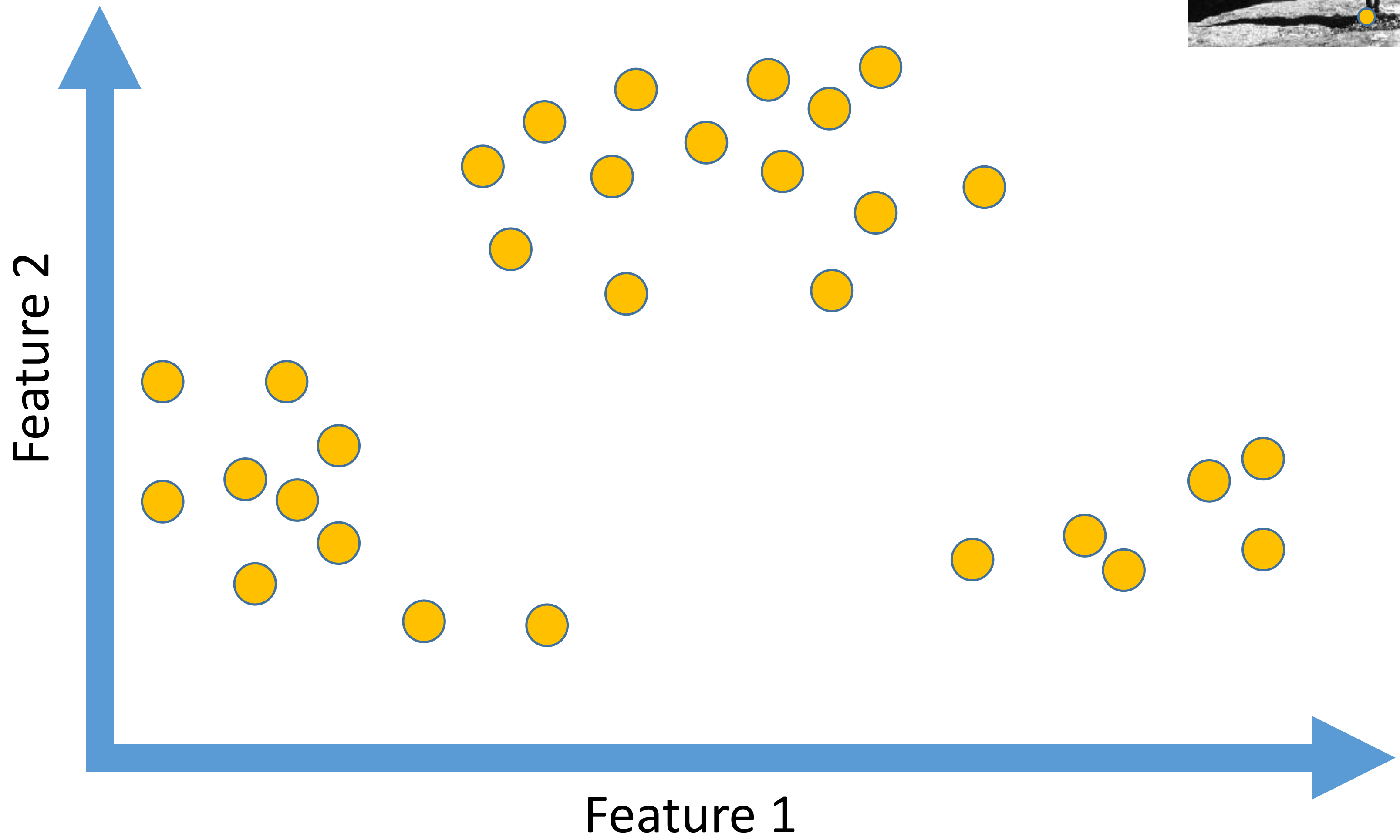


Image Representation: Histograms

- # of occurrence of data in each bin
- Marginal histogram of feature 1

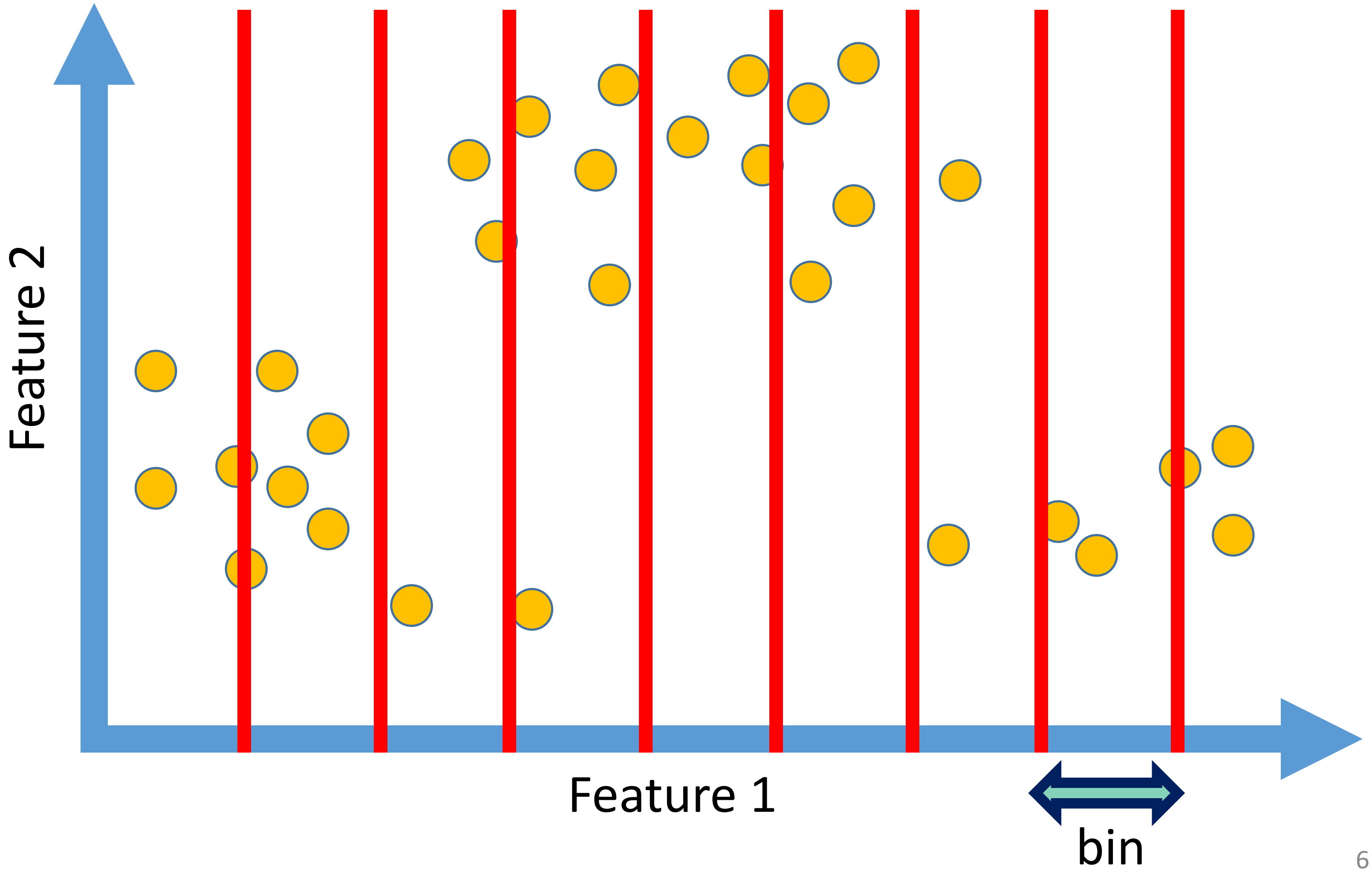


Image Representation: Histograms

- # of occurrence of data in each bin
- Marginal histogram of feature 2

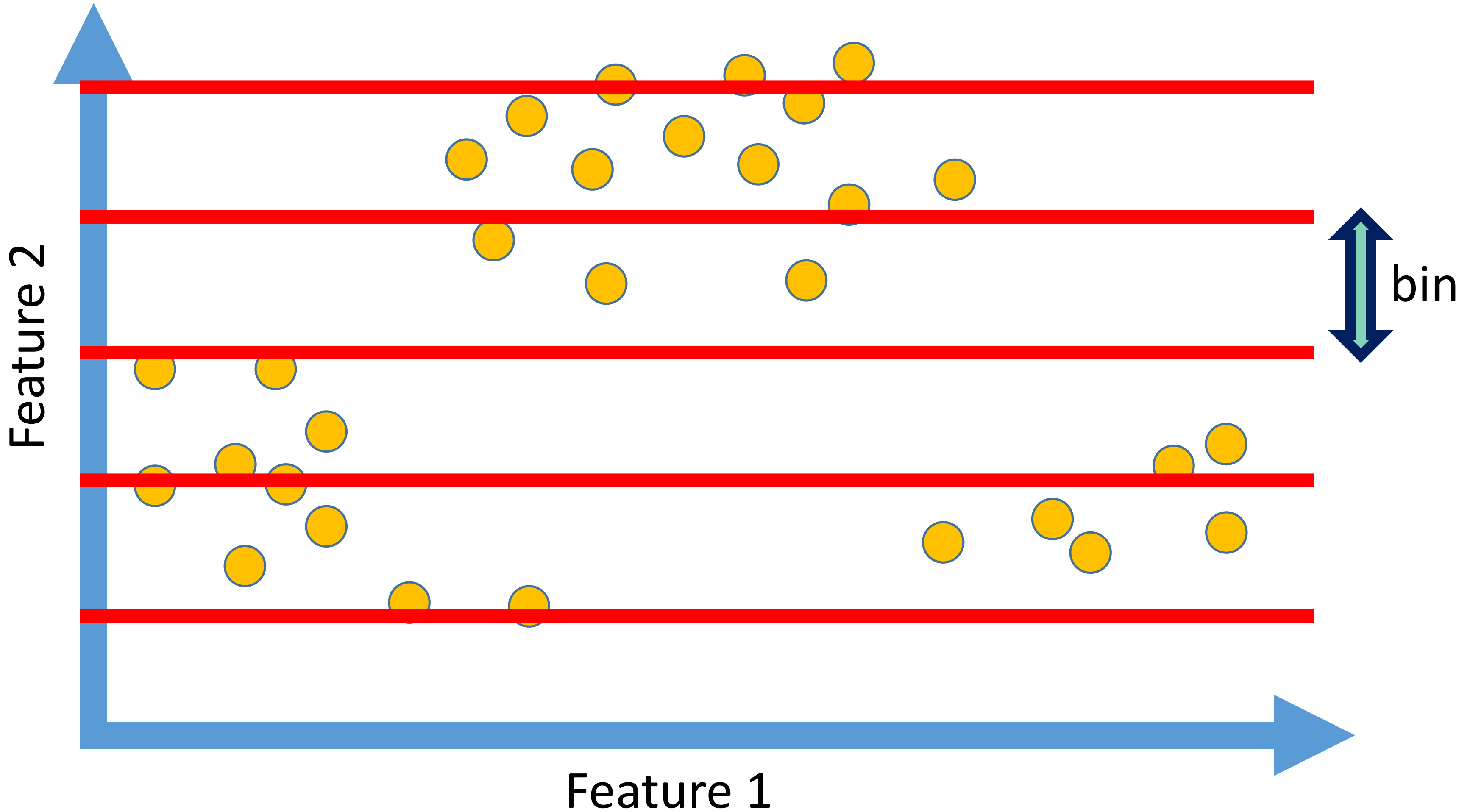


Image Representation: Histograms

- Better modeling (quantization) of multi-dimensional data
- Clustering
  - Use the same cluster center to represent the associated features

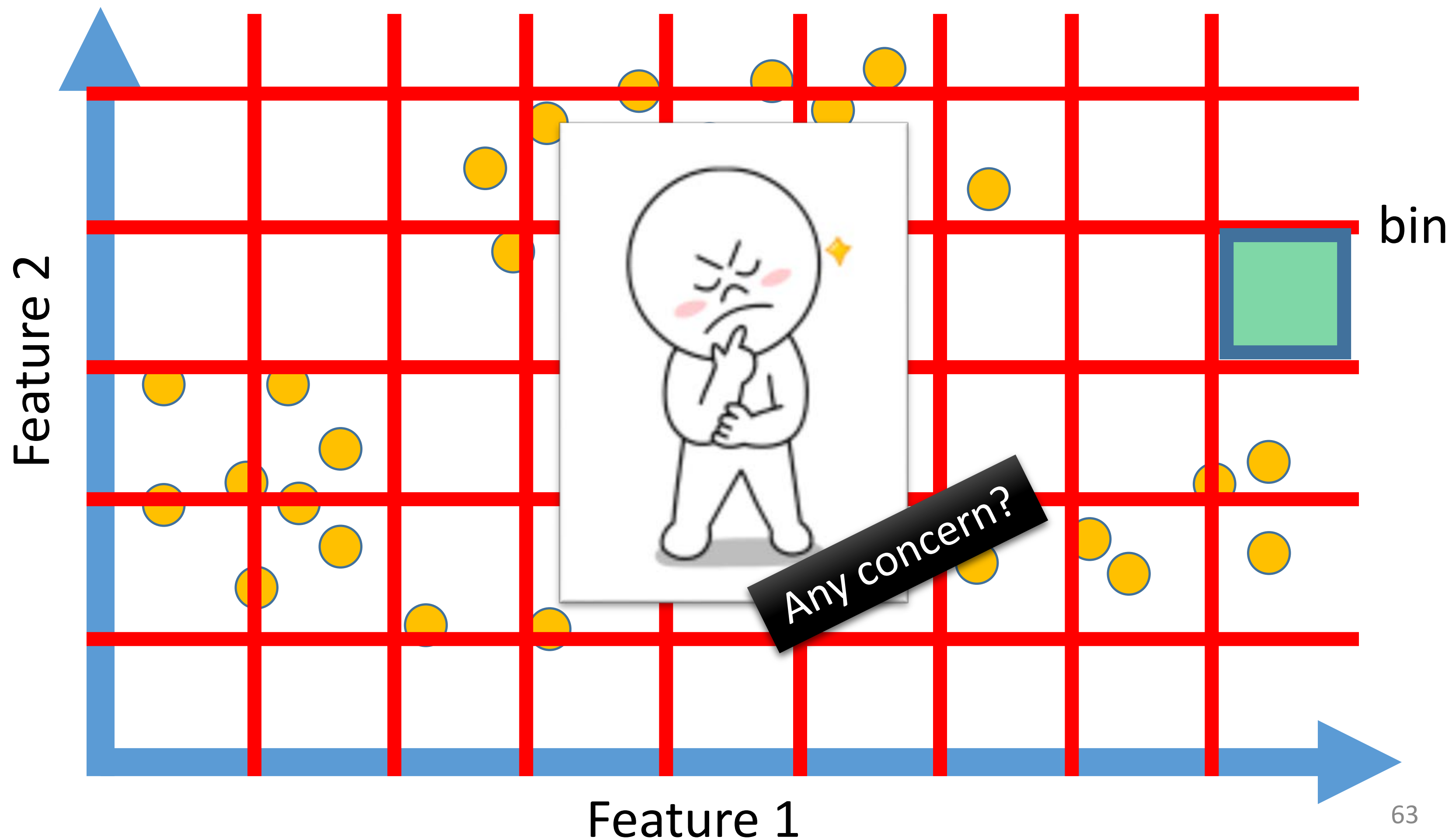
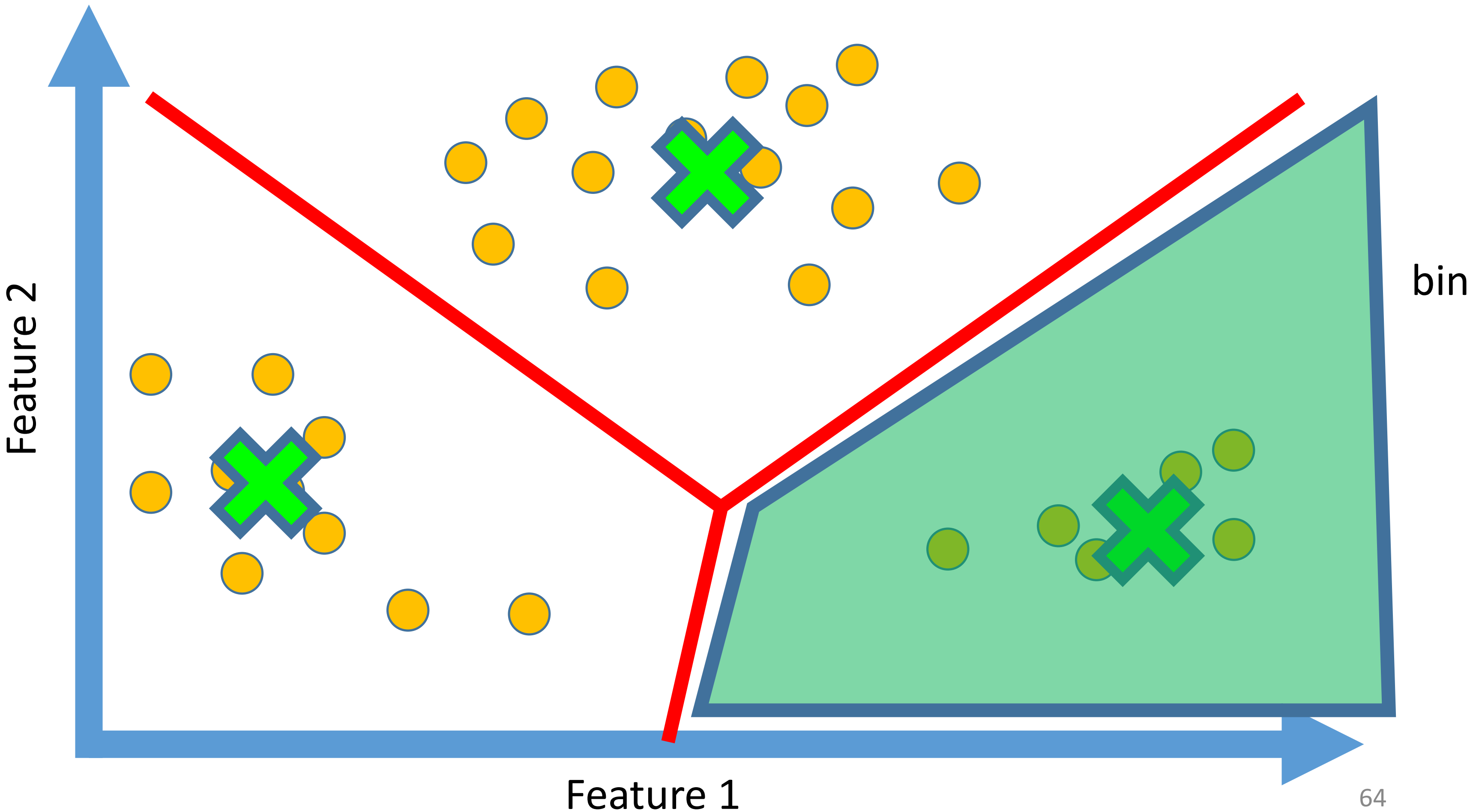


Image Representation: Histograms

- Better modeling (quantization) of multi-dimensional data
- Clustering
  - Use the same cluster center to represent the associated features



Remarks on Histogram-Based Image Representation

- Quantization
  - Grids vs. clusters

Fewer Bins
Need less data
Coarser representation

More Bins
Need more data
Finer representation

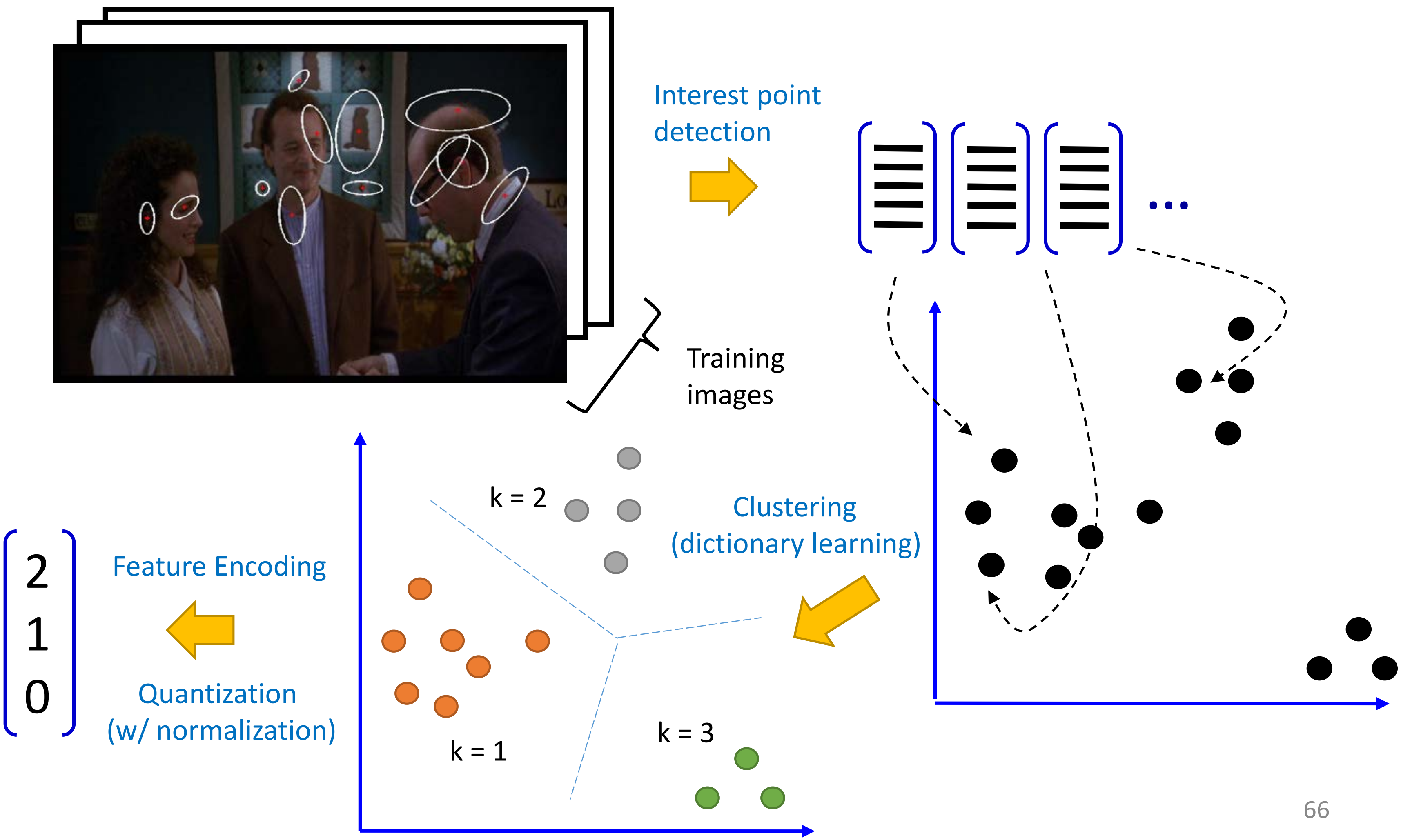
- Possible distance metrics
  - Euclidean distance
  - Histogram intersection kernel
  - Chi-squared distance
  - Earth mover's distance
    (min cost to transform one distribution to another)

$$\mathrm{histint}(h_i, h_j) = 1 - \sum_{m=1}^{K} \min\left(h_i(m), h_j(m)\right)$$

$$\chi^2(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{K} \frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

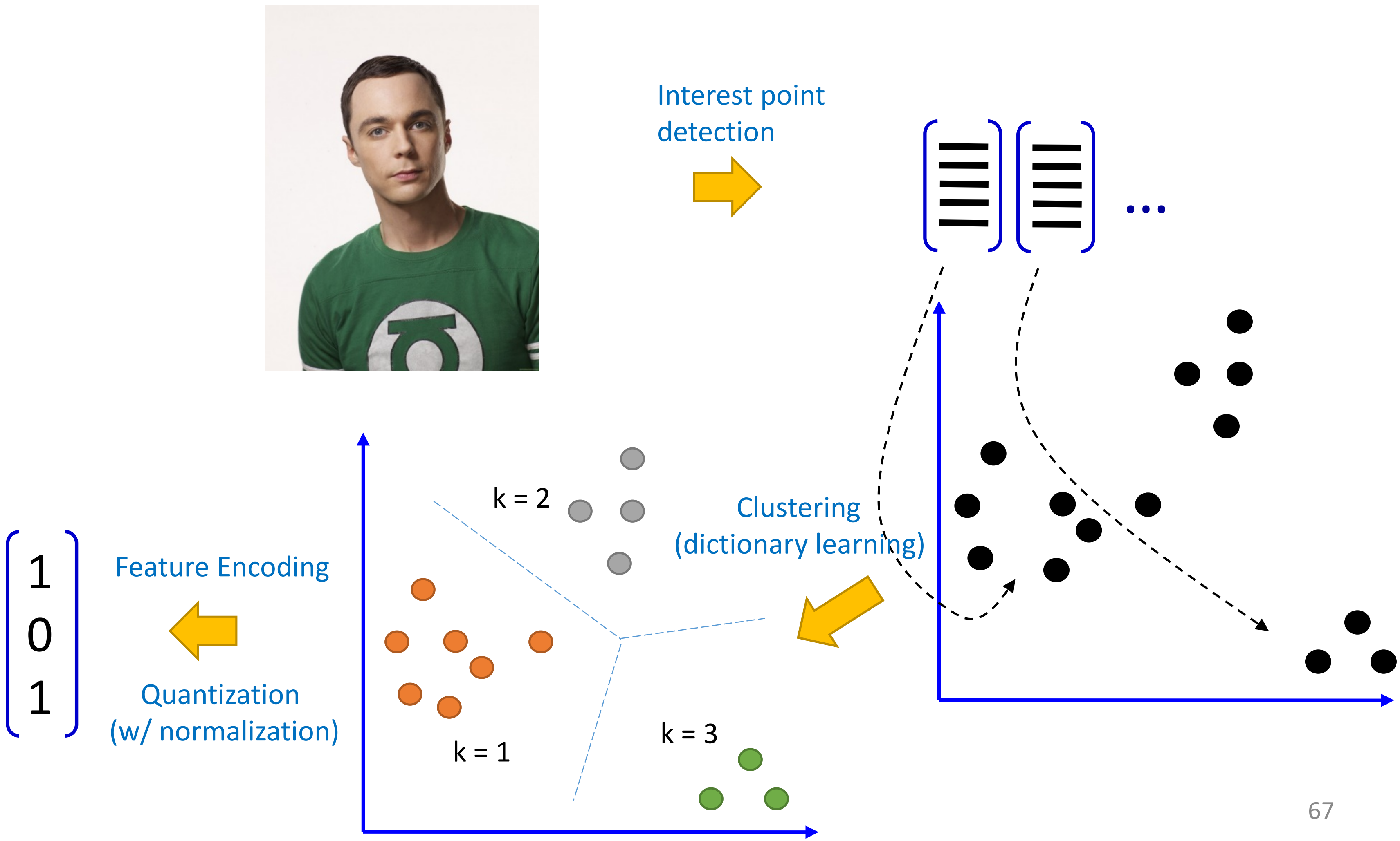
Bag-of-Words for Image Classification

- Training



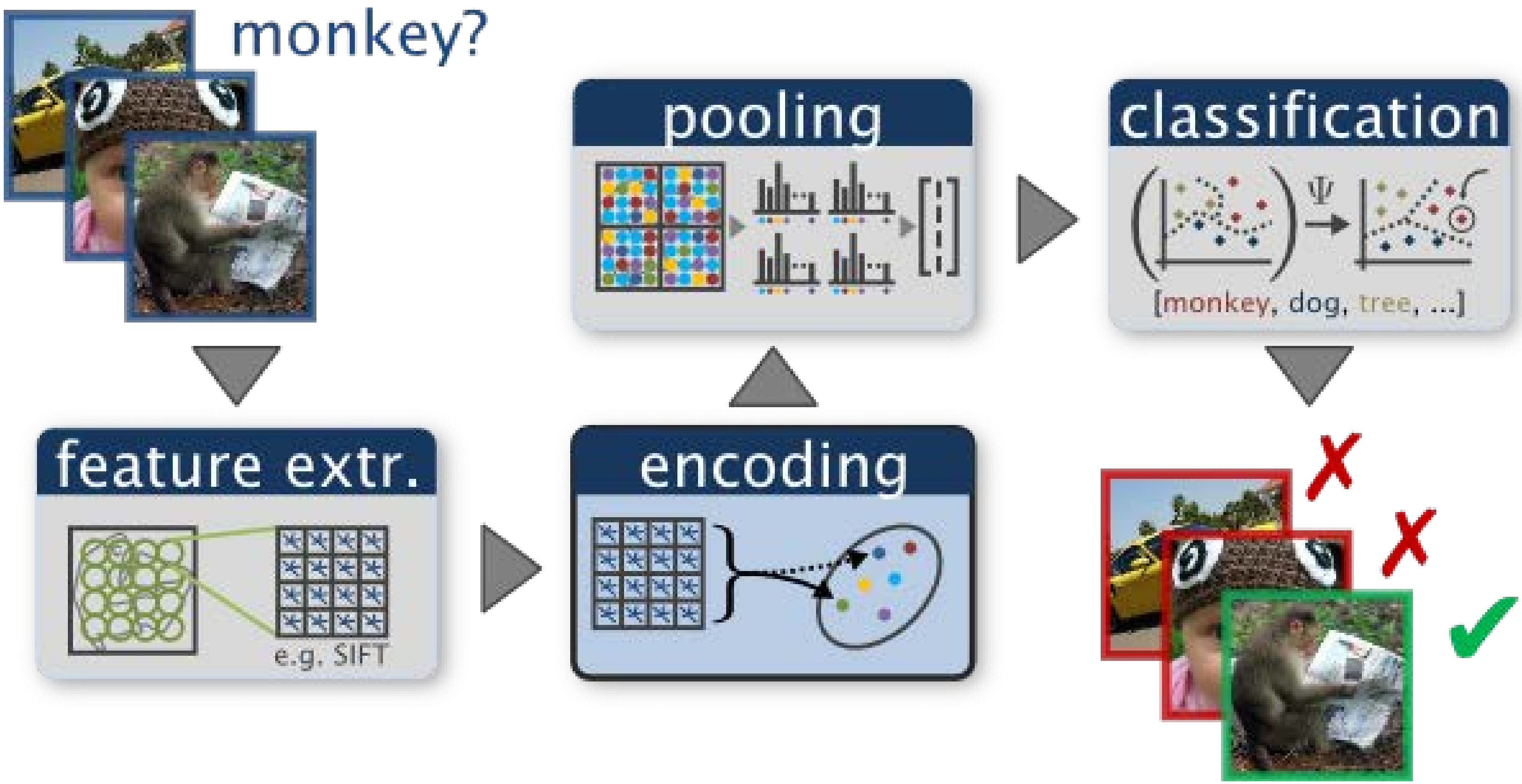
Bag-of-Words for Image Classification

- Testing



Bag-of-Words for Image Classification

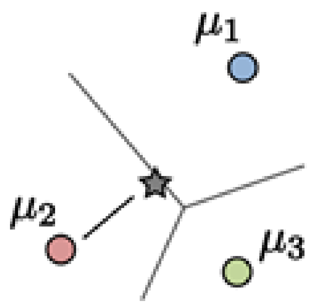
- Overview

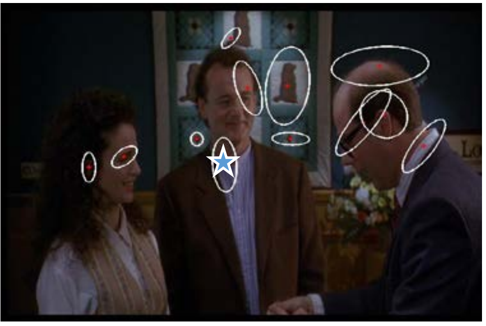


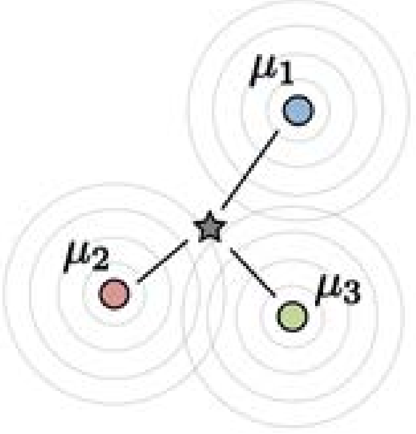
[Chatfieldet al. BMVC 2011]

About Feature Encoding for Bag-of-Words

- Hard vs. soft assignments to clusters

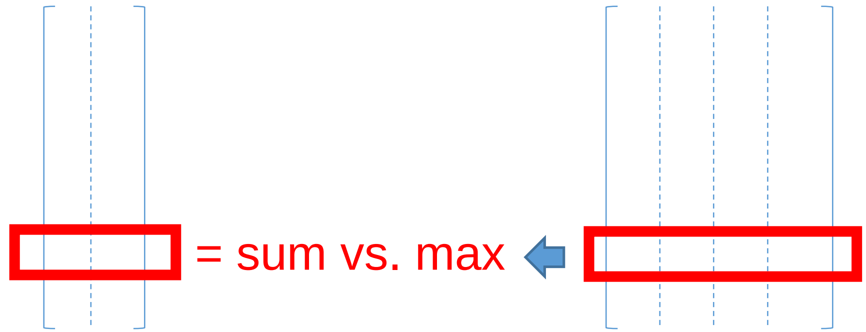


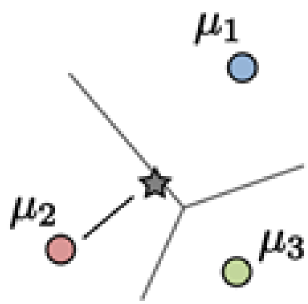


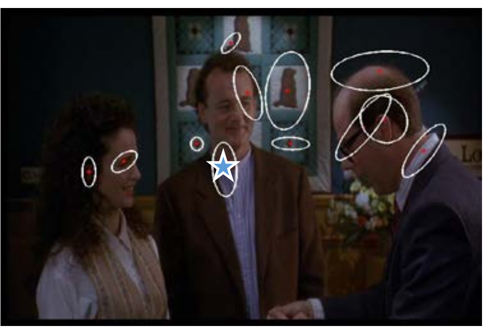


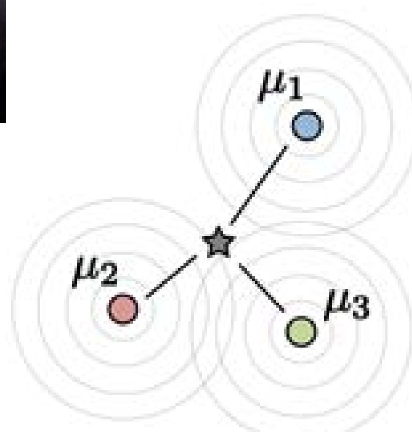
About Feature Encoding for Bag-of-Words

- Sum vs. max pooling



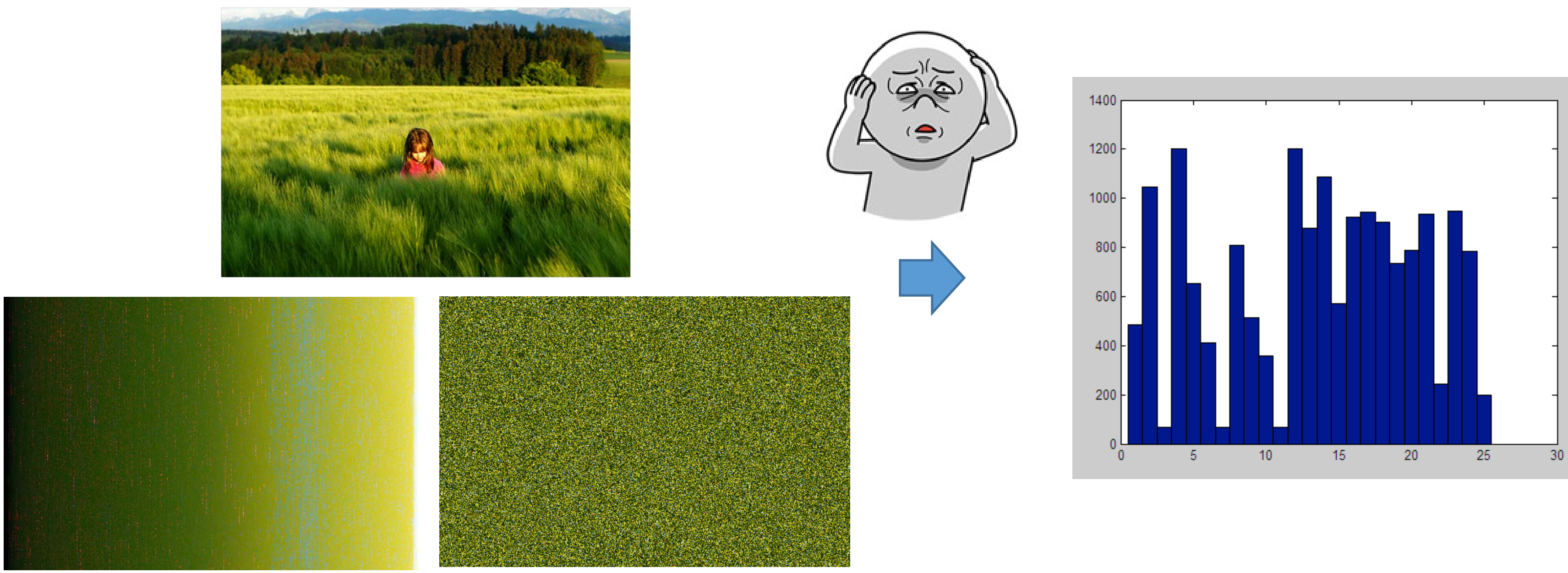






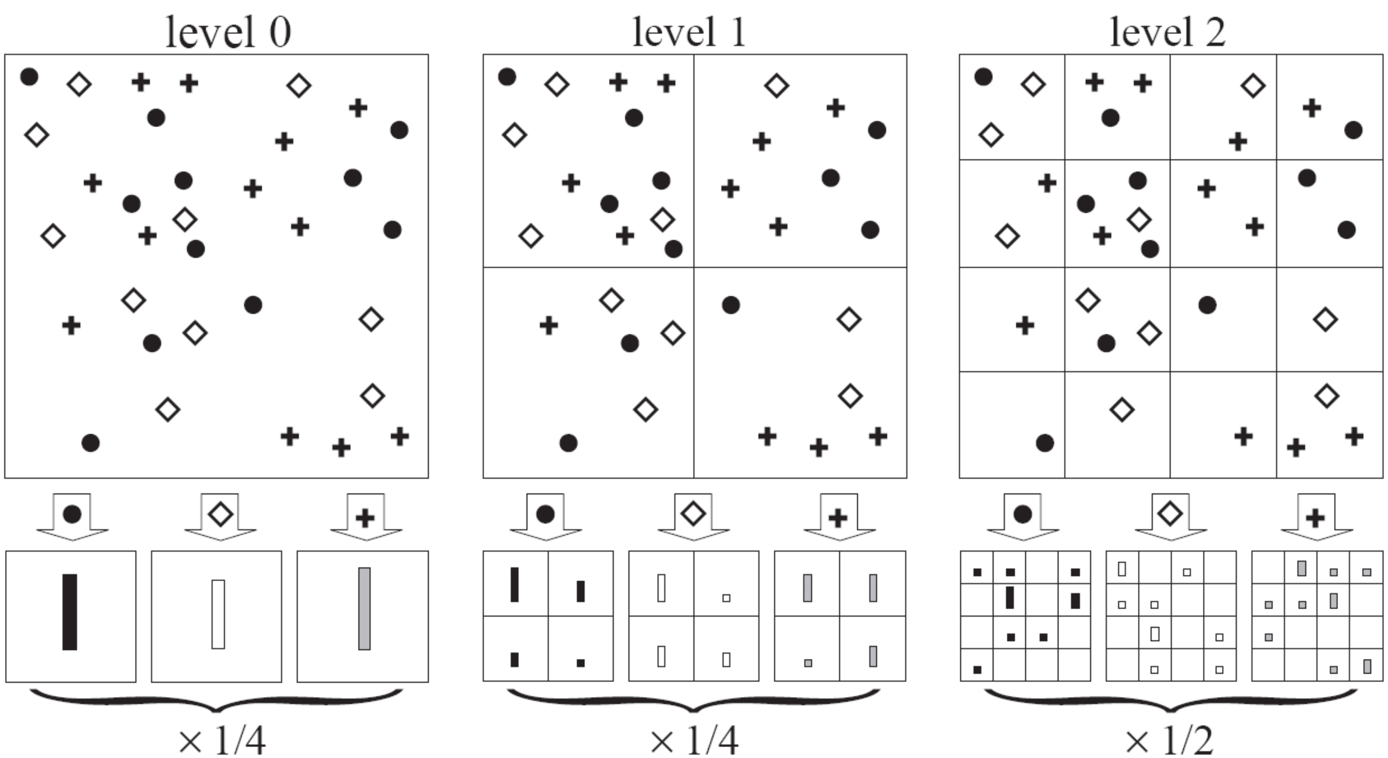
Final Remarks on BoW

- What's the limitation?
  - Loss of…
  - What's the possible solution?



Final Remarks on BoW

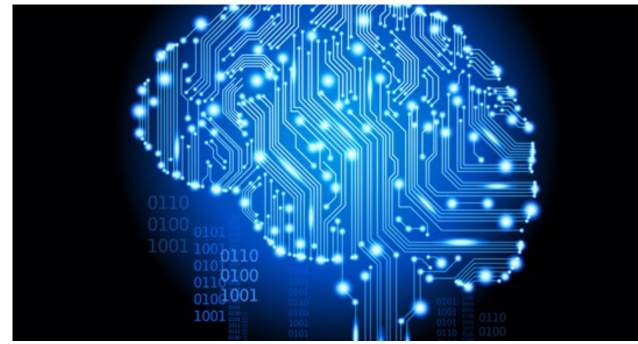
- Spatial pyramid
  - Compute BoW in each spatial grid + concatenation

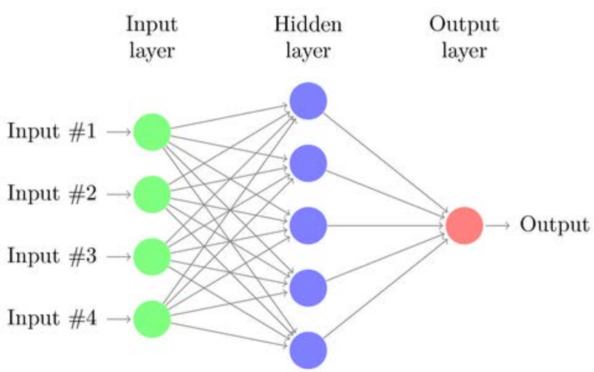


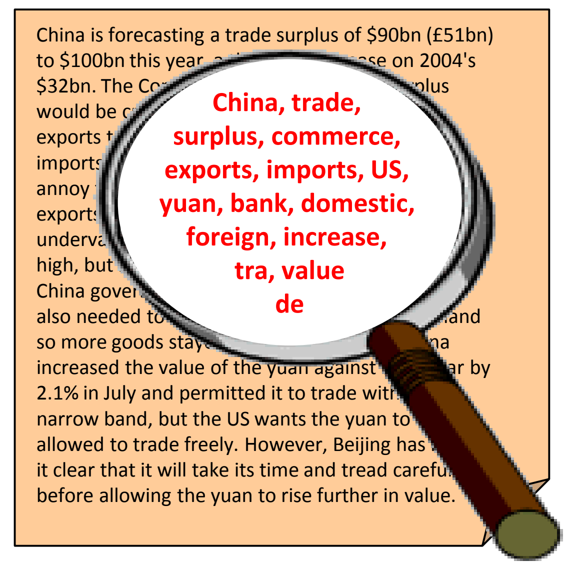
[Lazebnik et al. CVPR 2006]

What's to Be Covered Today…

- Unsupervised vs. Supervised Learning
  - Clustering
  - Unsup. vs. Sup. Dimension Reduction
  - Training, testing, & validation
- Image Representation
  - Bag-of-Words Representation
  - Linear Classification
  - Intro to Neural Networks

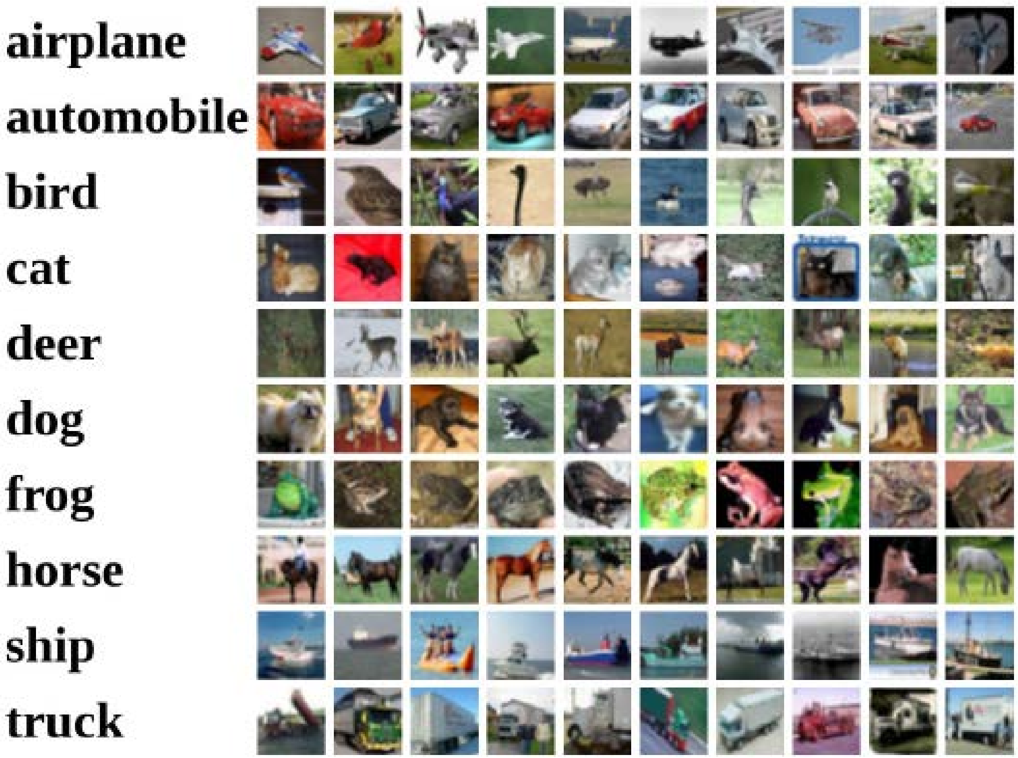






Linear Classification

- Linear Classifier
  - Can be viewed as a parametric approach. Why?
  - Assuming that we need to recognize 10 object categories of interest
  - E.g., CIFAR10 with 50K training & 10K test images of 10 categories.
    And, each image is of size 32 x 32 x 3 pixels.



Linear Classification (cont'd)

- Linear Classifier
  - Can be viewed as a parametric approach. Why?
  - Assuming that we need to recognize 10 object categories of interest (e.g., CIFAR10).
  - Let's take the input image as **x**, and the linear classifier as **W**. We hope to see that **y** = **Wx** + **b** as a 10-dimensional output indicating the score for each class.

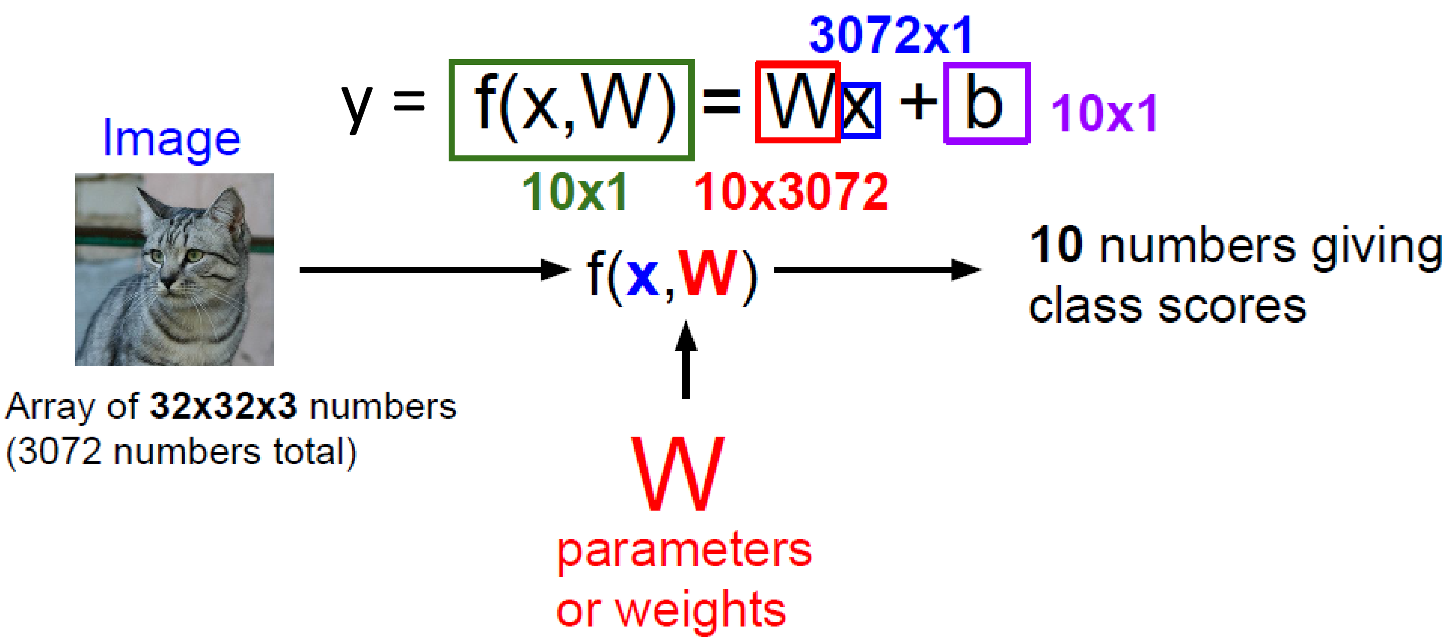


Image credit: Stanford CS231n

Linear Classification (cont'd)

- Linear Classifier
  - Can be viewed as a parametric approach. Why?
  - Assuming that we need to recognize 10 object categories of interest (e.g., CIFAR10).
  - Let's take the input image as **x**, and the linear classifier as **W**. We hope to see that **y** = **Wx** + **b** as a 10-dimensional output indicating the score for each class.
  - Take an image with 2 x 2 pixels & 3 classes of interest as example: we need to learn linear transformation/classifer **W** and bias **b**, so that desirable outputs **y** = **Wx** + **b** can be expected.

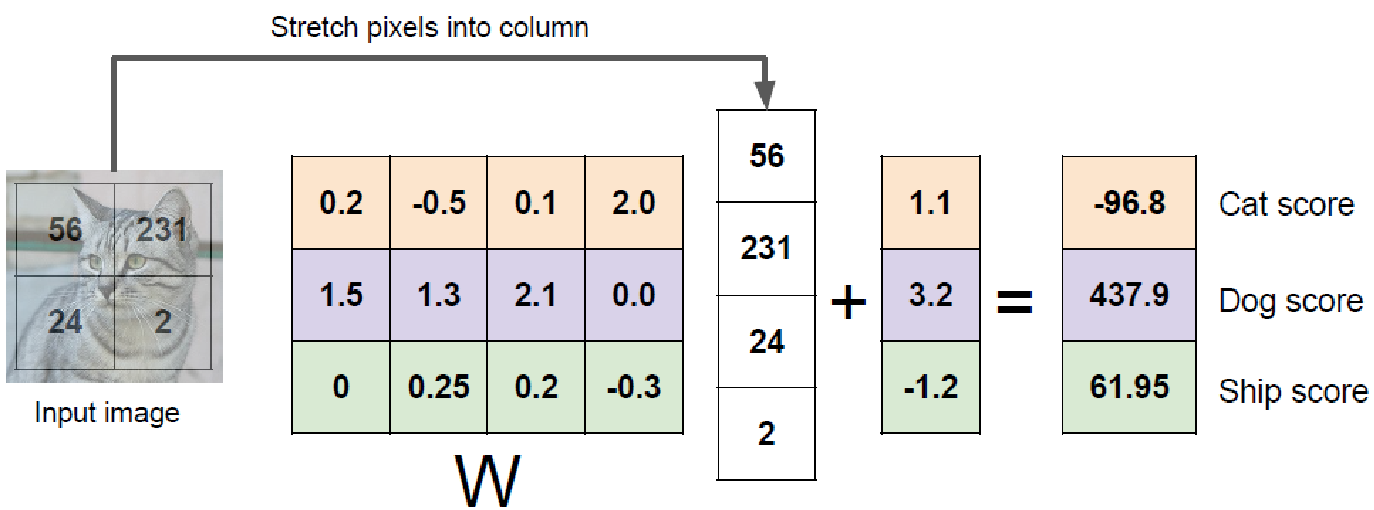
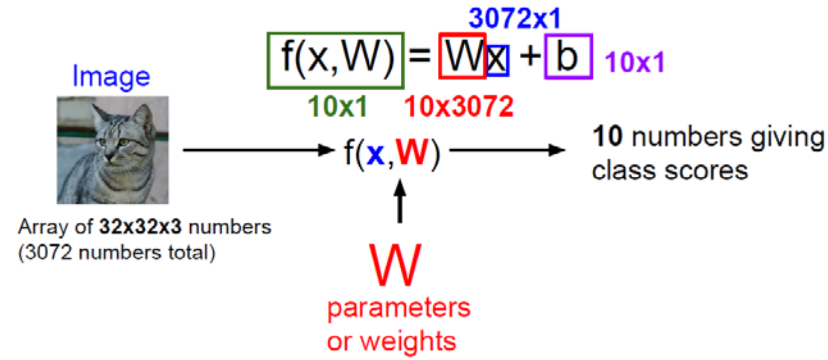


Image credit: Stanford CS231n

Some Remarks



- Interpreting **y** = **Wx** + **b**
  - What can we say about the learned **W**?
  - The weights in **W** are trained by observing training data **X** and their ground truth **Y**.
  - Each column in **W** can be viewed as an exemplar of the corresponding class.
  - Thus, **Wx** basically performs inner product (or correlation) between the input **x** and the exemplar of each class. (Signal & Systems!)

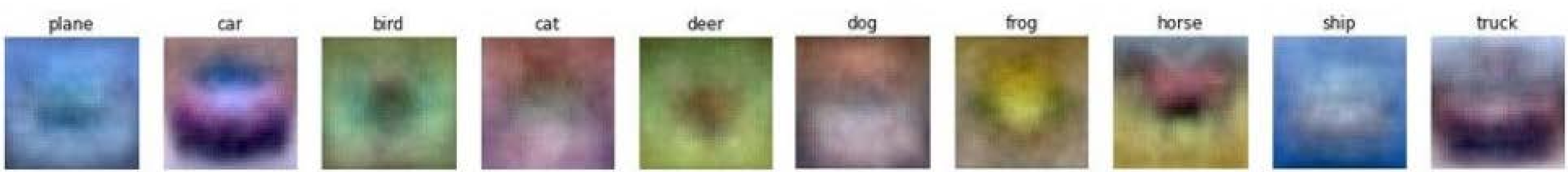


Image credit: Stanford CS231n

Linear Classification

- Remarks
  - Starting points for many multi-class or complex/nonlinear classifier
  - How to determine a proper loss function for matching **y** and **Wx+b**, and thus how to learn the model **W** (including the bias **b**), are the keys to the learning of an effective classification model.

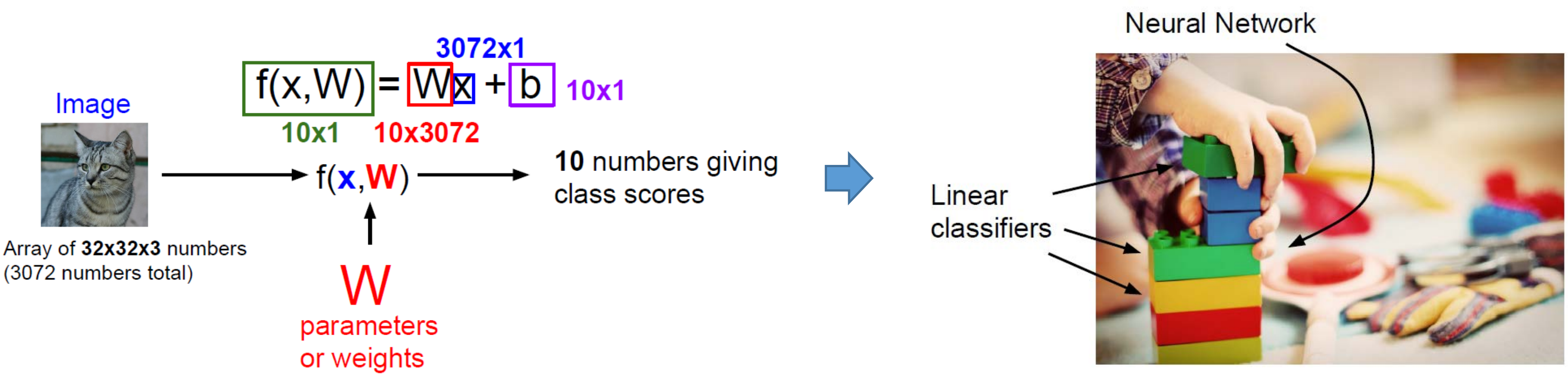
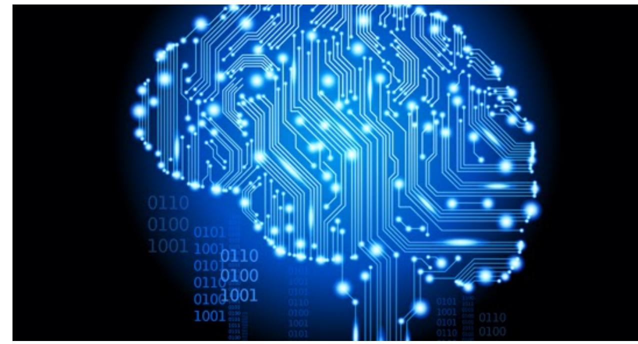
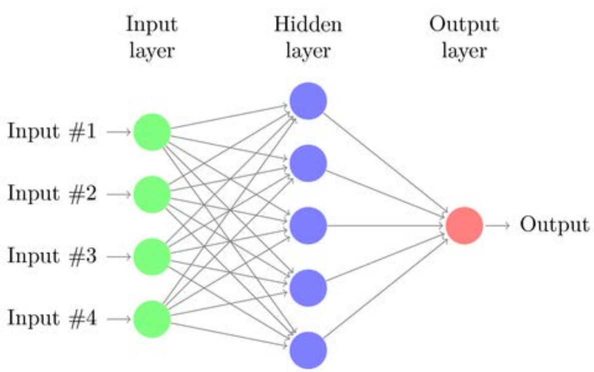


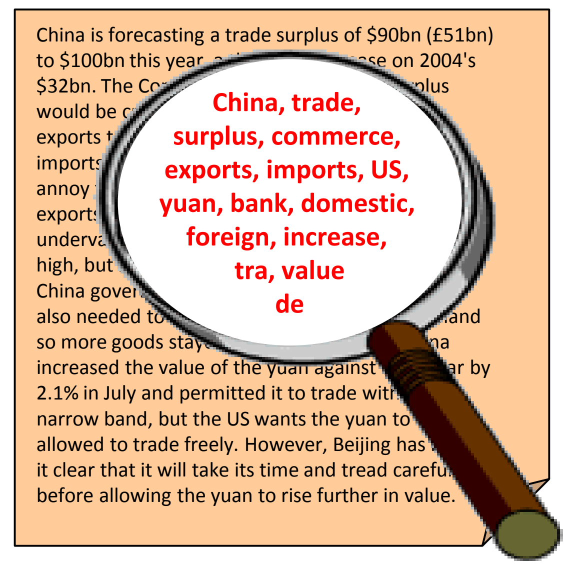
Image credit: Stanford CS231n

What's to Be Covered Today…

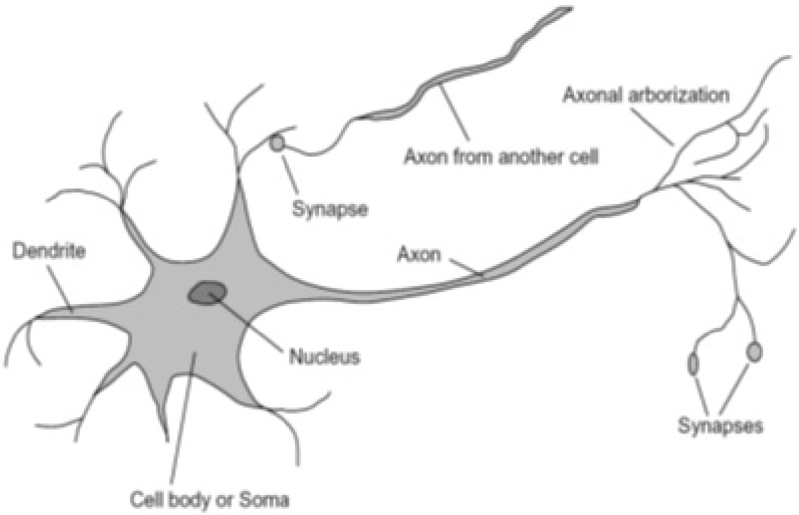
- Unsupervised vs. Supervised Learning
  - Clustering
  - Unsup. vs. Sup. Dimension Reduction
  - Training, testing, & validation
- Image Representation
  - Bag-of-Words Representation
  - Linear Classification
  - Intro to Neural Networks



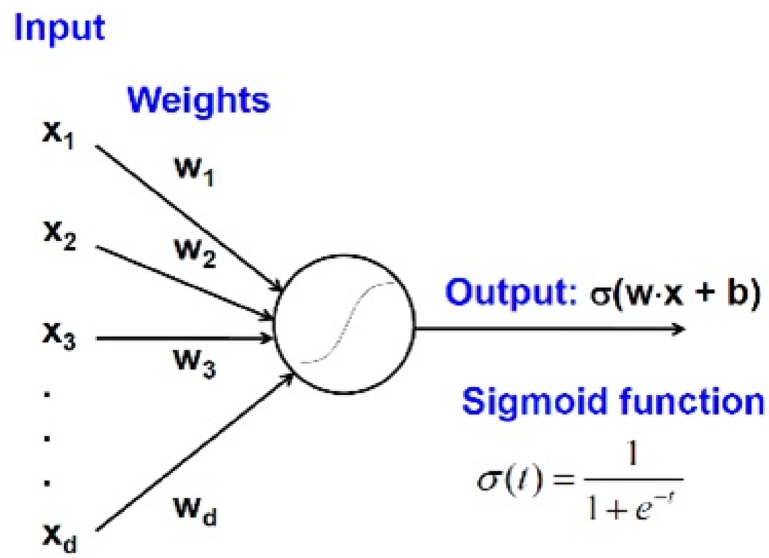




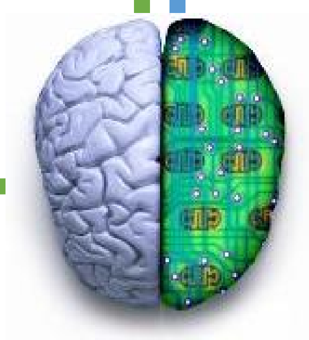
Biological neuron and Perceptrons



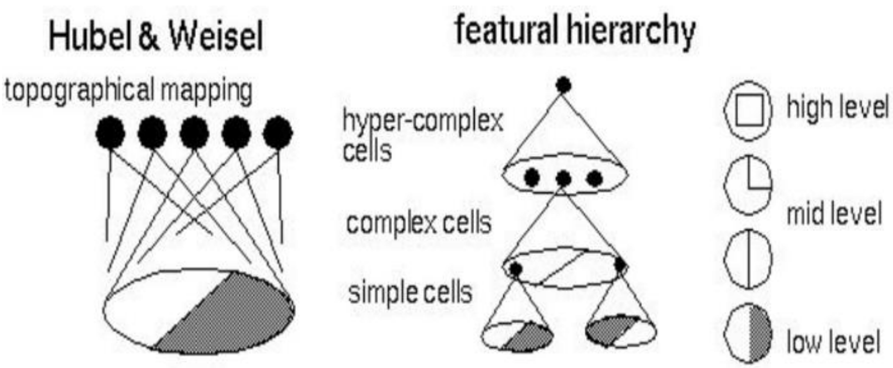
A biological neuron



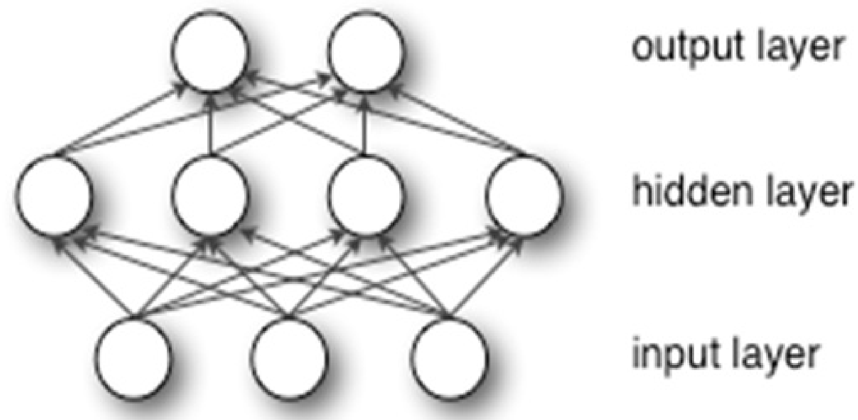
An artificial neuron (Perceptron)
- a linear classifier



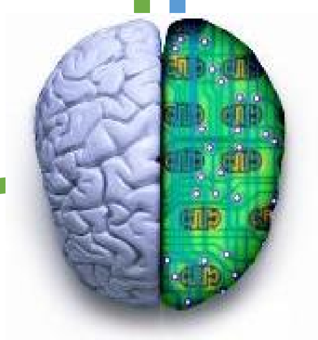
Hubel/Wiesel Architecture and Multi-layer Neural Network



Hubel and Weisel's architecture

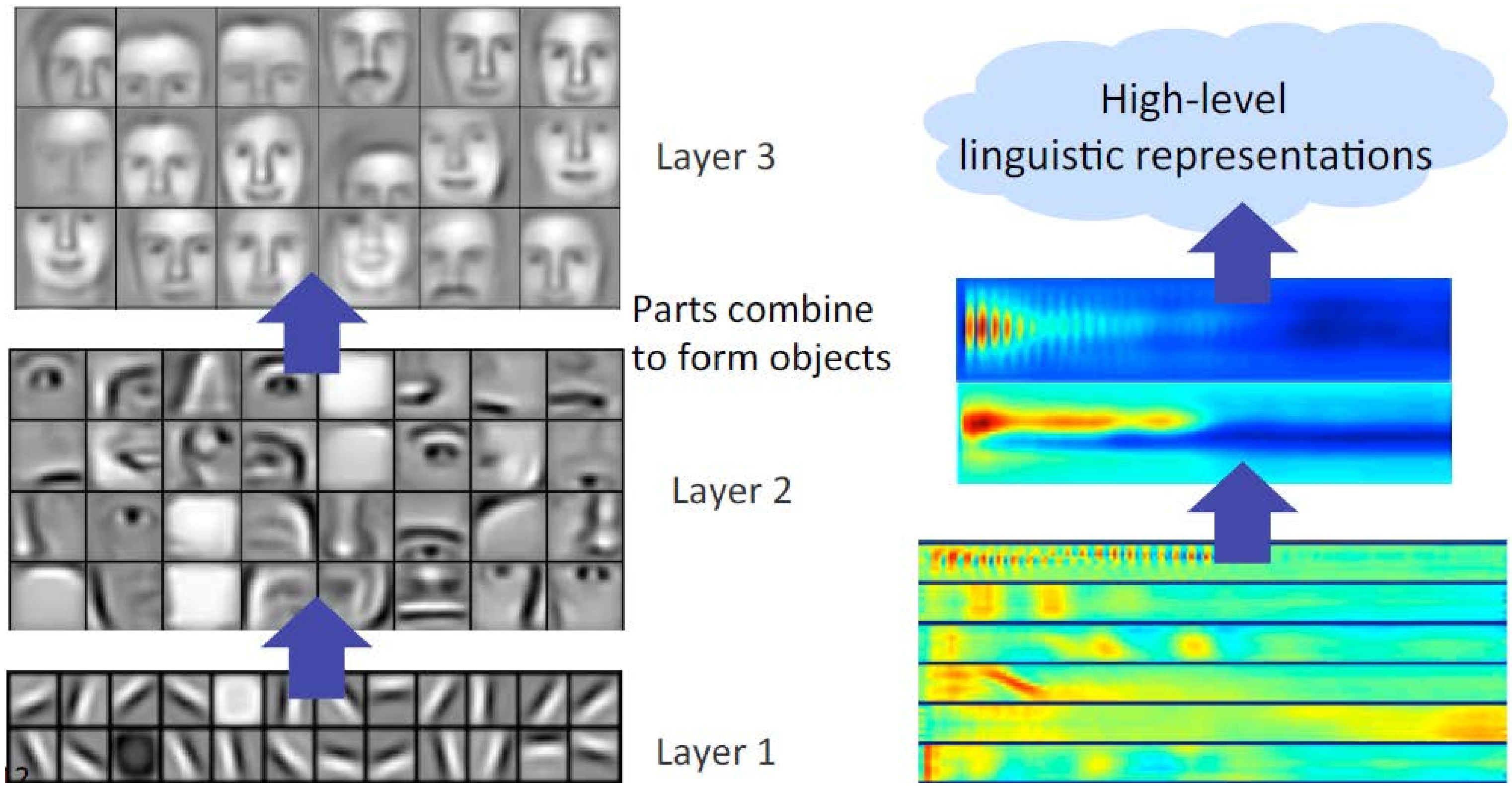


Multi-layer Neural Network
- A *non-linear* classifier



Hierarchical Learning

- Successive model layers learn deeper intermediate representations.



Revisit of Linear Classification

- Linear Classifier
  - Can be viewed as a parametric approach. Why?
  - Assuming that we need to recognize 10 object categories of interest (e.g., CIFAR10).
  - Let's take the input image as **x**, and the linear classifier as **W**. We hope to see that **y** = **Wx** + **b** as a 10-dimensional output indicating the score for each class.

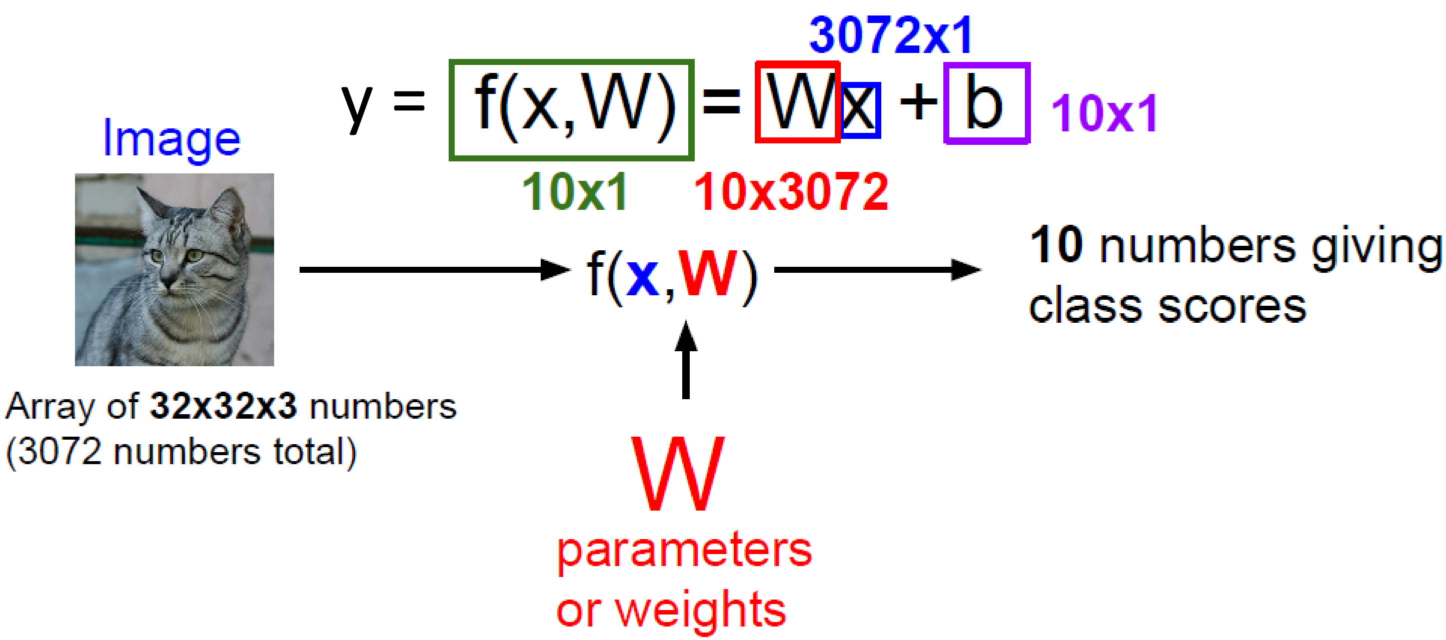
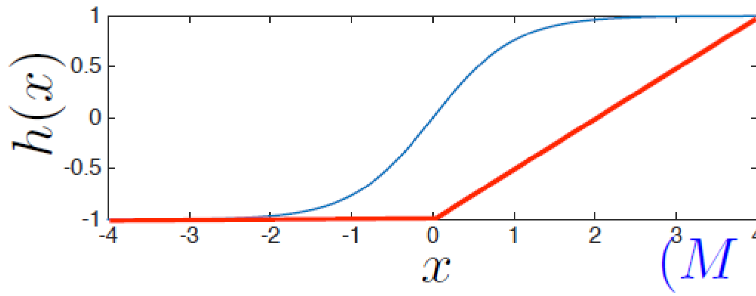
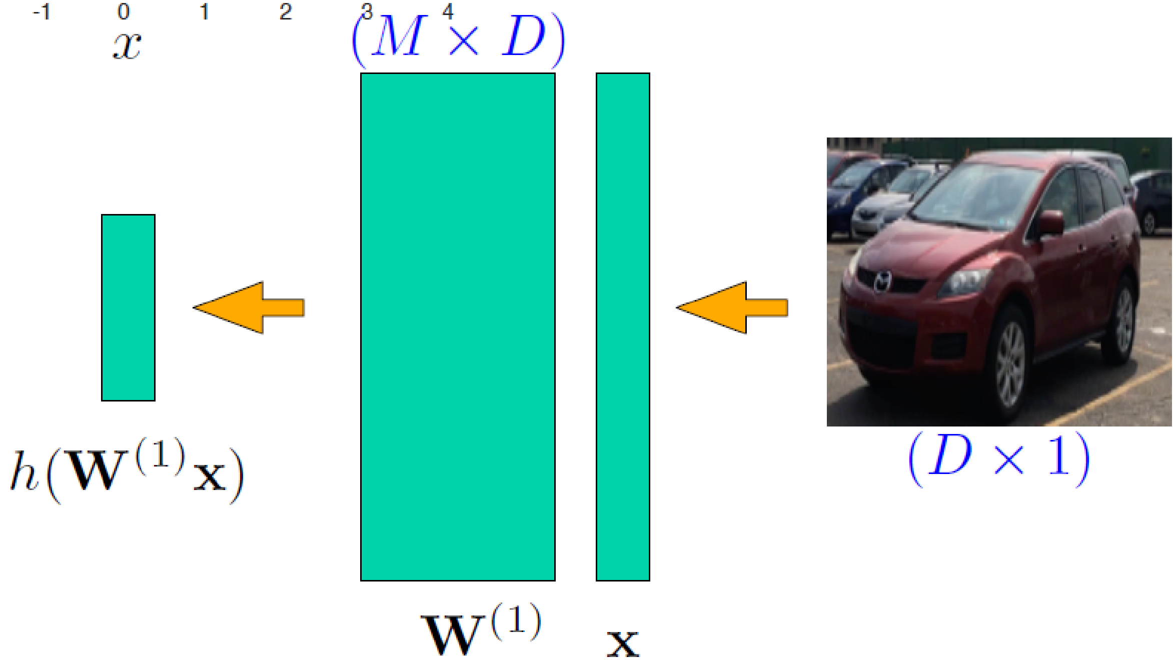


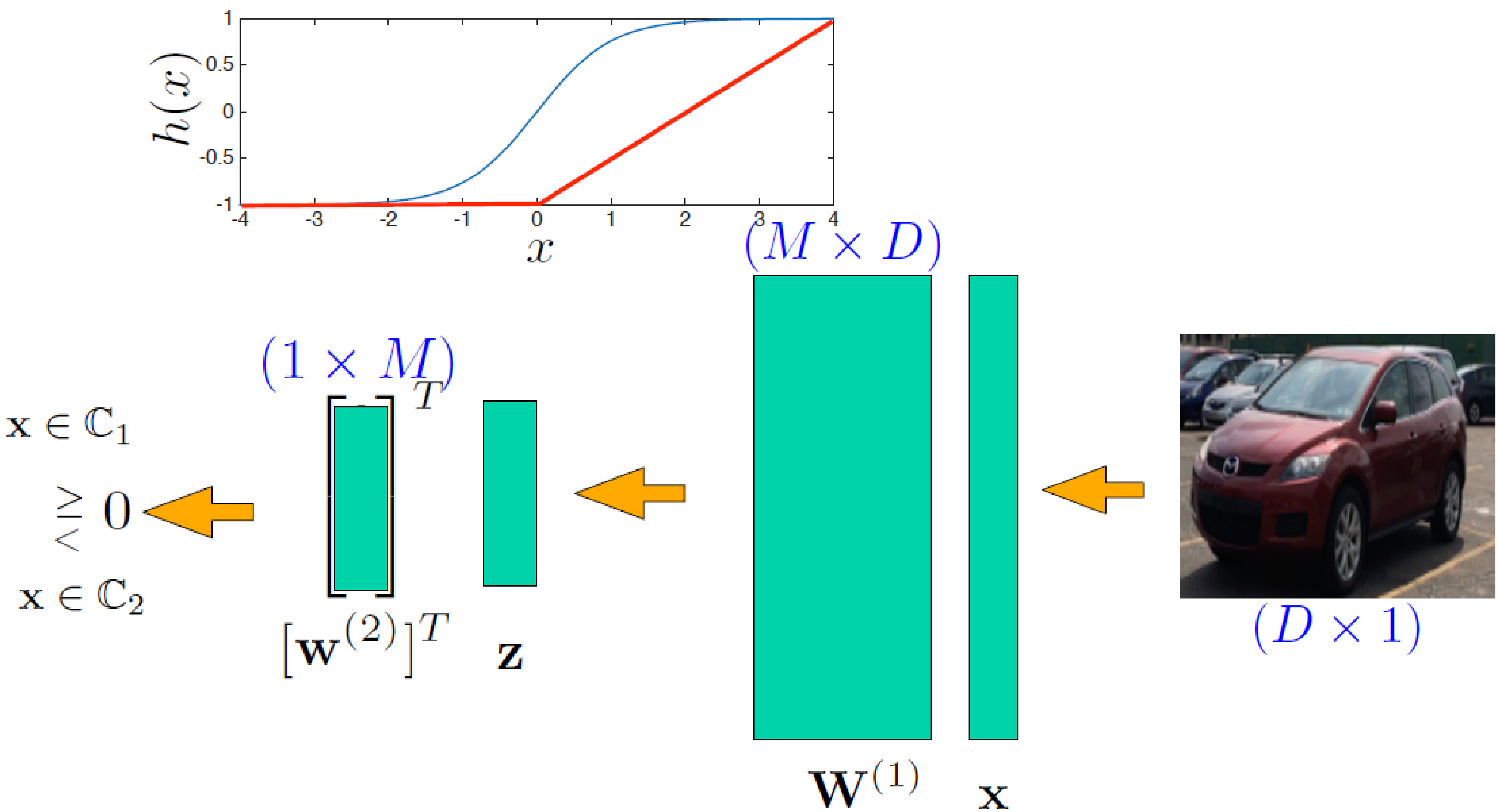
Image credit: Stanford CS231n

Multi-Layer Perceptron: A Nonlinear Classifier

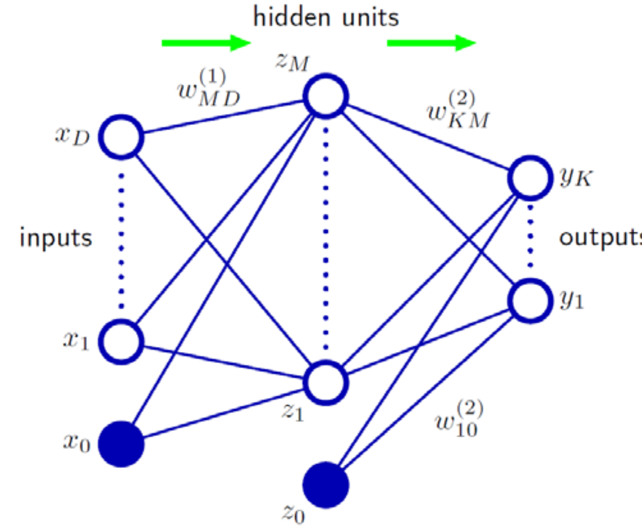




Multi-Layer Perceptron: A Nonlinear Classifier (cont'd)



Layer 1 in MLP



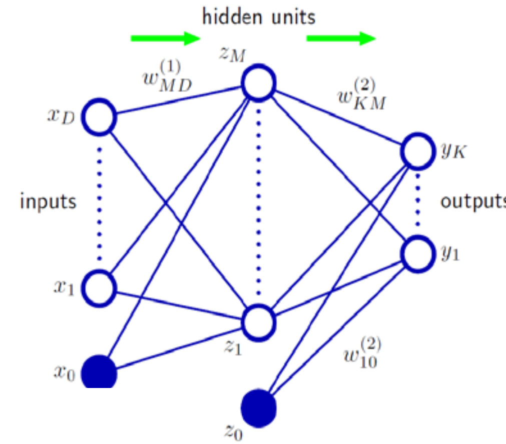
$$\mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_M \end{bmatrix} \leftarrow \begin{bmatrix} h[\mathbf{x}^T \mathbf{w}_1^{(1)}] \\ \vdots \\ h[\mathbf{x}^T \mathbf{w}_M^{(1)}] \end{bmatrix}$$

$h() = \text{non-linear function}$

$[\mathbf{w}_1^{(1)}, \ldots, \mathbf{w}_M^{(1)}] = \text{1st layer's } D \times M \text{ weights}$

$\mathbf{x} = D \times 1 \text{ raw input}$

Layer 2 in MLP



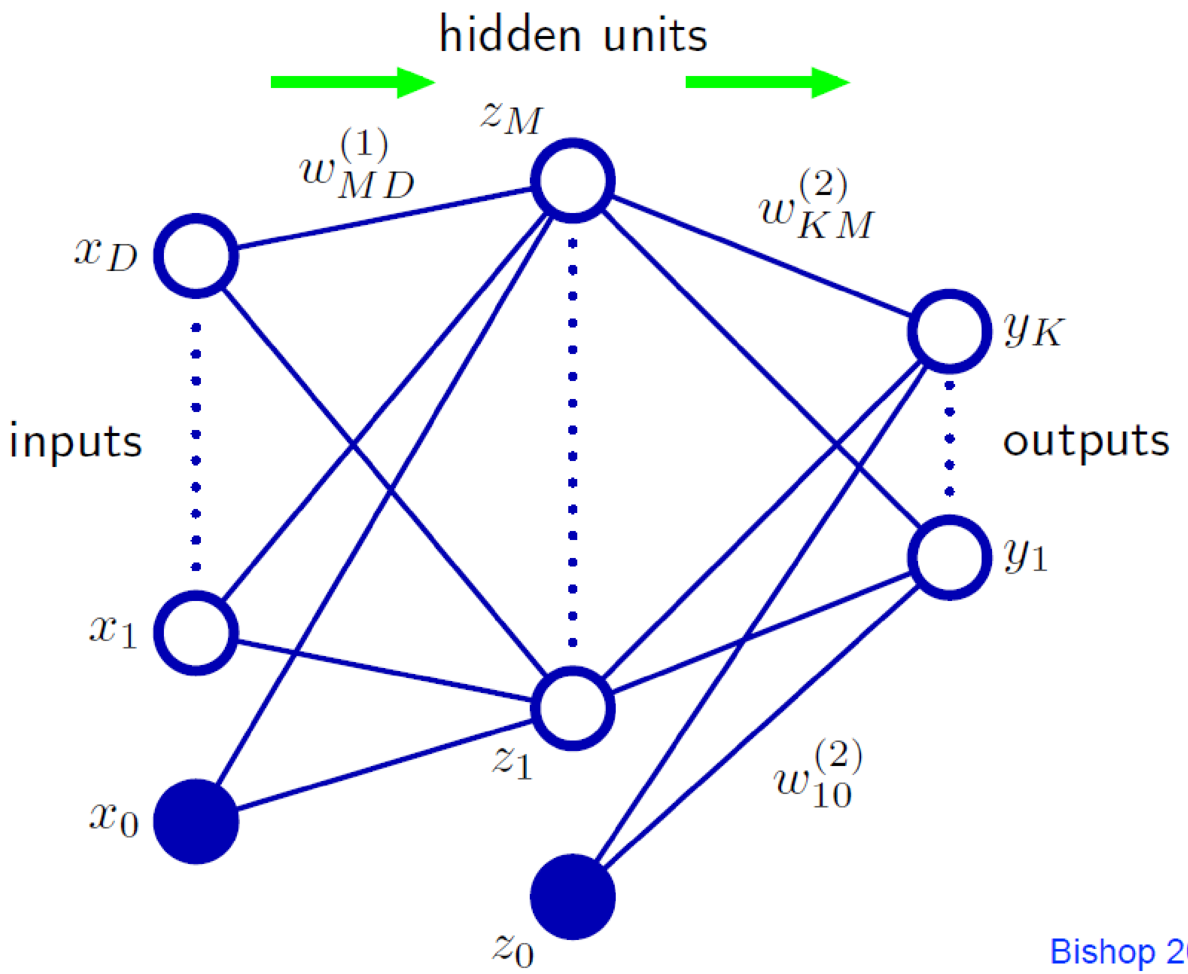
$\mathbf{z} \in \mathbb{R}^M$

$\mathbf{x} \in \mathbb{R}^D$

$$\mathbf{z}^T\mathbf{w}^{(2)} \quad \begin{array}{c} \mathbf{z} \in \mathbb{C}_1 \\ \geq \\ < \\ \mathbf{z} \in \mathbb{C}_2 \end{array} \quad 0$$

$\mathbf{z} = M \times 1$ output of layer 1
$\mathbf{w}^{(2)} = $ 2nd layer's $M \times 1$ weight vector

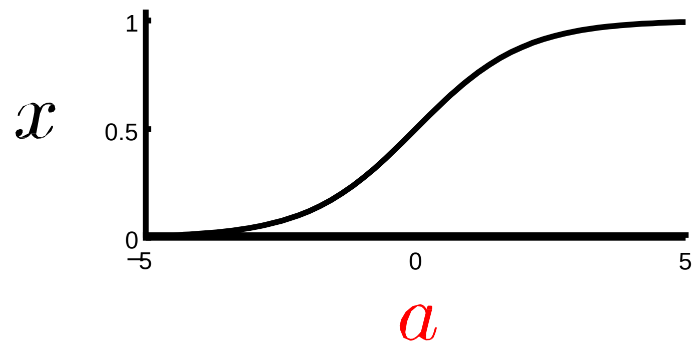
Multi-Layer Perceptron: A Nonlinear Classifier (cont'd)



Bishop 2006

Let’s Get a Closer Look…

- A single neuron

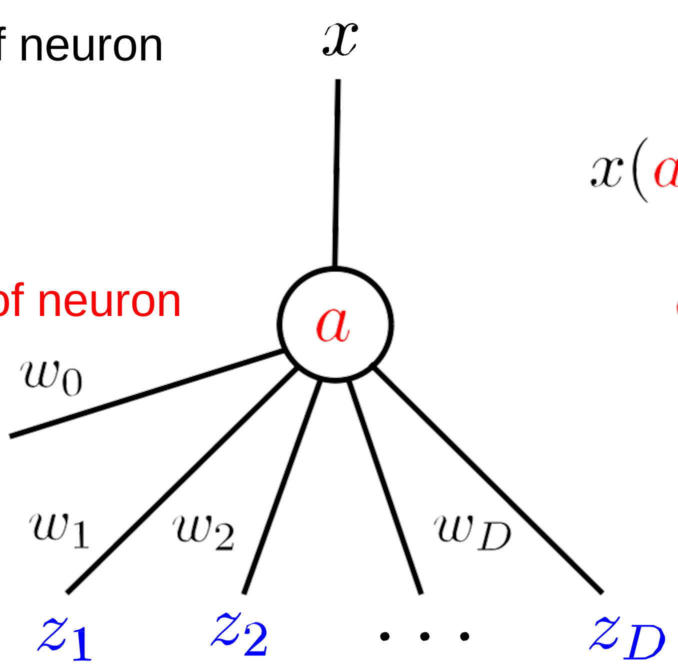


$$x(a) = \frac{1}{1+\exp(-a)} \qquad x \in (0, 1)$$

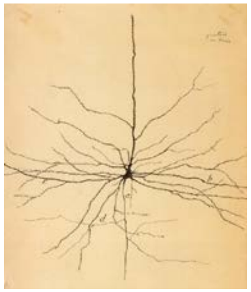
$$a = w_0 + \sum_{d=1}^{D} w_d z_d$$
$$= \sum_{d=0}^{D} w_d z_d$$

output of neuron

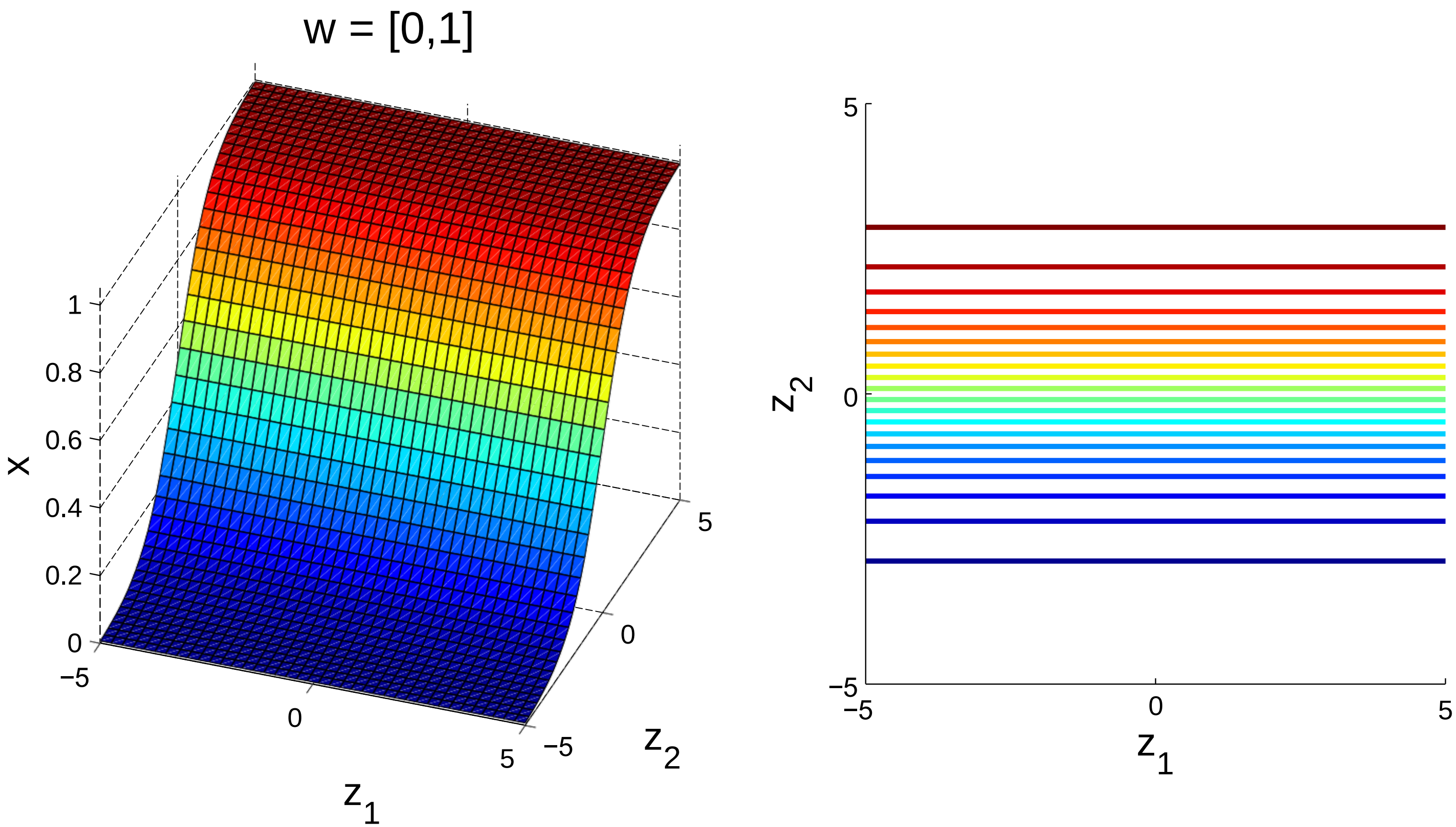
activity of neuron



inputs to neuron

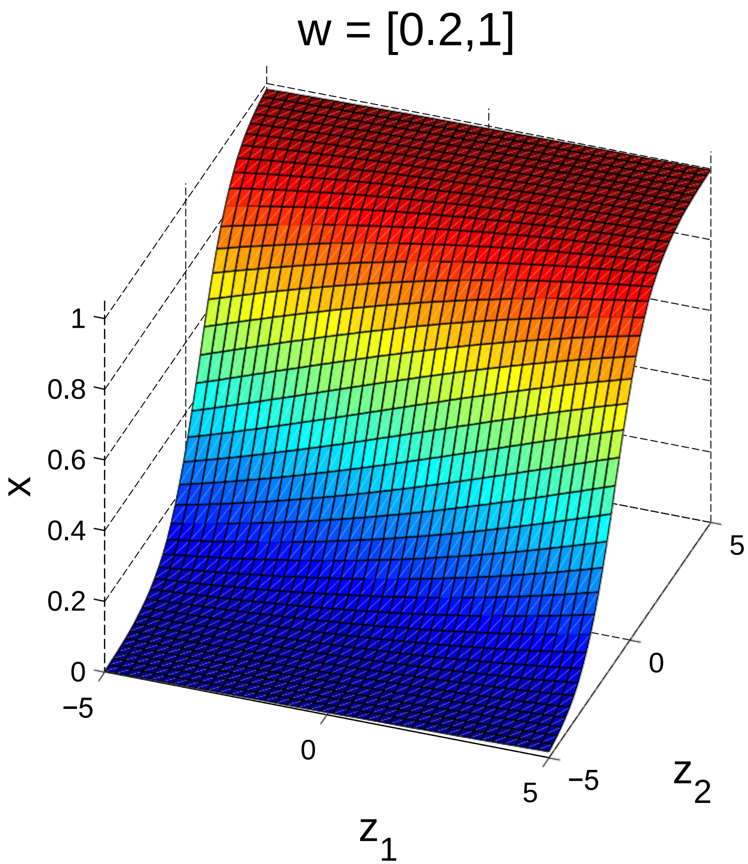


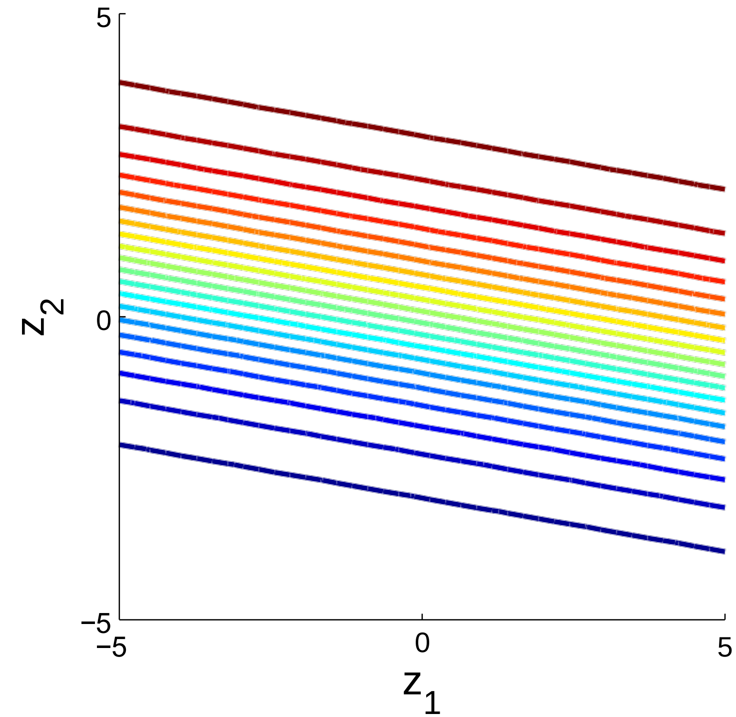
Input-Output Function of a Single Neuron



$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

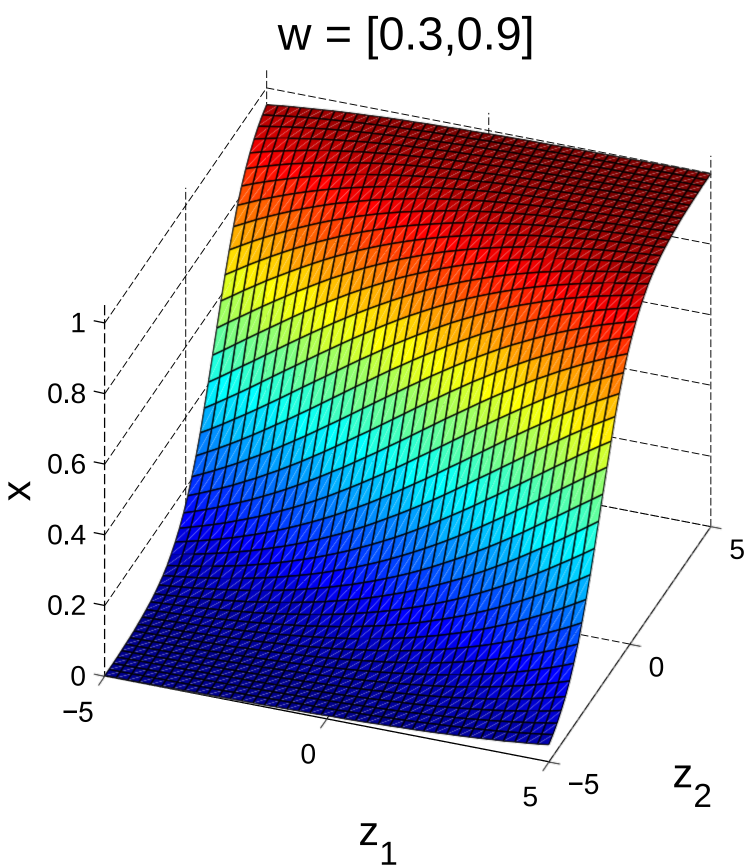
Input-Output Function of a Single Neuron

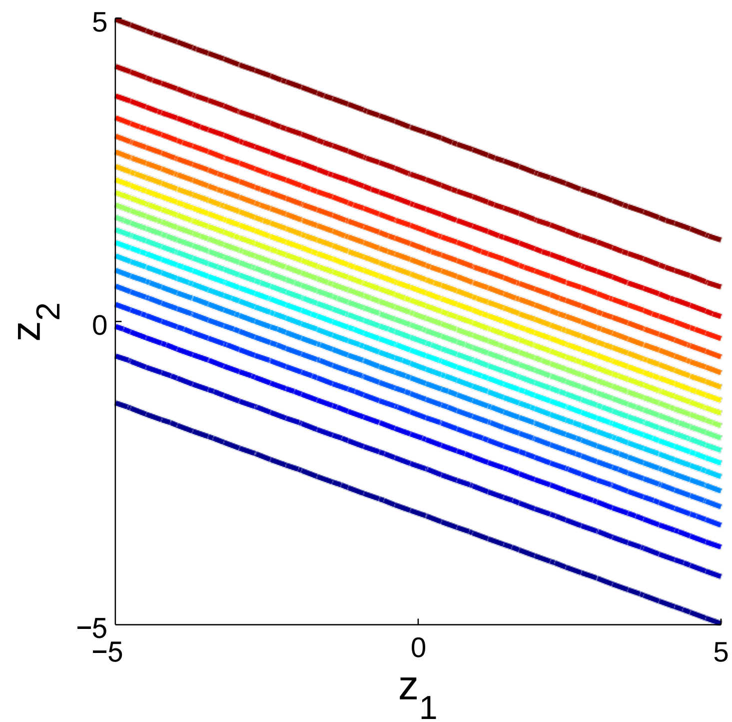




$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

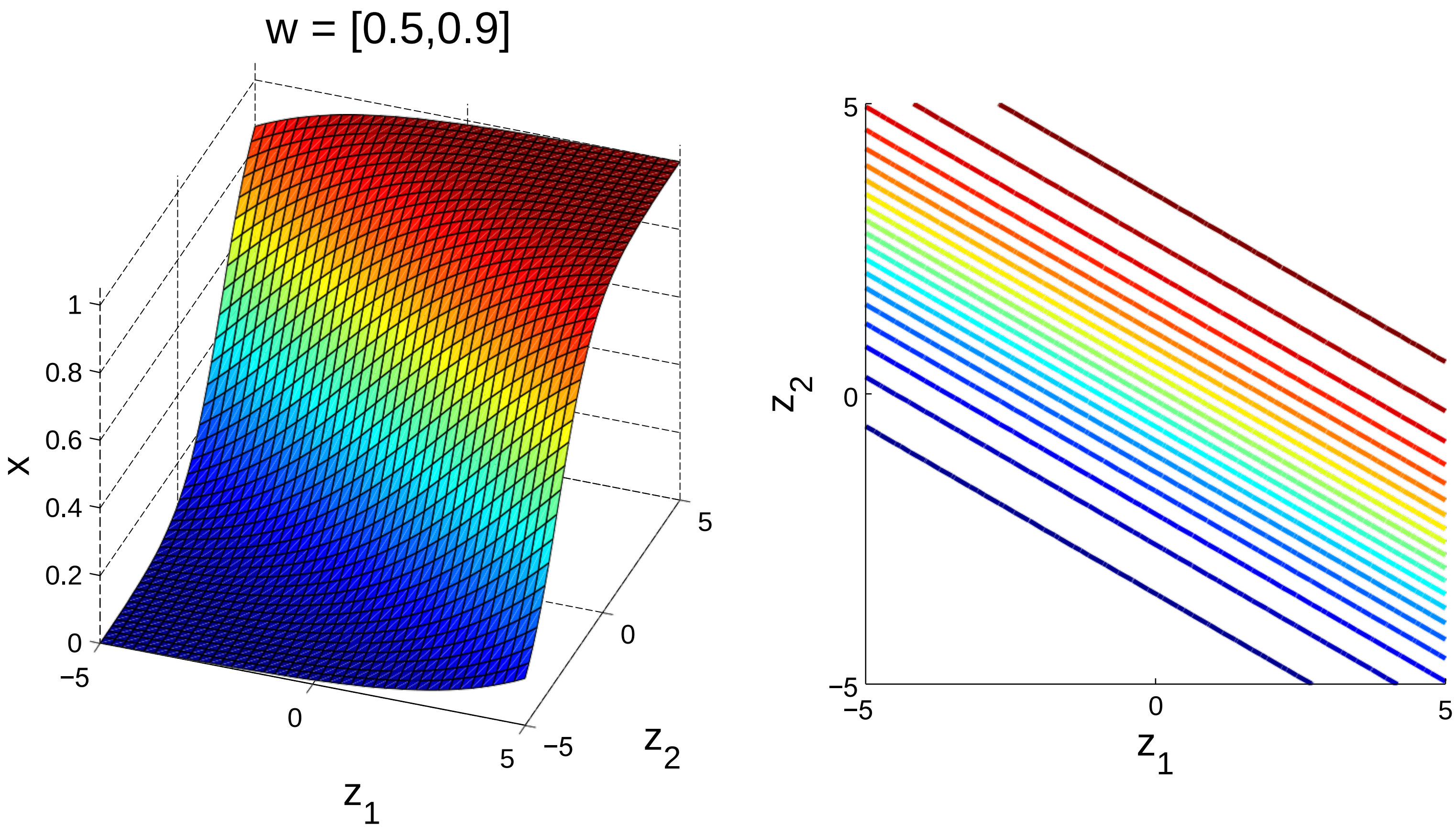
Input-Output Function of a Single Neuron





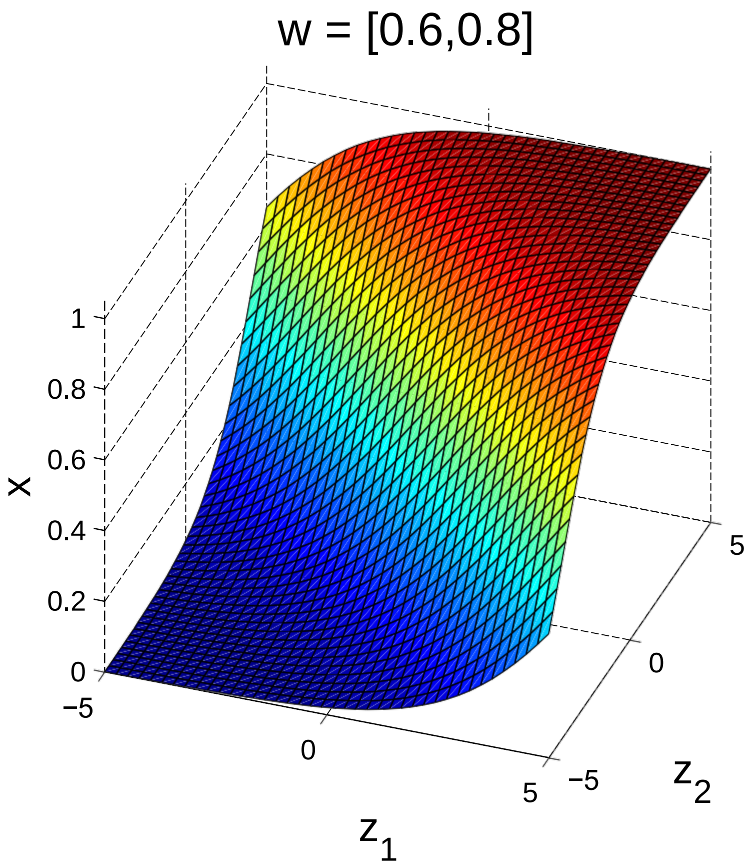
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

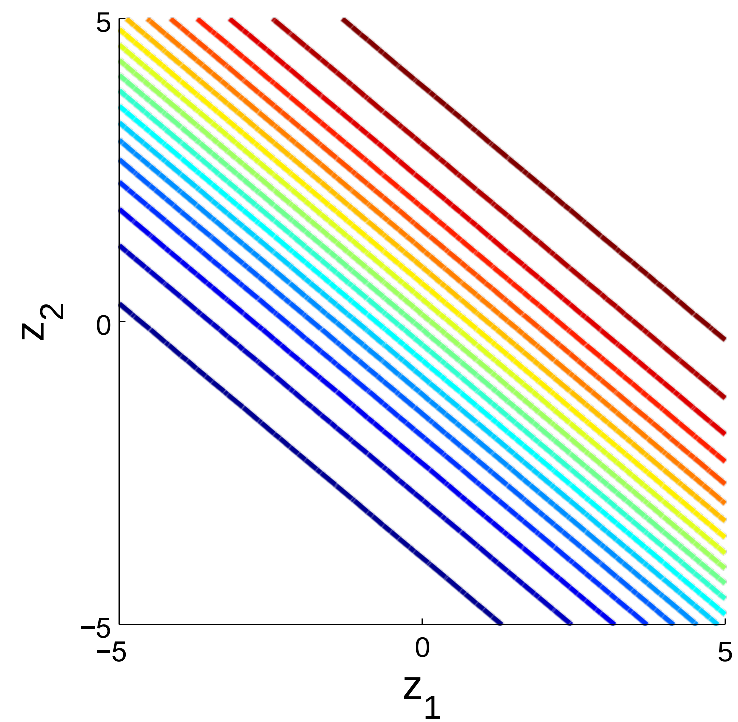
Input-Output Function of a Single Neuron



$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

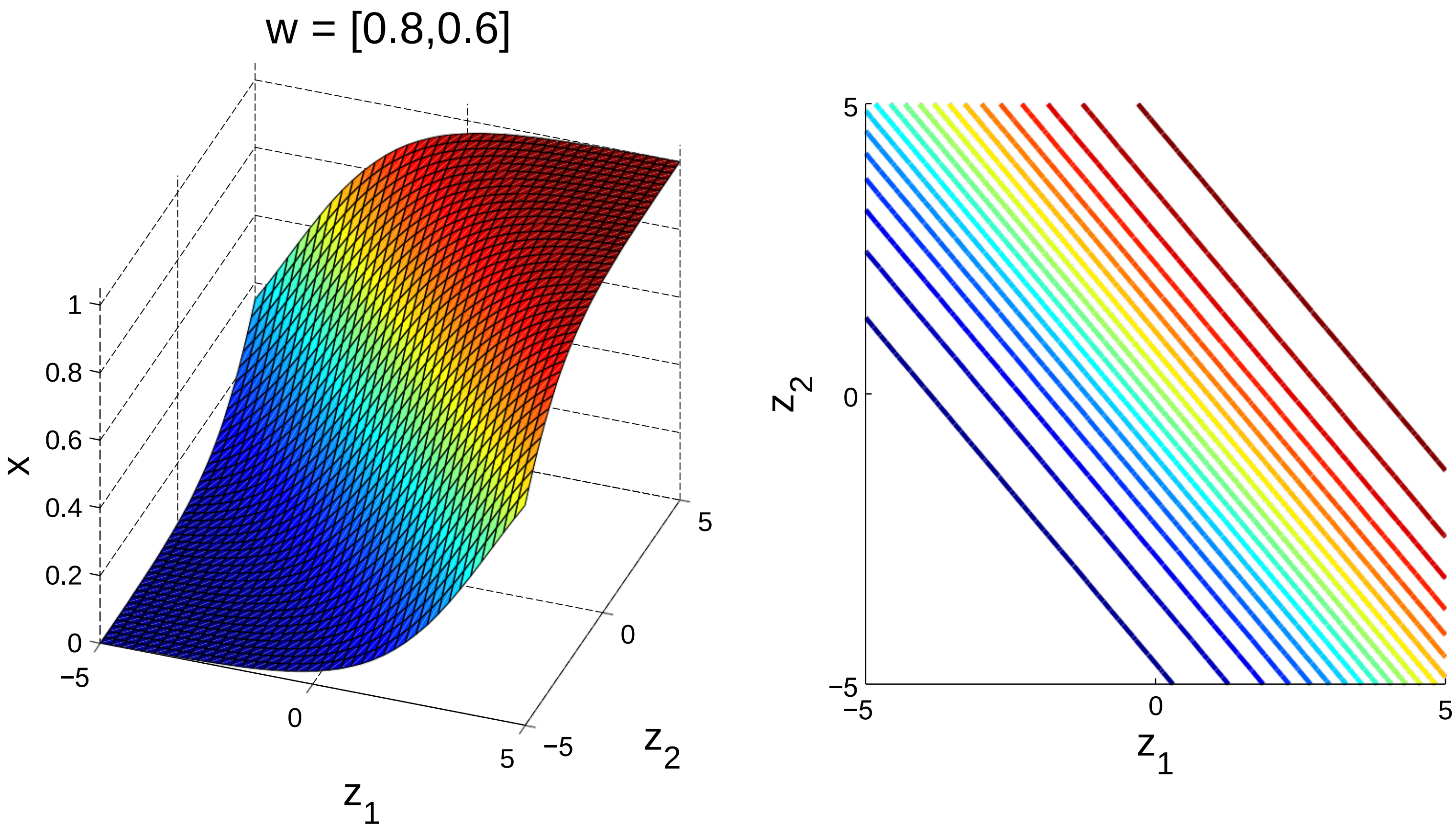
Input-Output Function of a Single Neuron





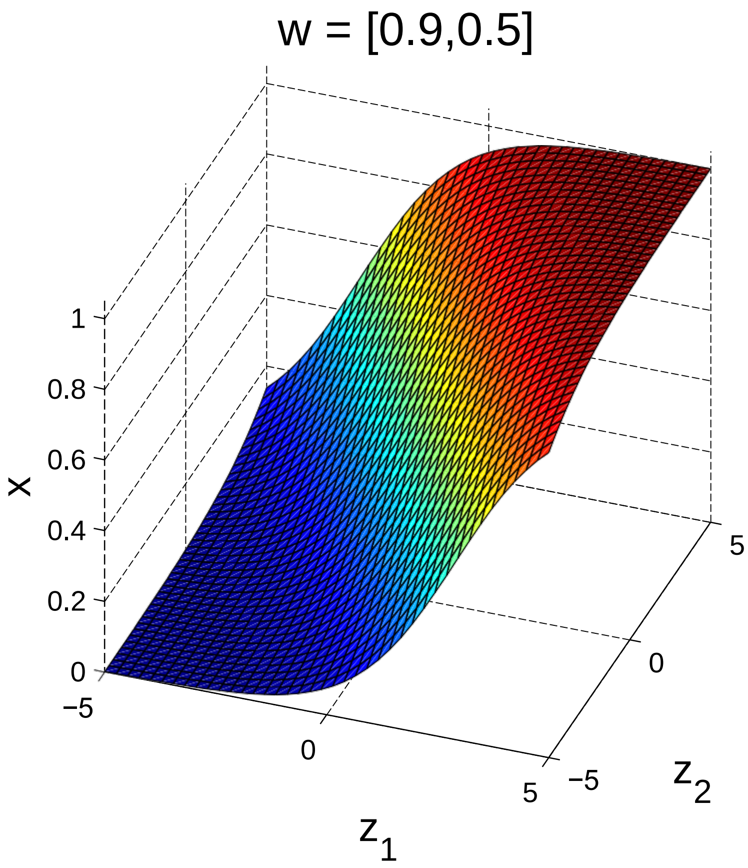
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

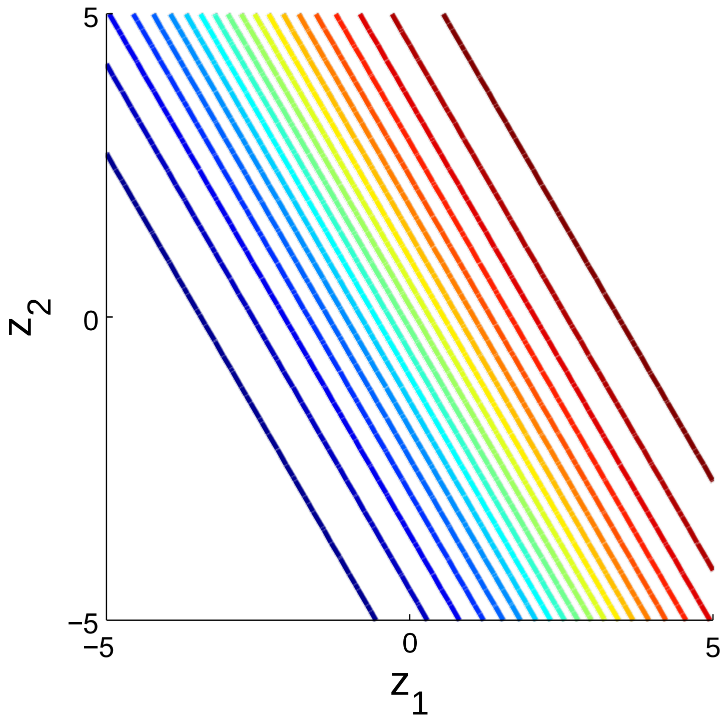
Input-Output Function of a Single Neuron



$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

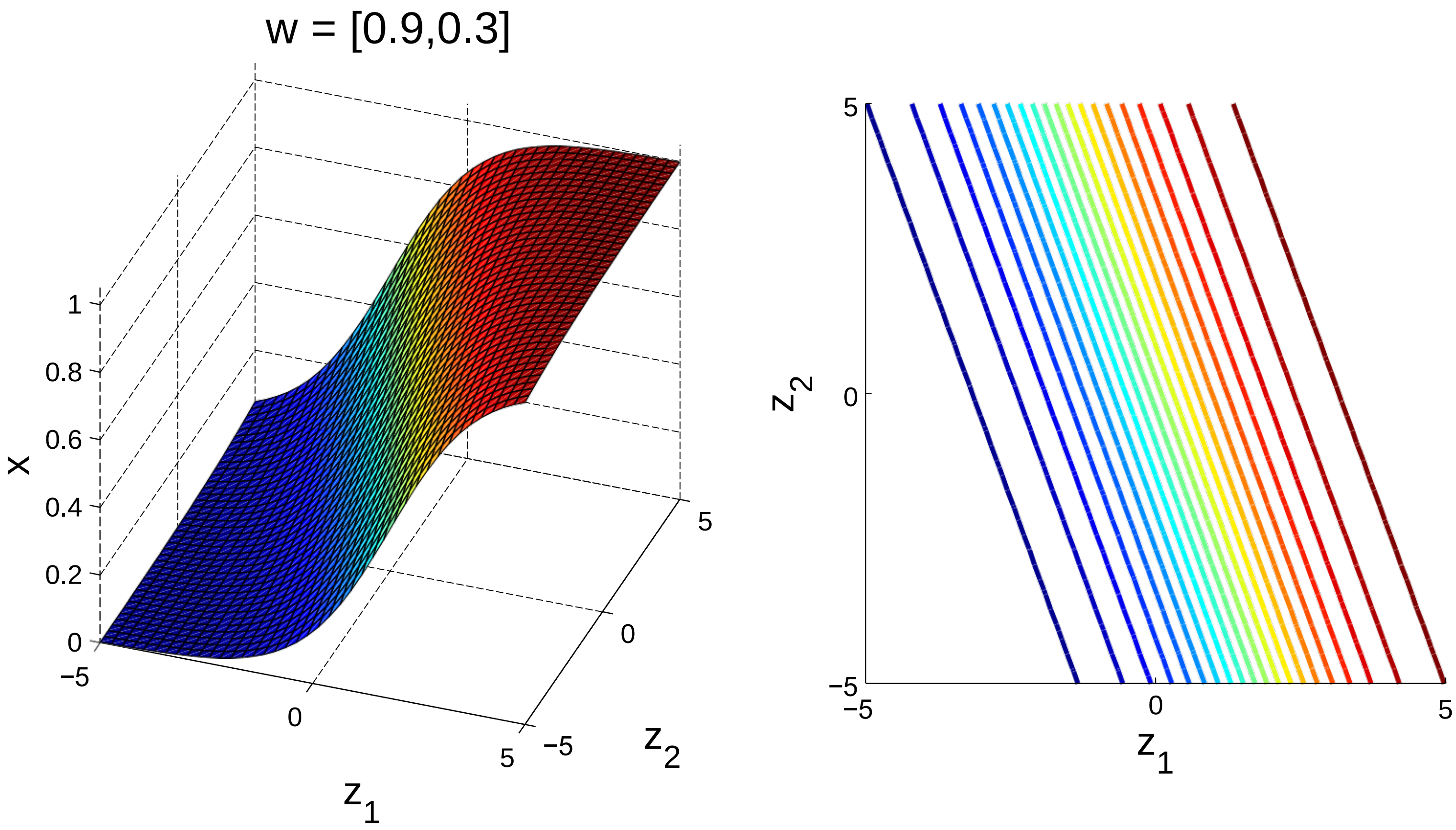
Input-Output Function of a Single Neuron





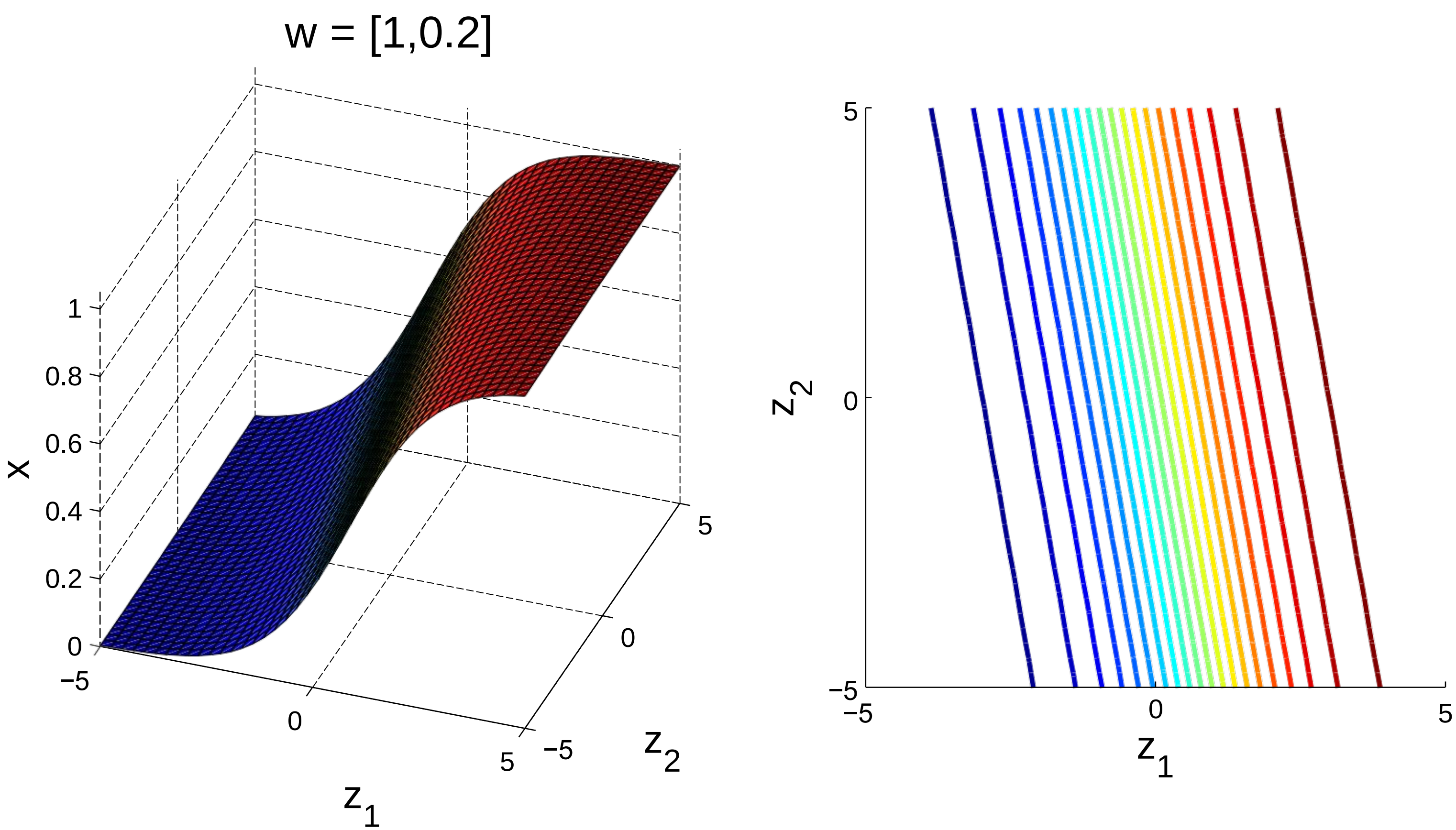
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



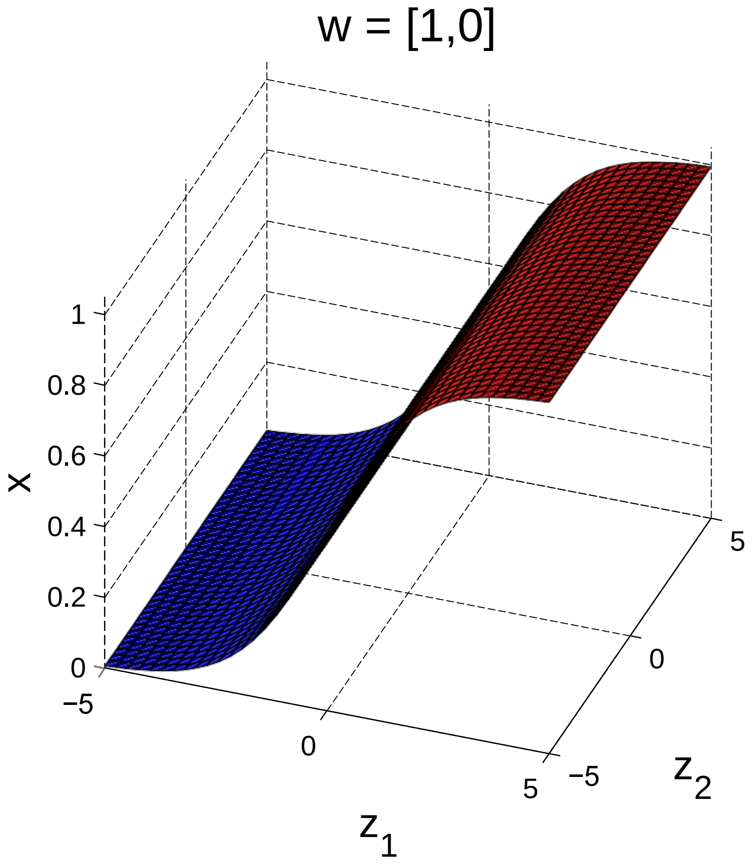
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

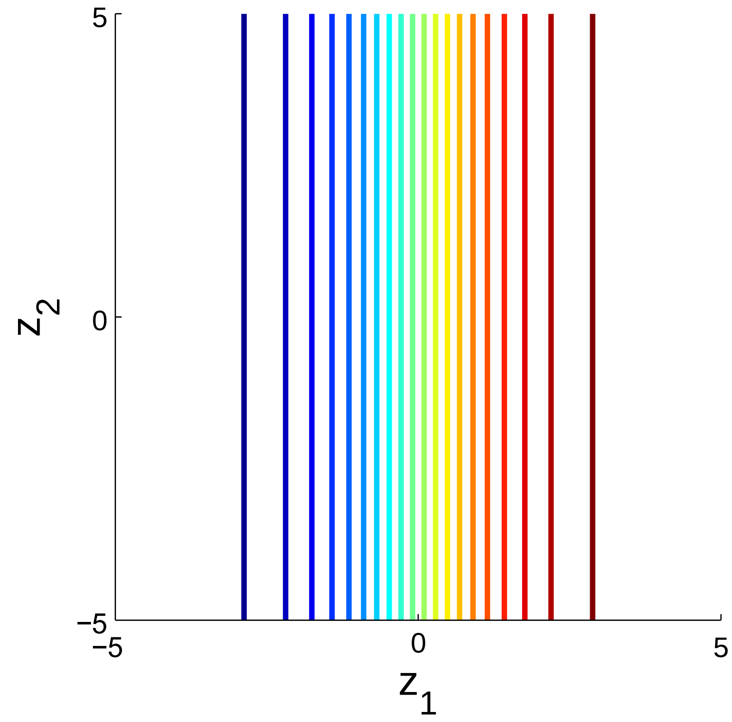
Input-Output Function of a Single Neuron



$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

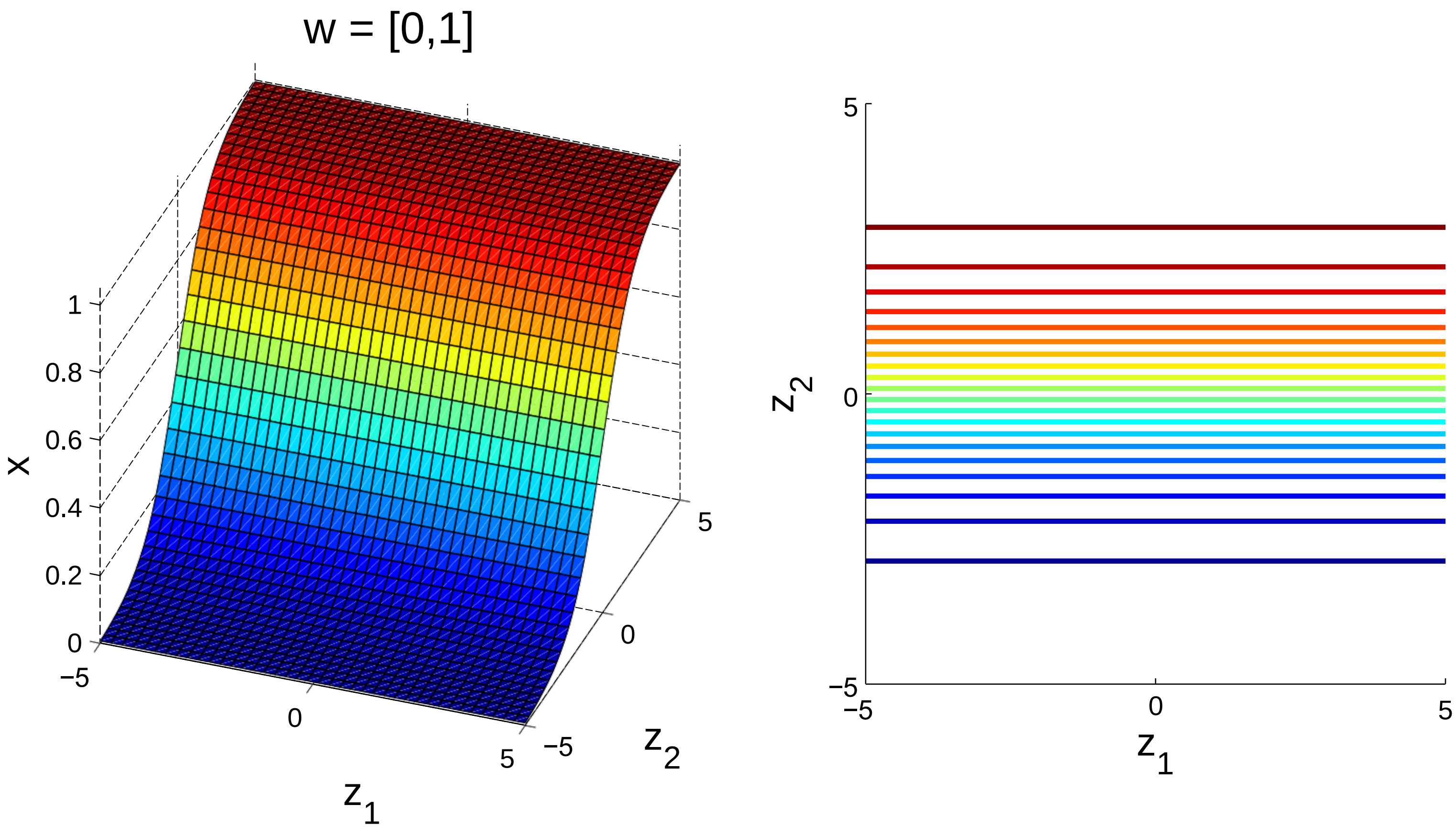
Input-Output Function of a Single Neuron





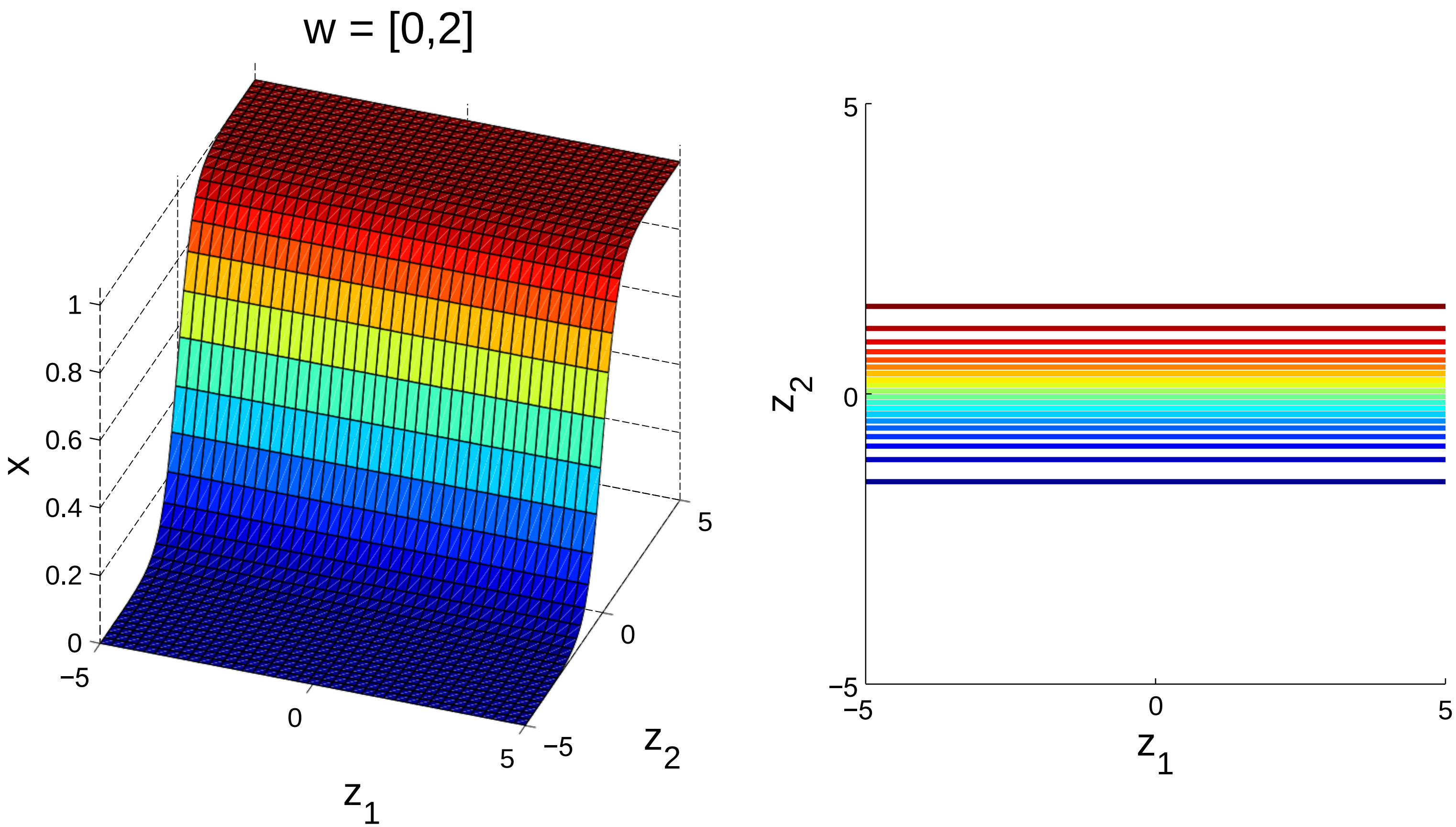
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron (cont'd)



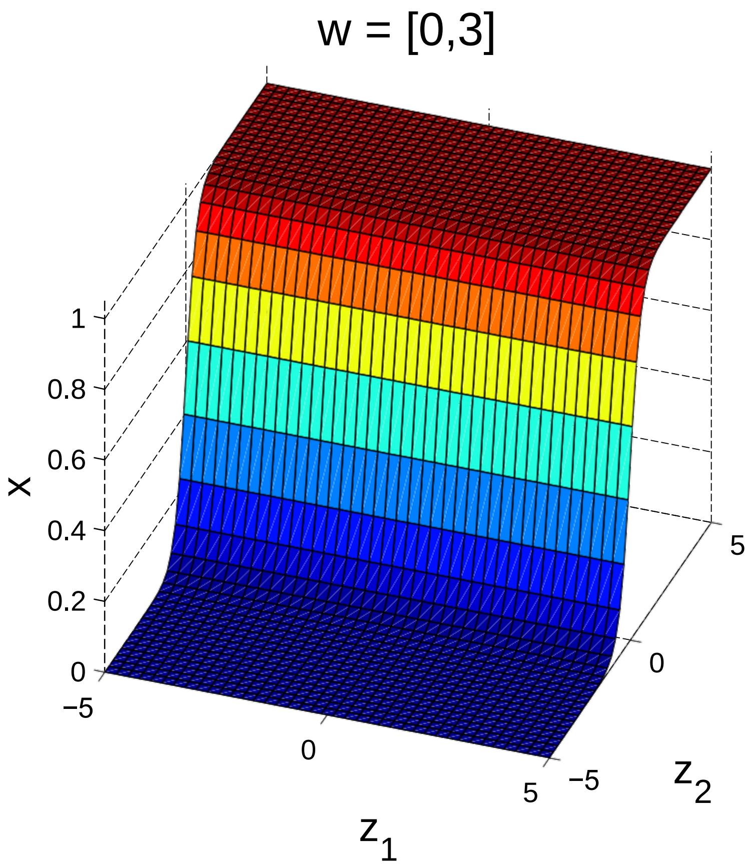
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

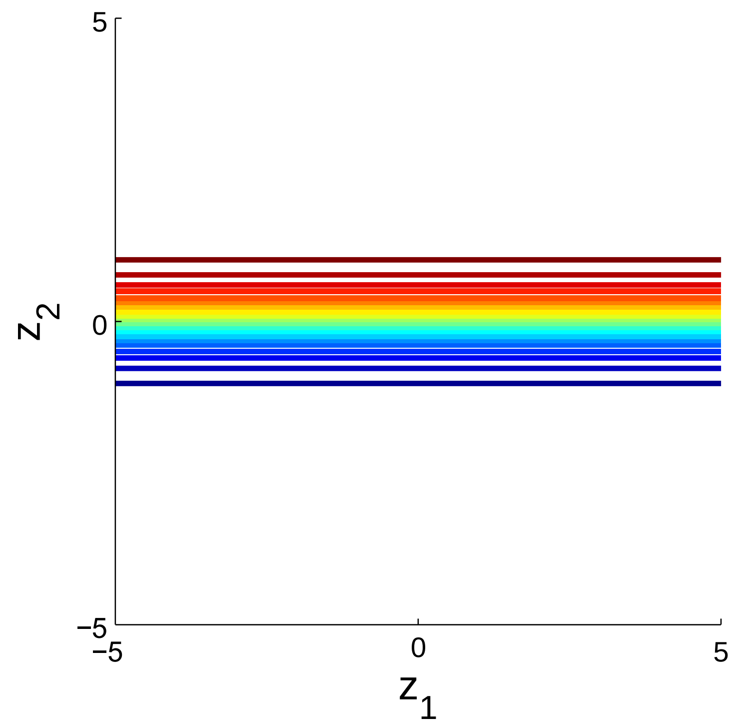
Input-Output Function of a Single Neuron (cont'd)



$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

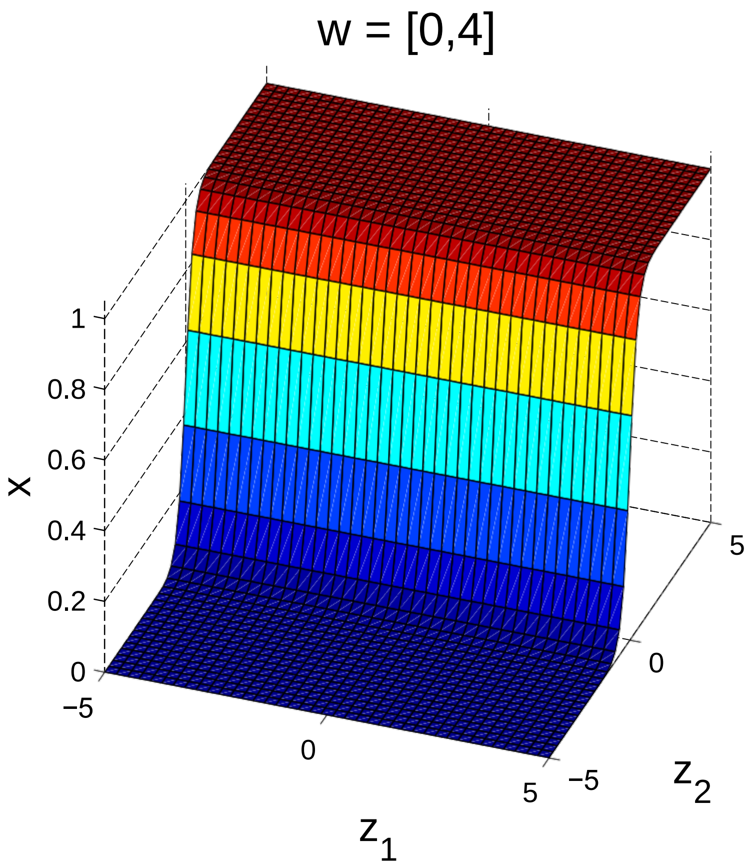
Input-Output Function of a Single Neuron (cont'd)

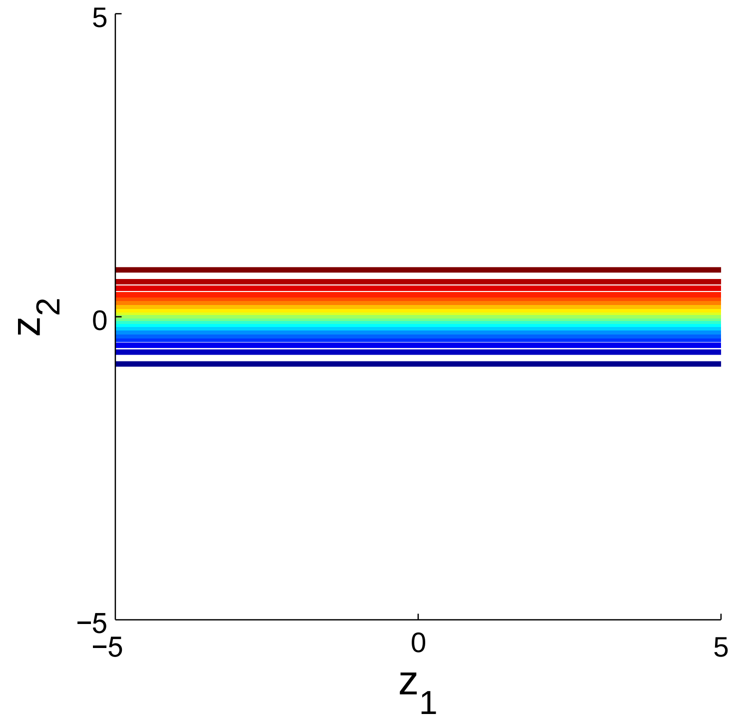




$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

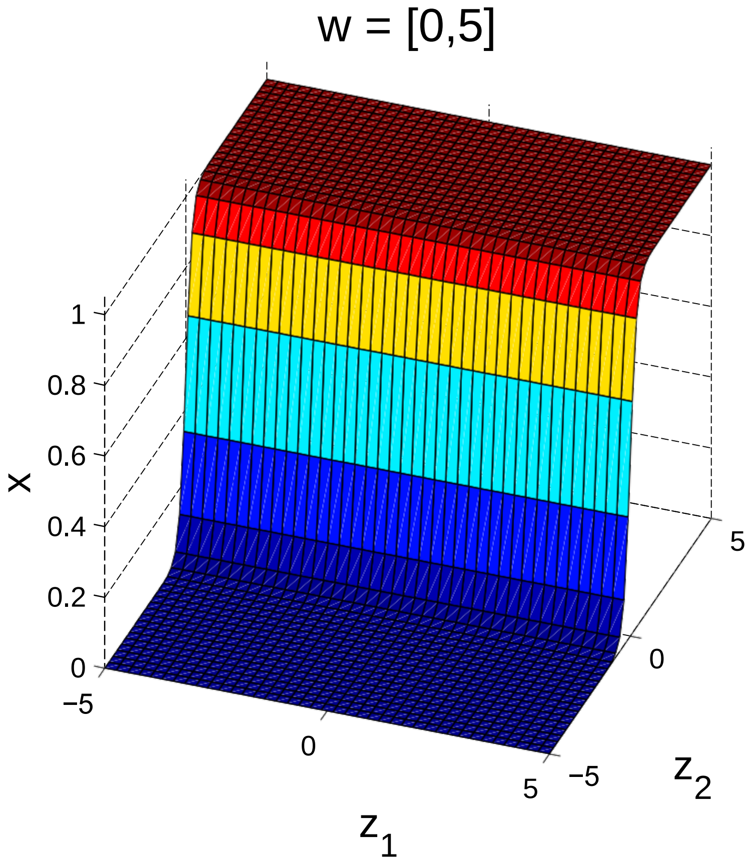
Input-Output Function of a Single Neuron (cont'd)

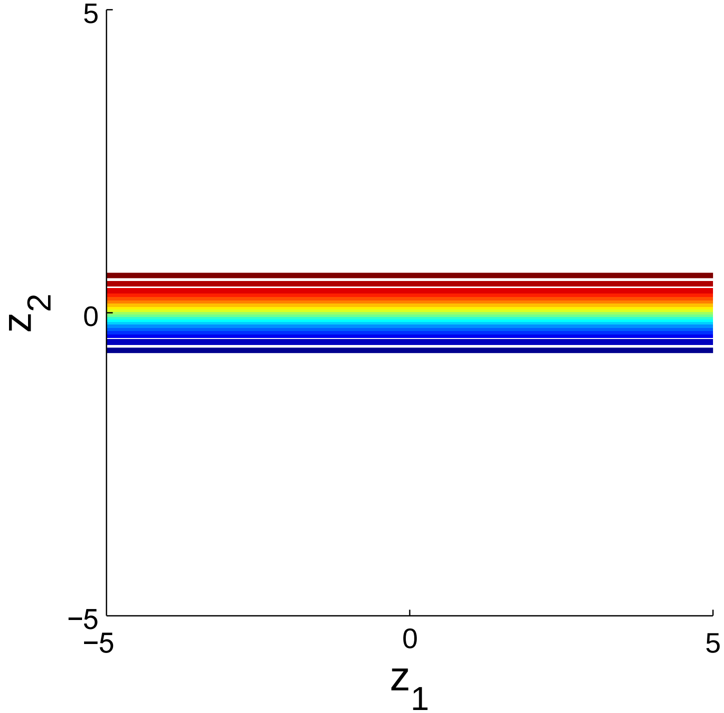




$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

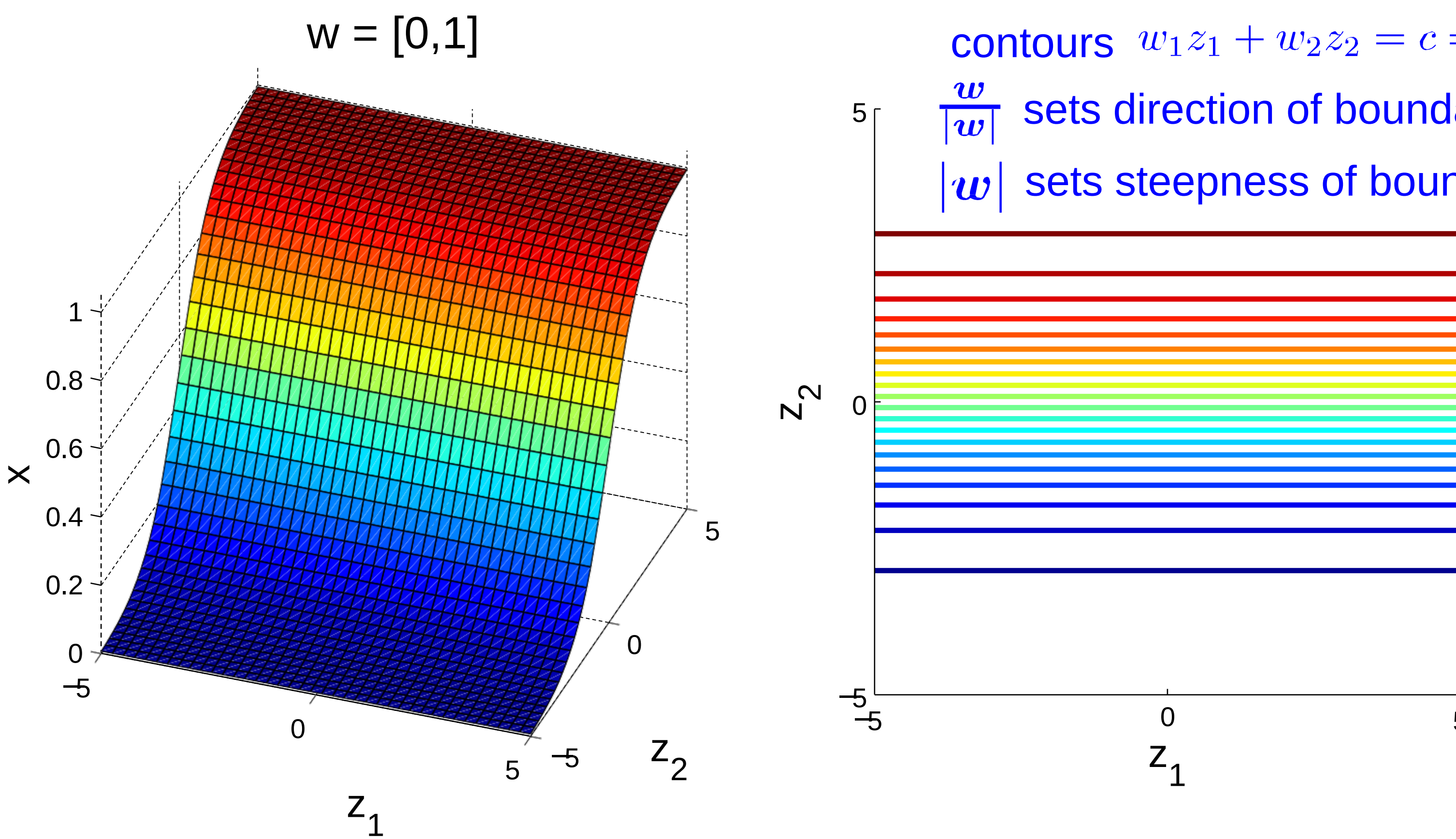
Input-Output Function of a Single Neuron (cont'd)





$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron (cont'd)



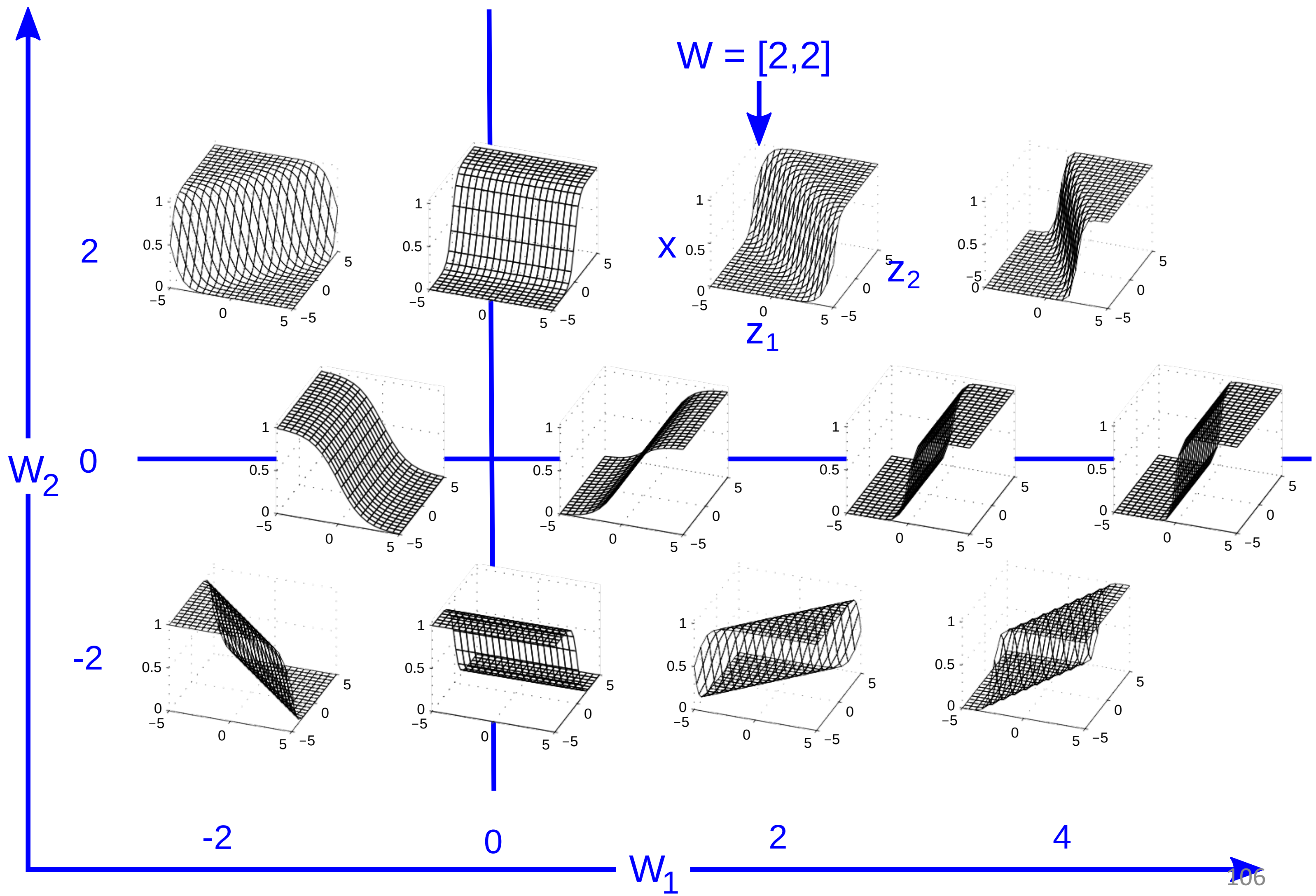
contours $w_1 z_1 + w_2 z_2 = c = \boldsymbol{w}^\top \boldsymbol{z}$

$\frac{\boldsymbol{w}}{|\boldsymbol{w}|}$ sets direction of boundary

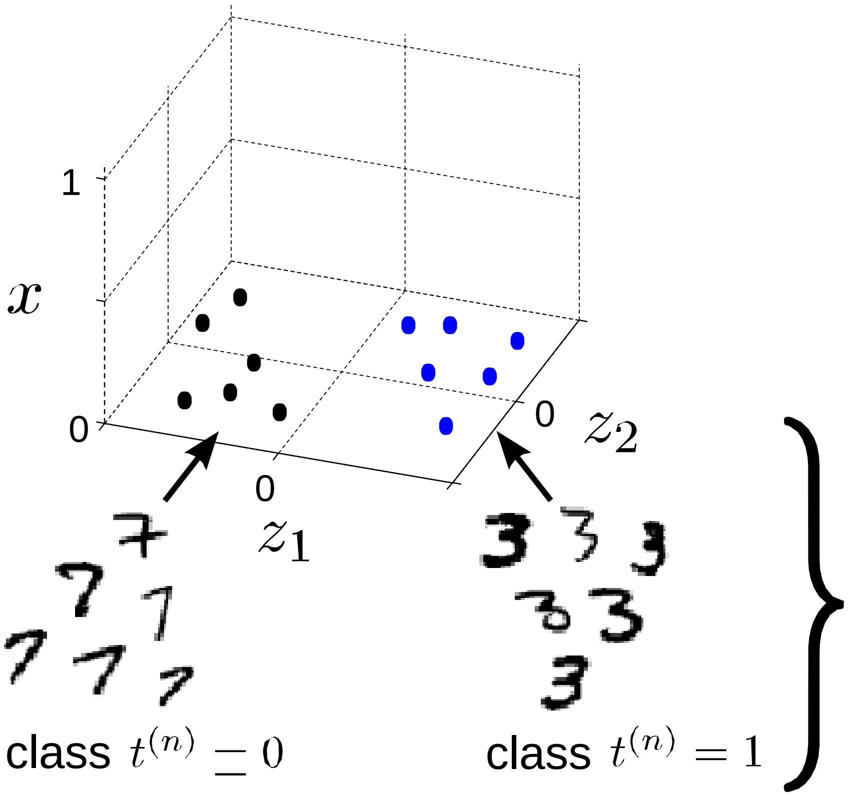
$|\boldsymbol{w}|$ sets steepness of boundary

$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Weight Space of a Single Neuron



Training a Single Neuron

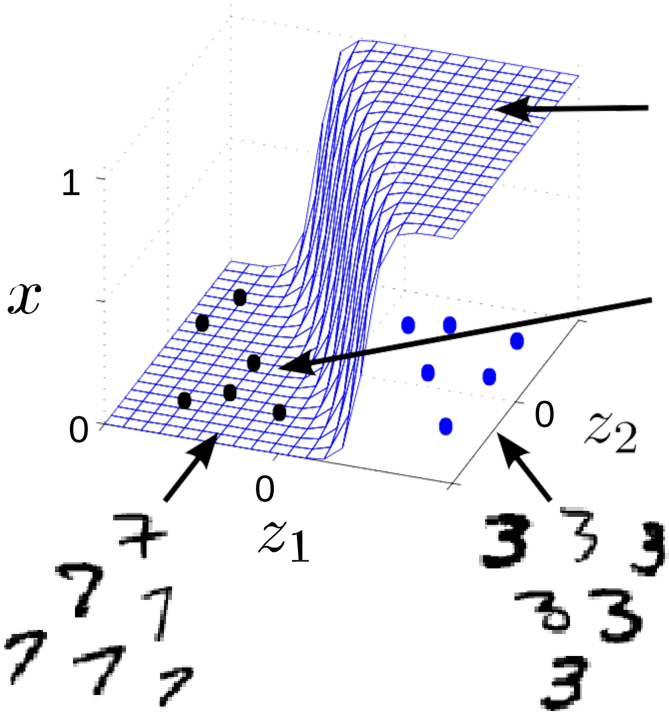


training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

Training a Single Neuron



desired result of training:

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 1$ for $t^{(n)} = 1$

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 0$ for $t^{(n)} = 0$

class $t^{(n)} = 0$

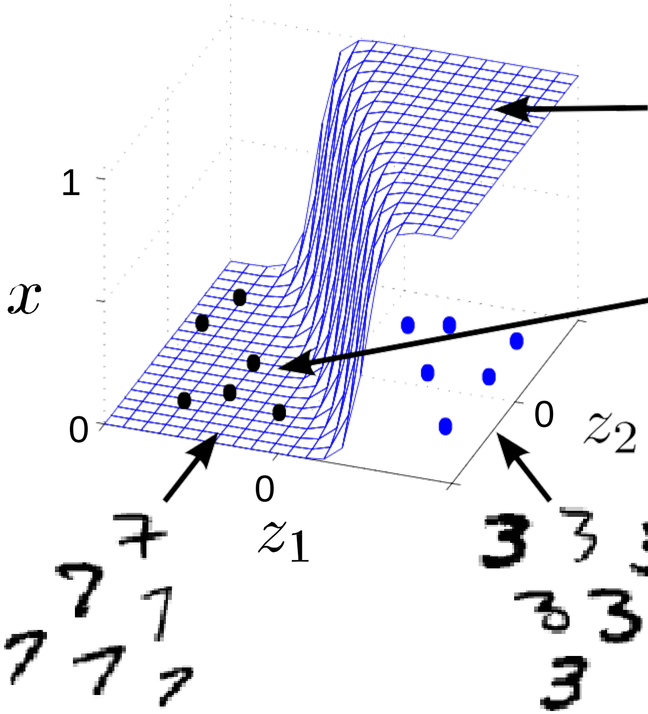
class $t^{(n)} = 1$

training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

Training a Single Neuron



desired result of training:

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 1$ for $t^{(n)} = 1$

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 0$ for $t^{(n)} = 0$

class $t^{(n)} = 0$ class $t^{(n)} = 1$

training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

objective function:

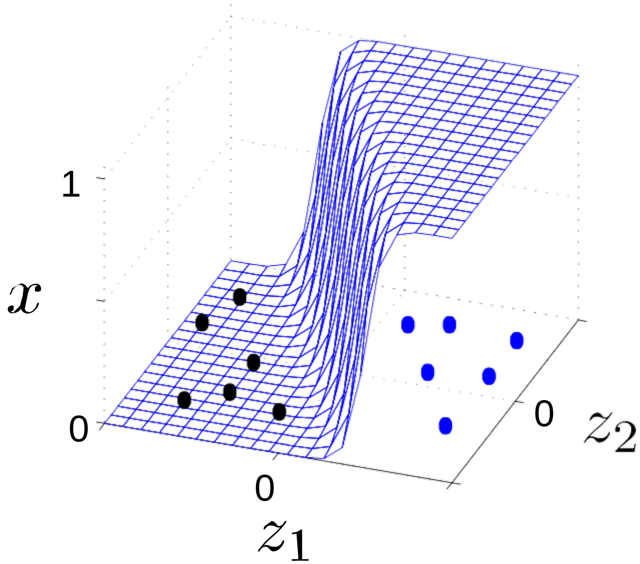
$$G(\boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - t^n) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right)\right] \geq 0$$

surprise $-\log p(\mathrm{outcome})$ when observing $t^{(n)}$

relative entropy between $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})$ and $t^{(n)}$

encourages neuron output to match training data

Training a Single Neuron



training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

objective function:

$$G(\boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - t^{n}) \log\left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right)\right] \geq 0$$

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w})$$

choose the weights that minimise the network's surprise about the training data

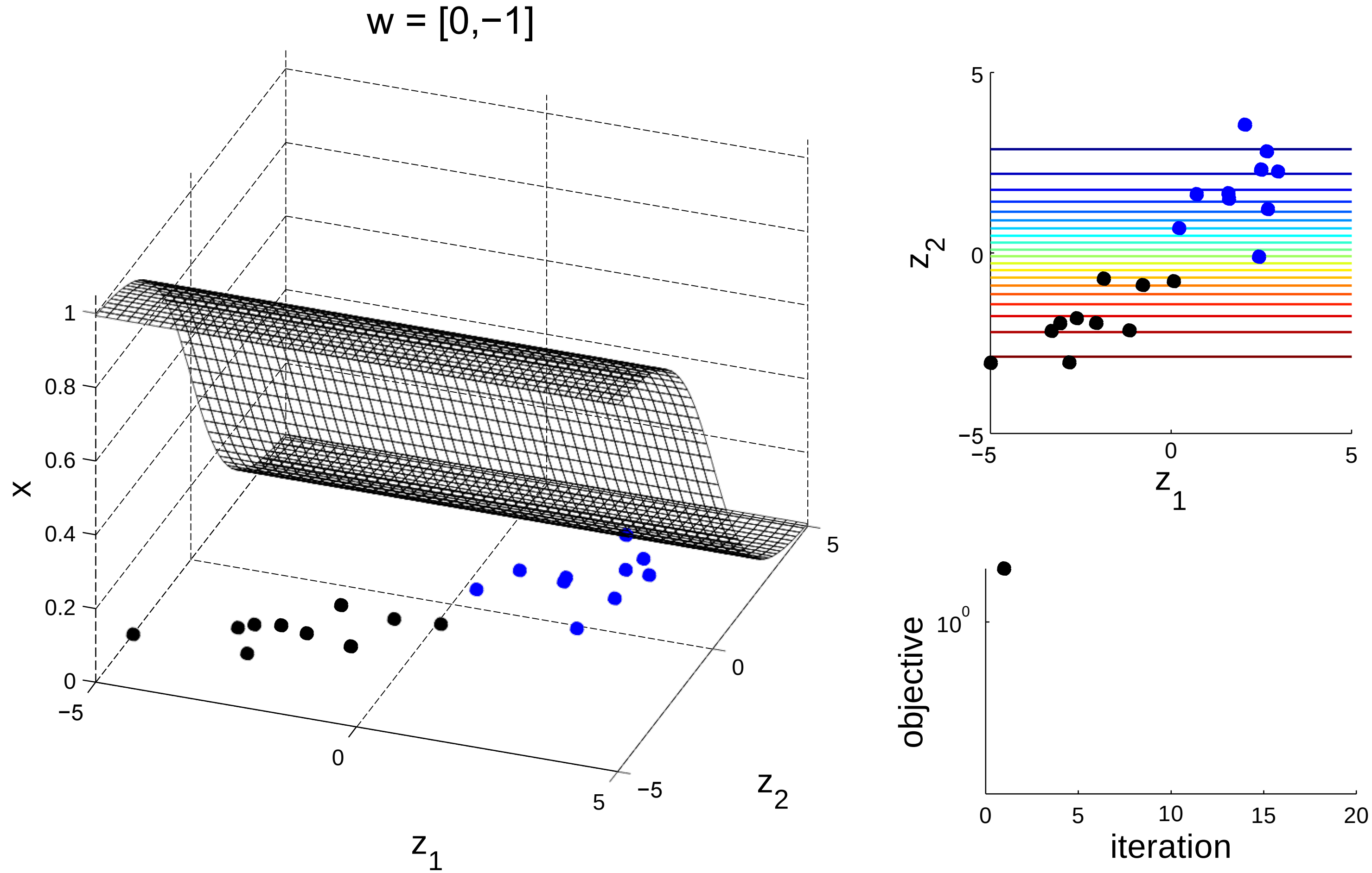
$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} G(\boldsymbol{w}) = \sum_n \frac{\mathrm{d}G(\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}\boldsymbol{w}} = -\sum_n (t^{(n)} - x^{(n)})\boldsymbol{z}^{(n)}$$

= prediction error x feature

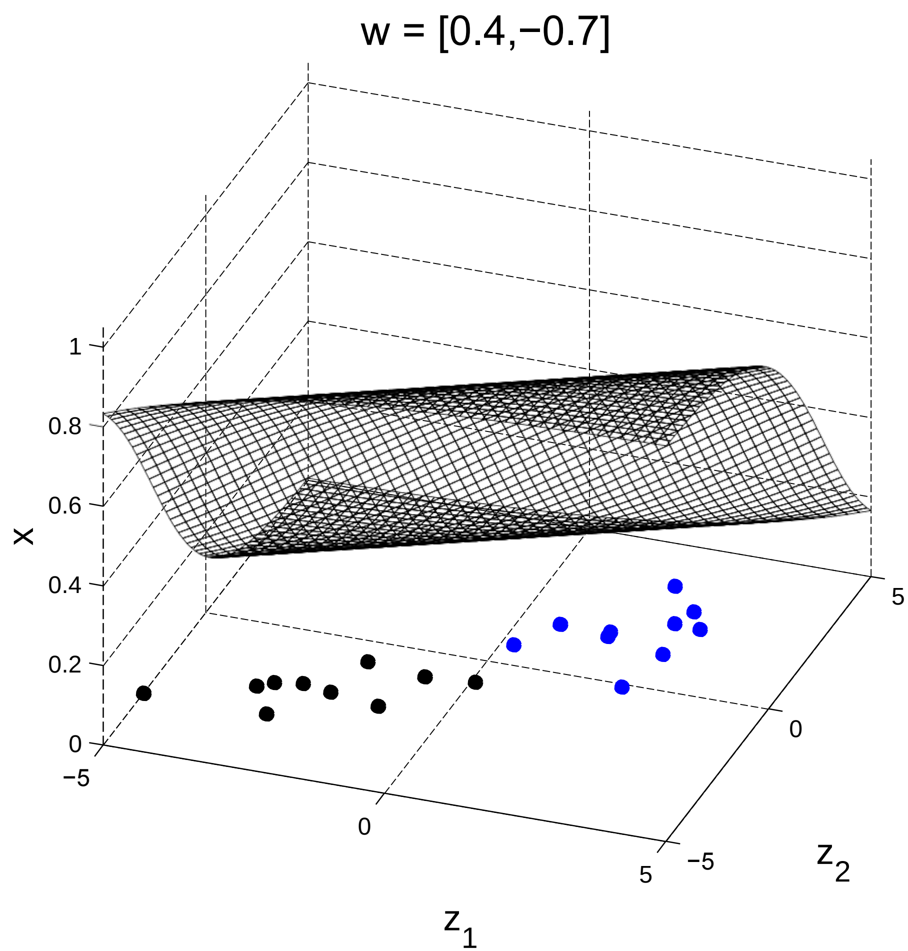
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} G(\boldsymbol{w})$$

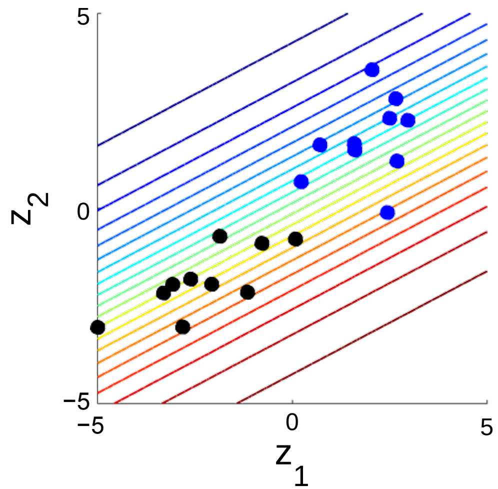
iteratively step down the objective (gradient points up hill)

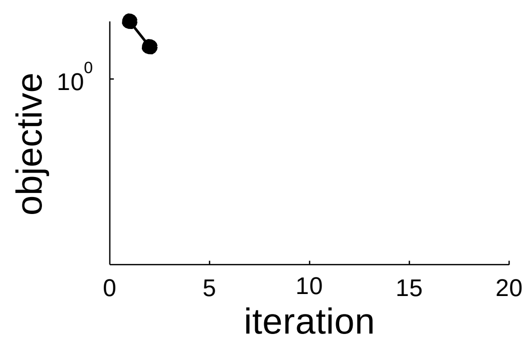
Training a Single Neuron



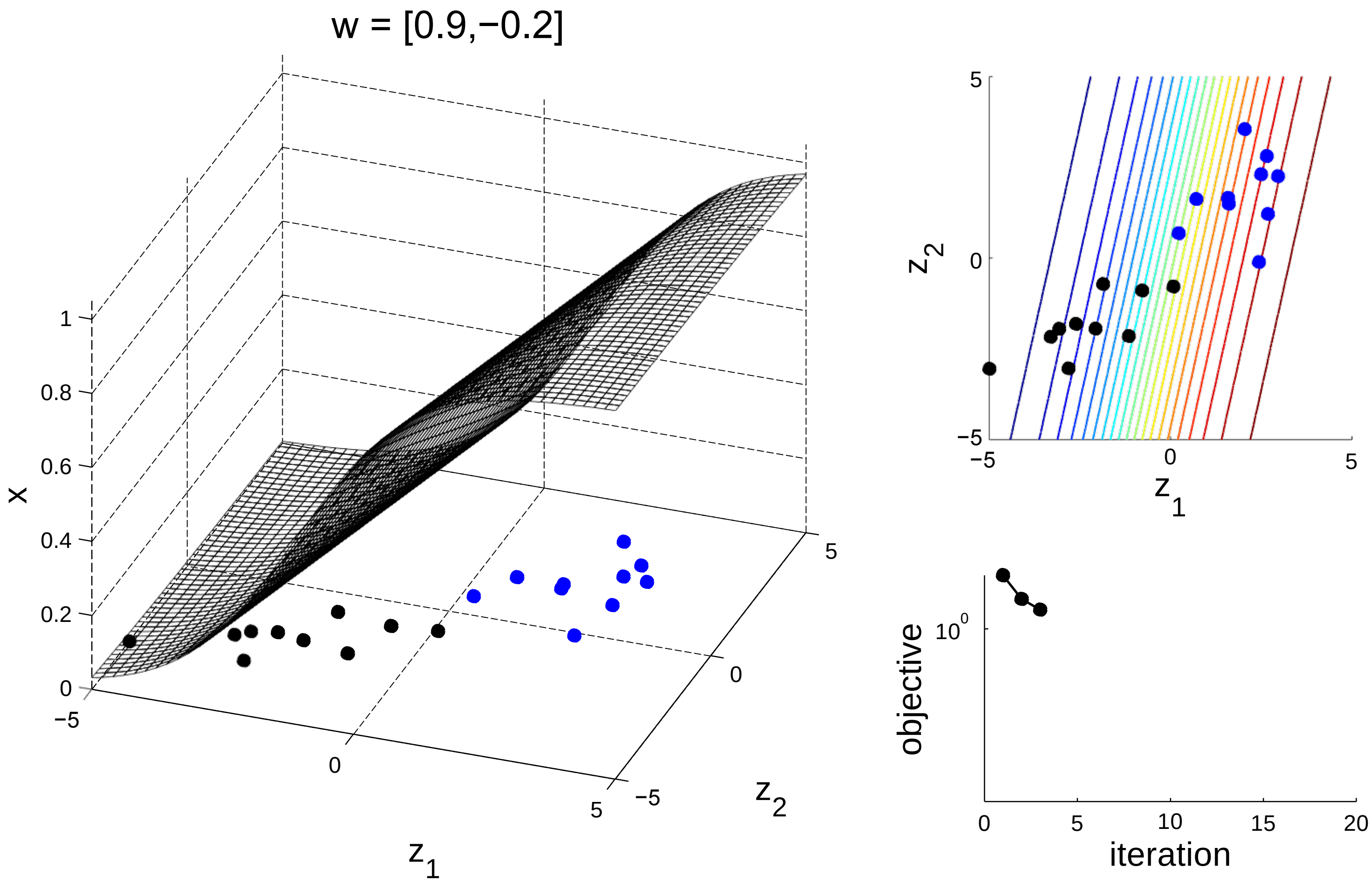
Training a Single Neuron



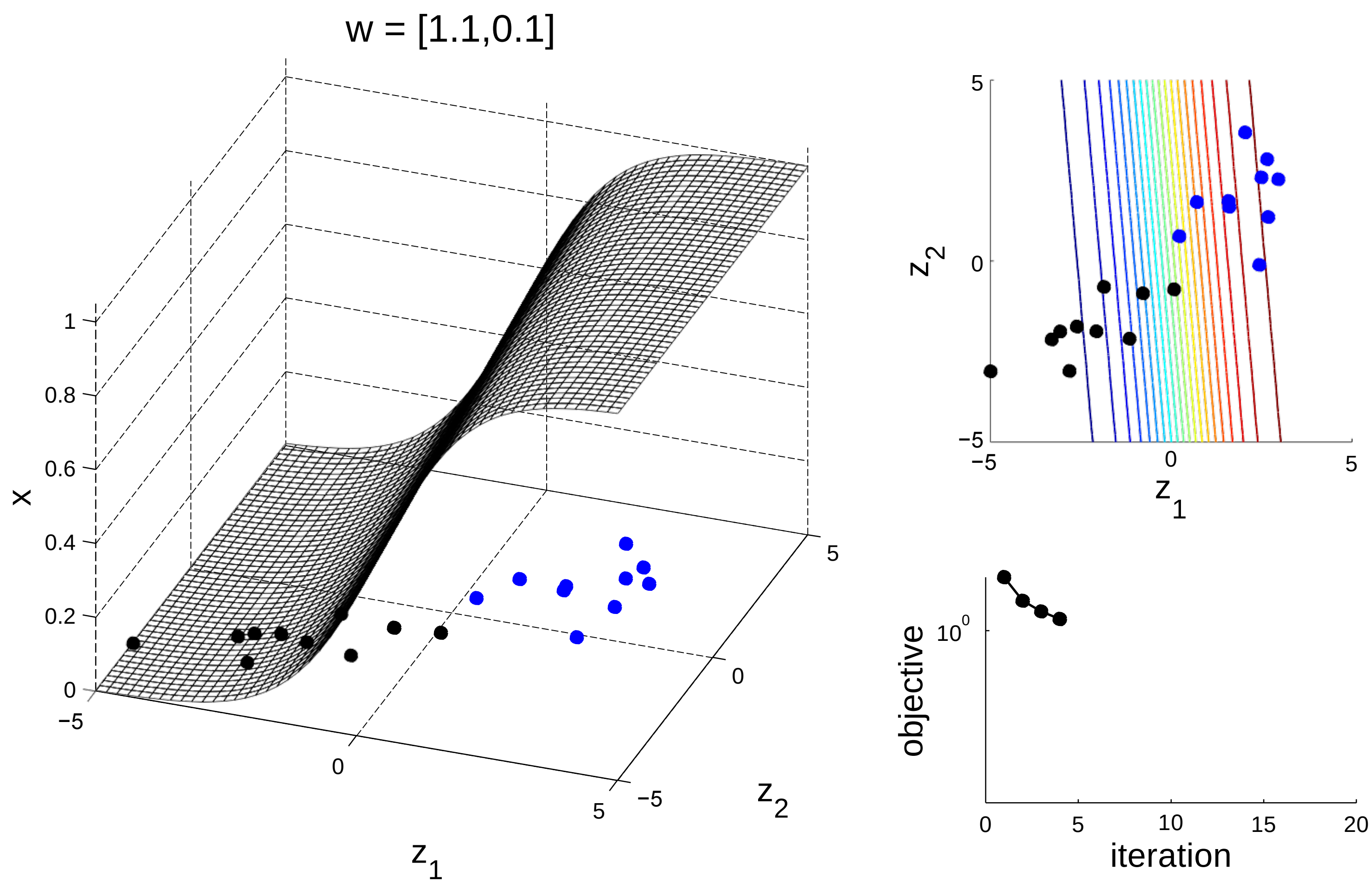




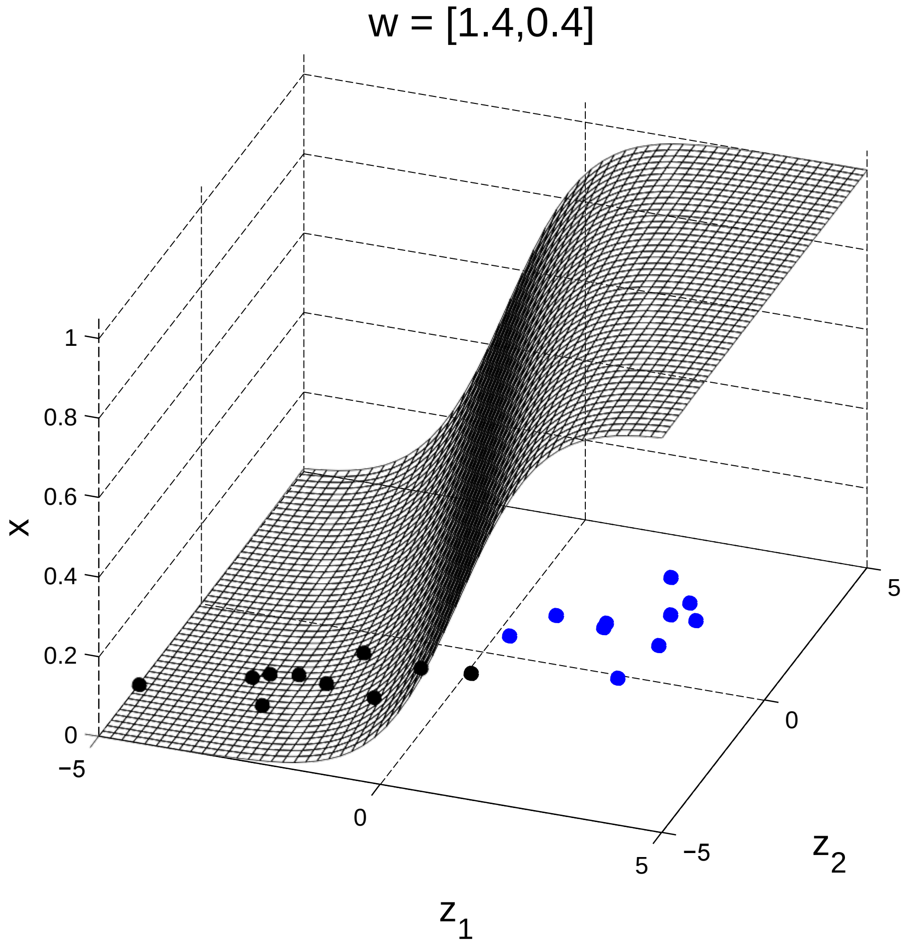
Training a Single Neuron

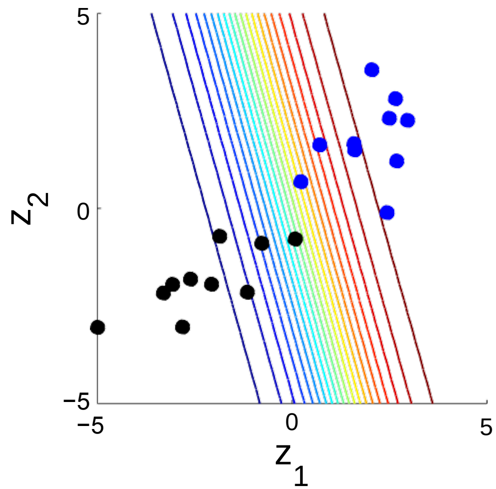


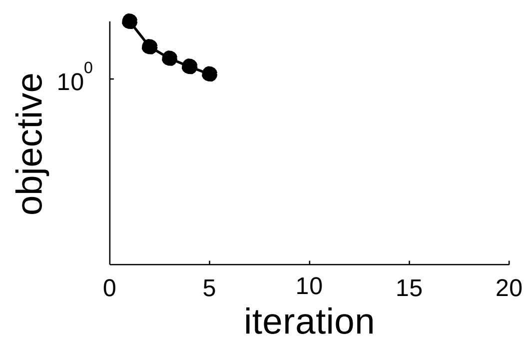
Training a Single Neuron



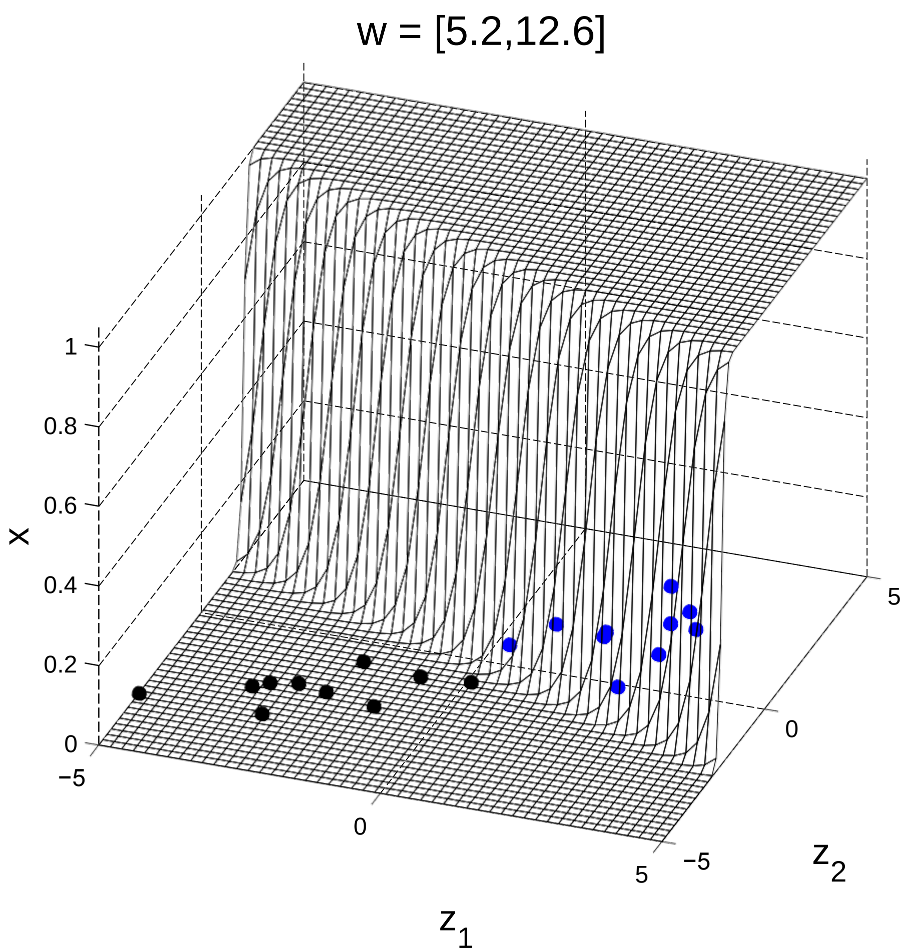
Training a Single Neuron

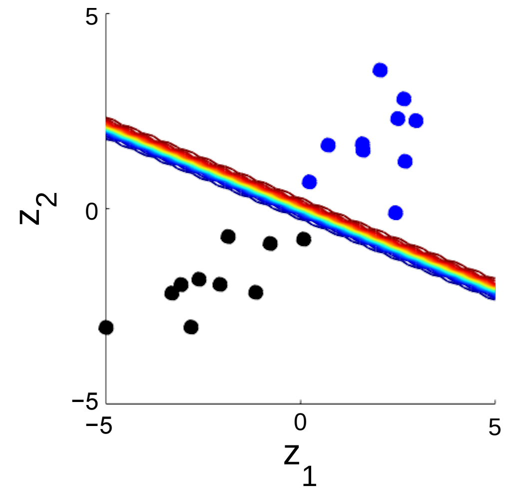


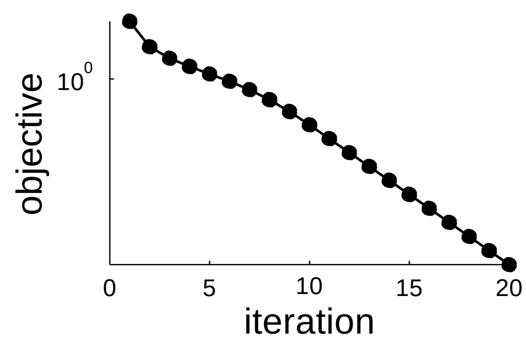




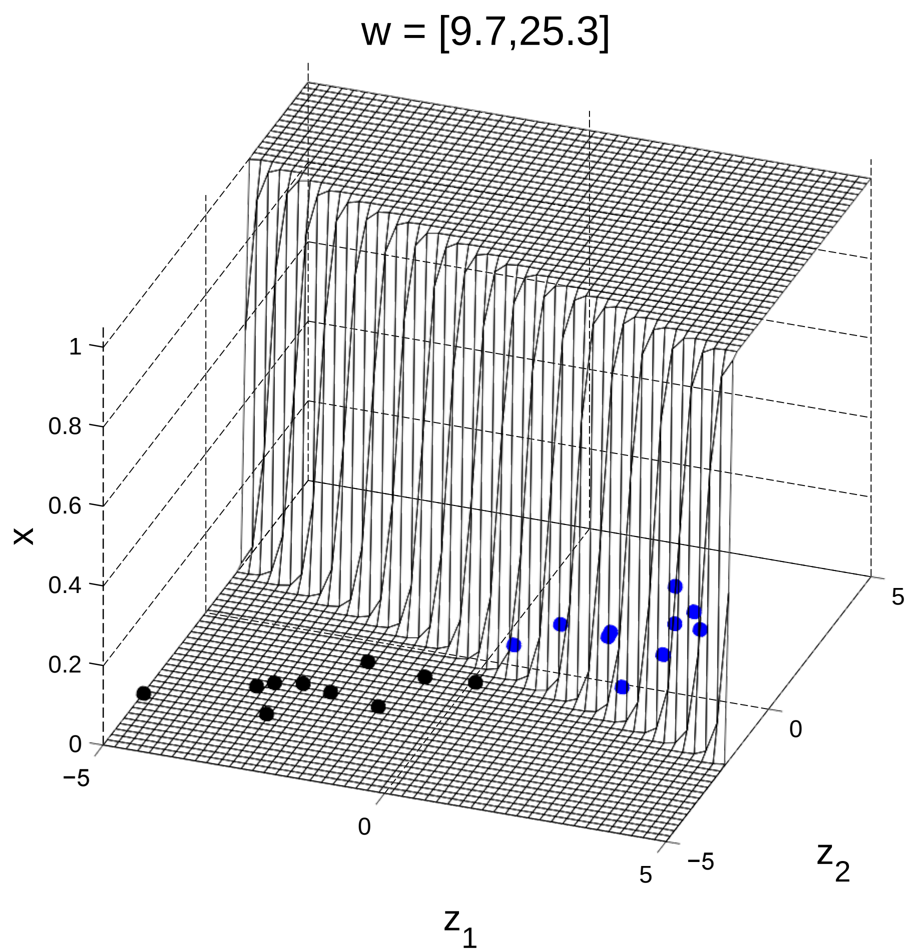
Training a Single Neuron

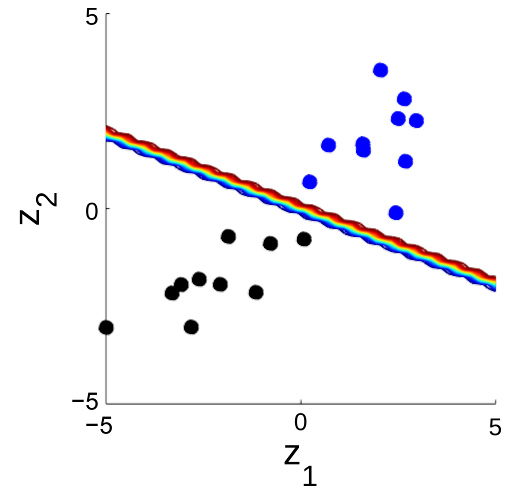


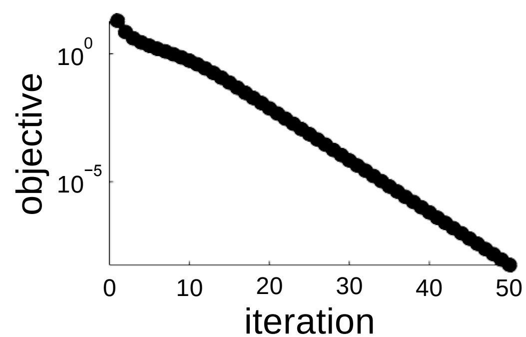




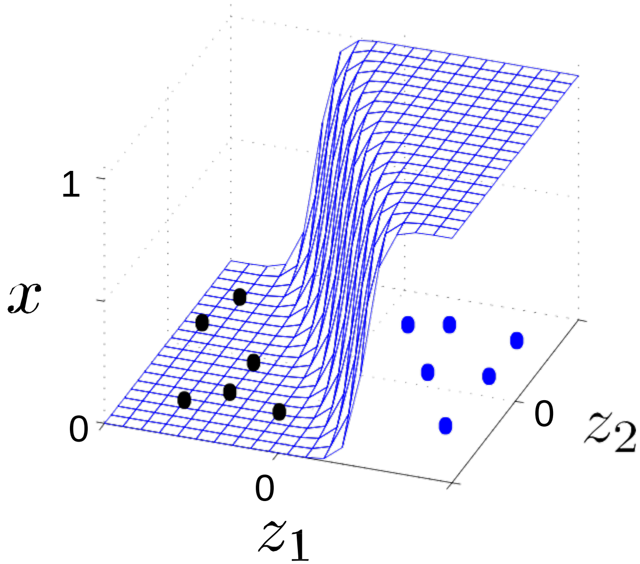
Training a Single Neuron







Overfitting and Weight Decay



training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

objective function:

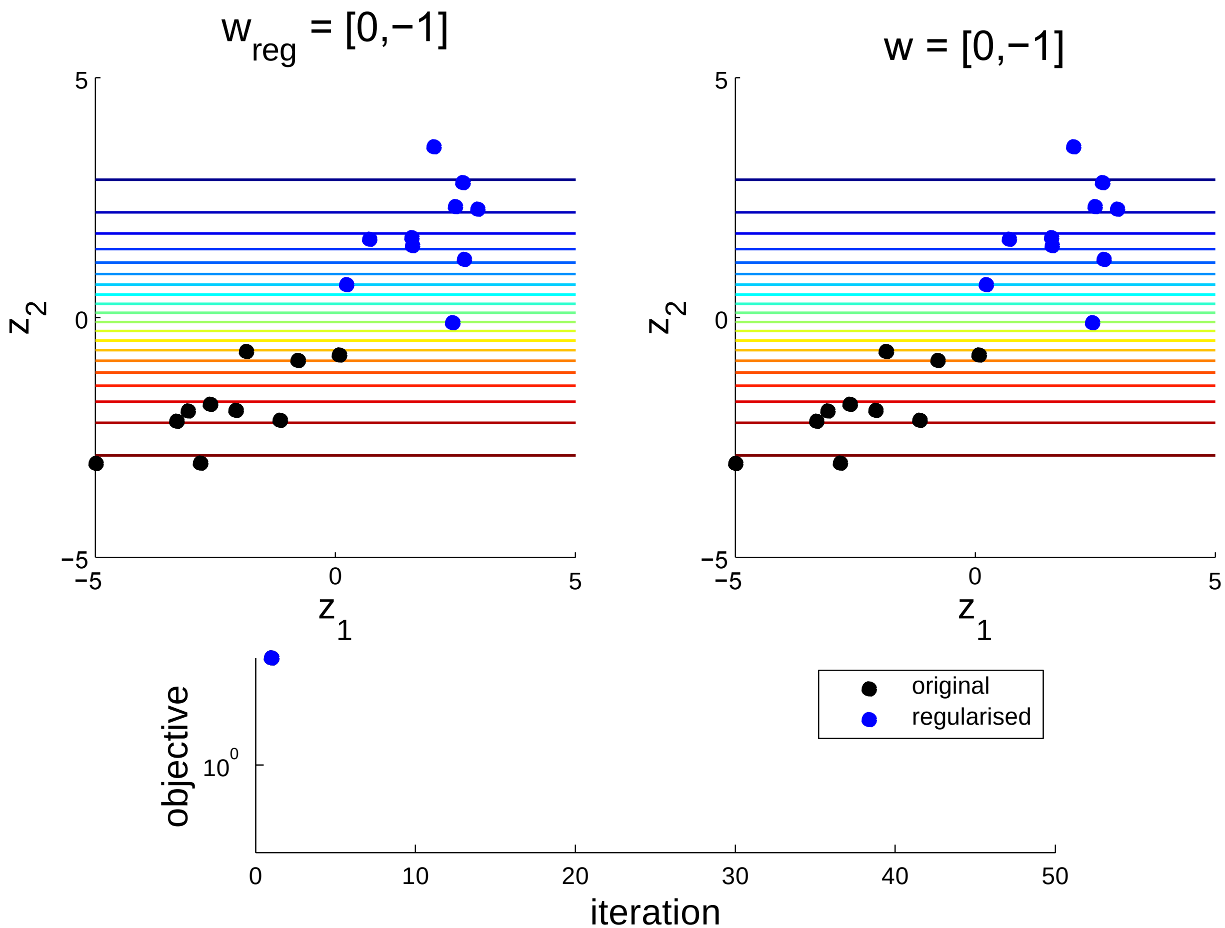
$$G(\boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) + (1-t^n)\log\left(1-\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w})\right)\right]$$

$$E(\boldsymbol{w}) = \tfrac{1}{2}\sum_i w_i^2$$ regulariser discourages the network using extreme weights

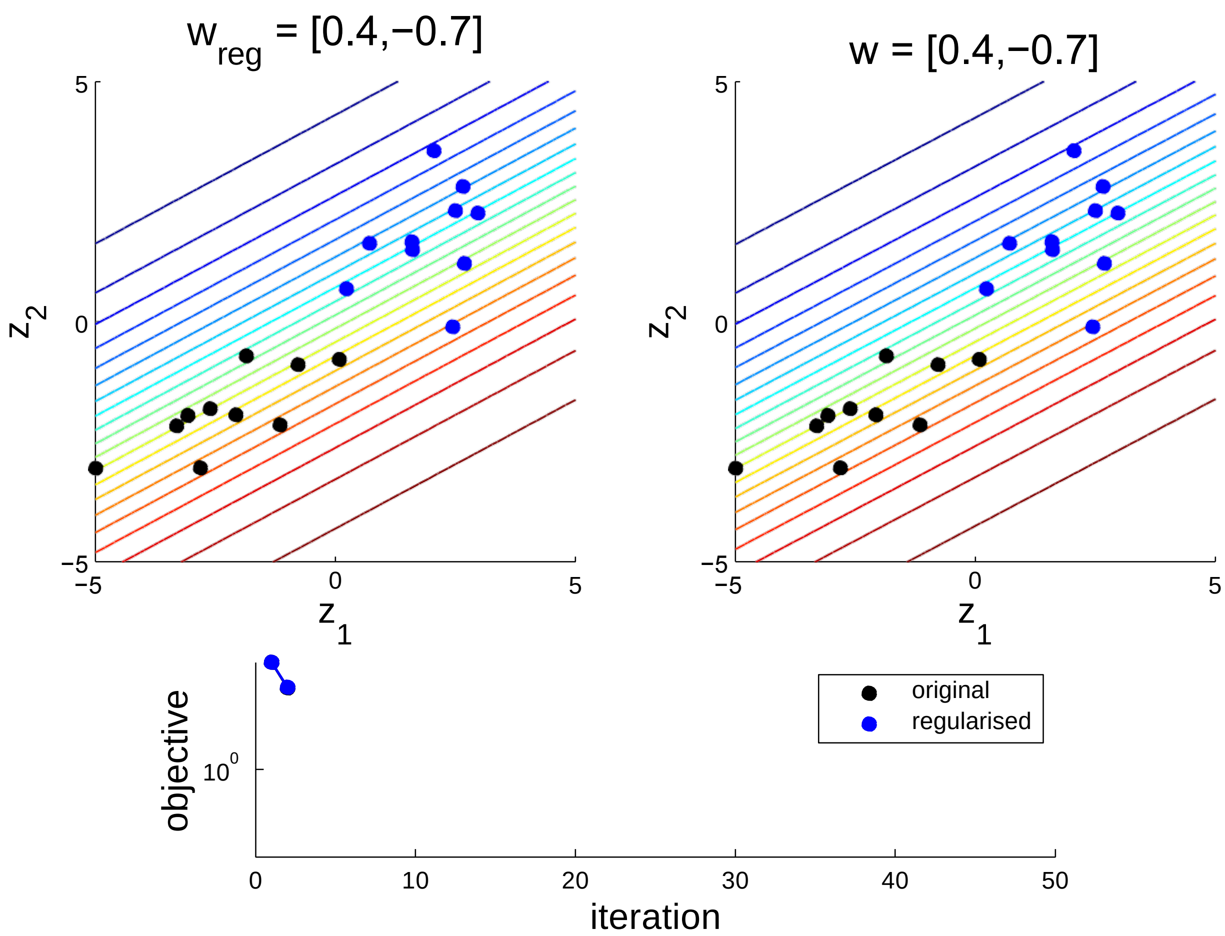
$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} M(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[G(\boldsymbol{w}) + \alpha E(\boldsymbol{w})\right]$$

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} M(\boldsymbol{w}) = -\sum_n (t^{(n)} - x^{(n)})\boldsymbol{z}^{(n)} + \alpha\boldsymbol{w}$$ weight decay - shrinks weights towards zero

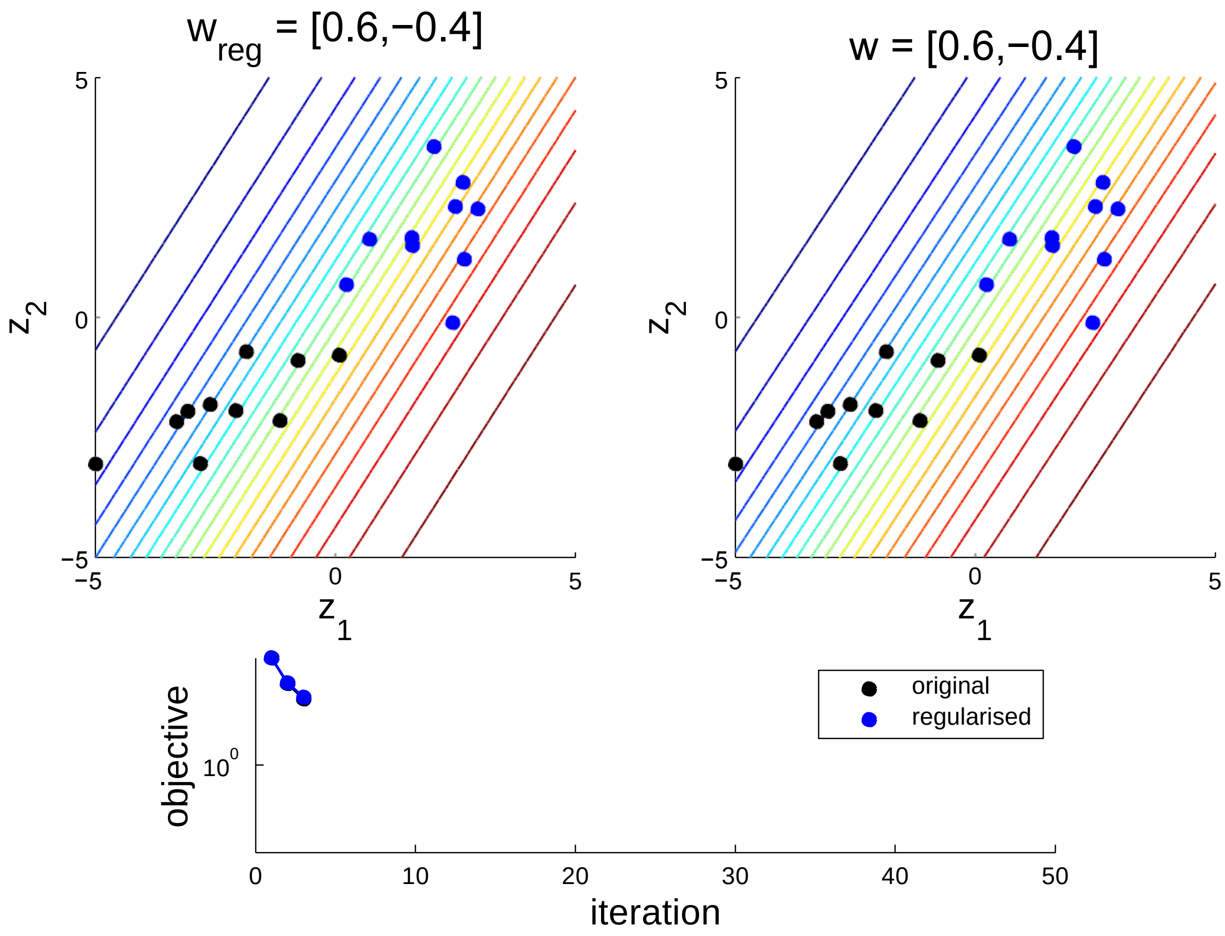
Training a Single Neuron (cont'd)



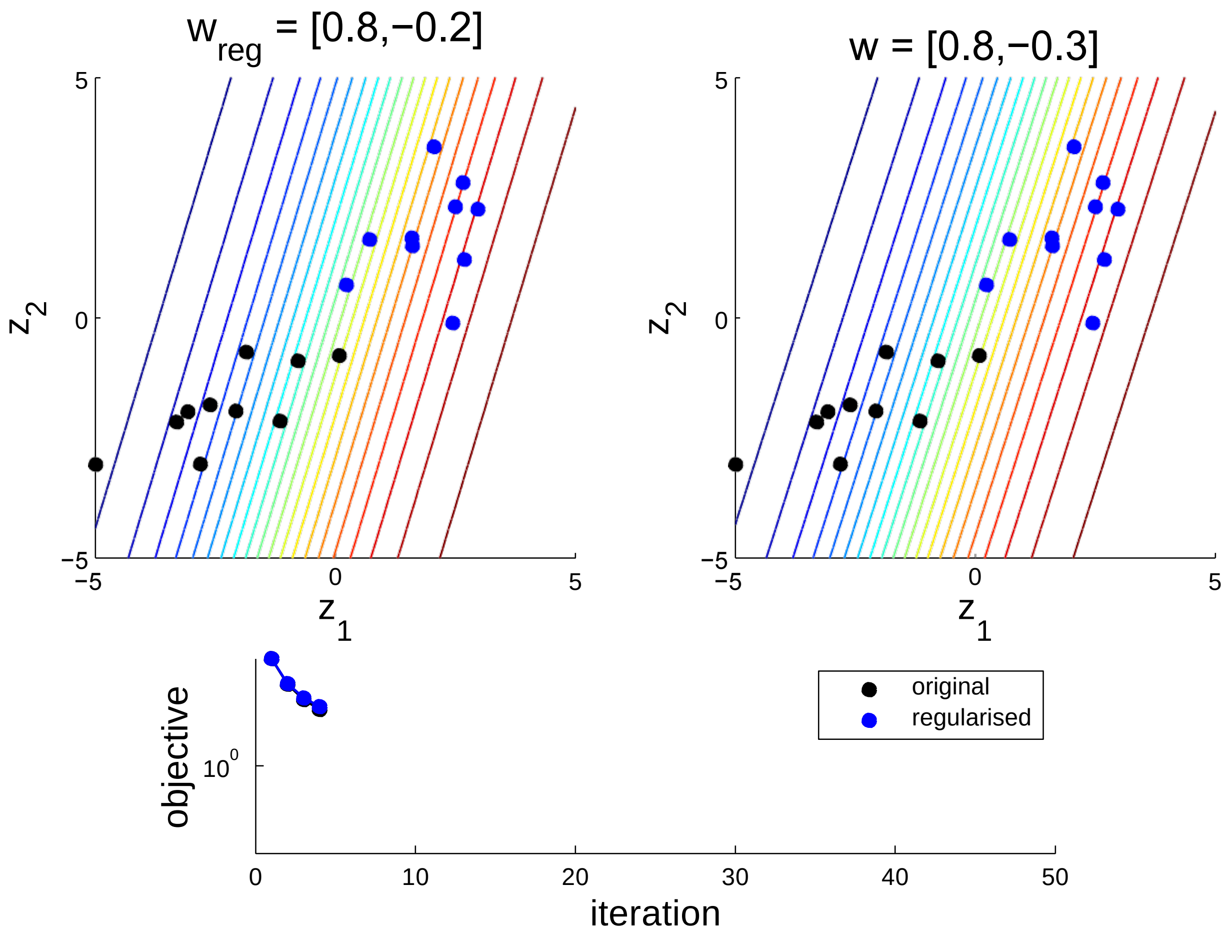
Training a Single Neuron (cont'd)



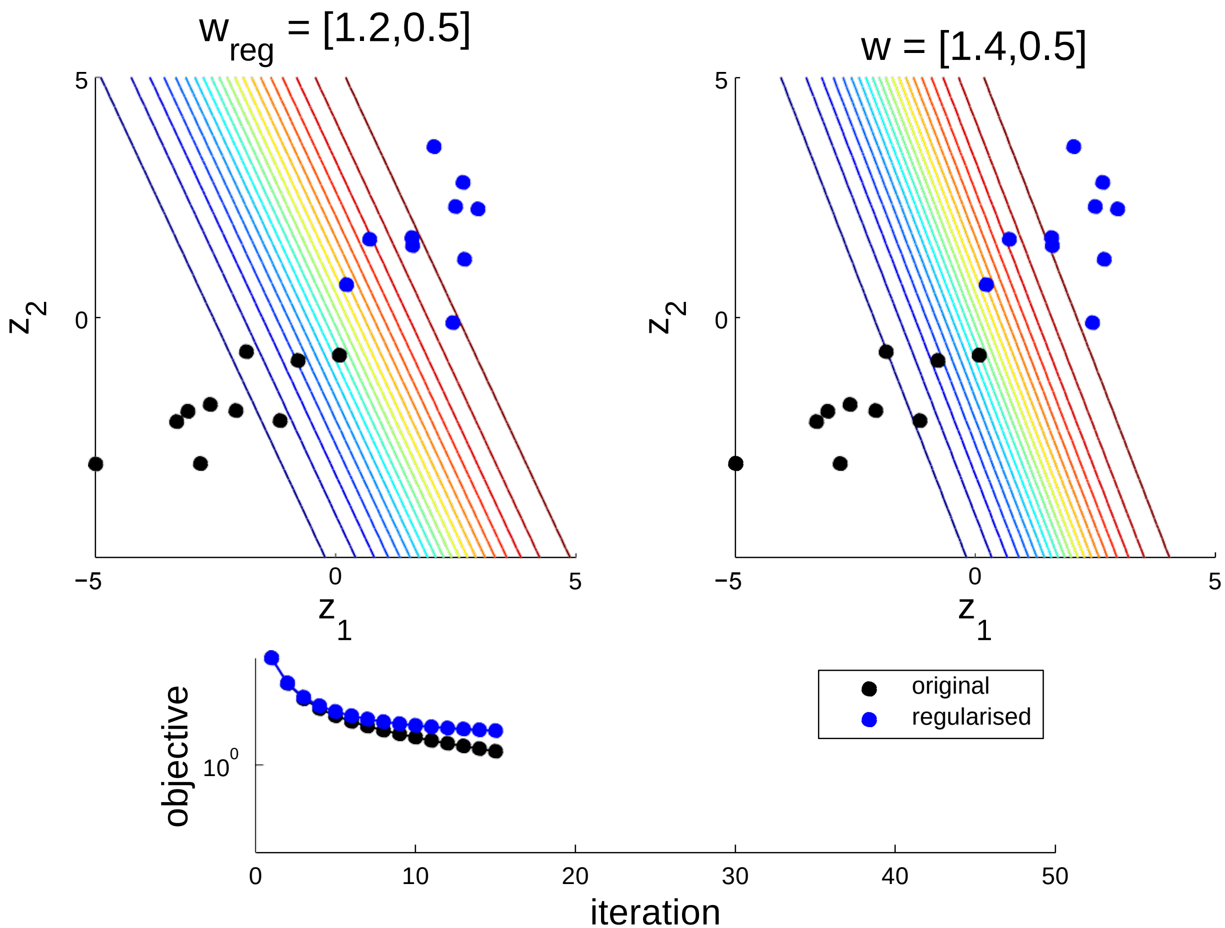
Training a Single Neuron (cont'd)



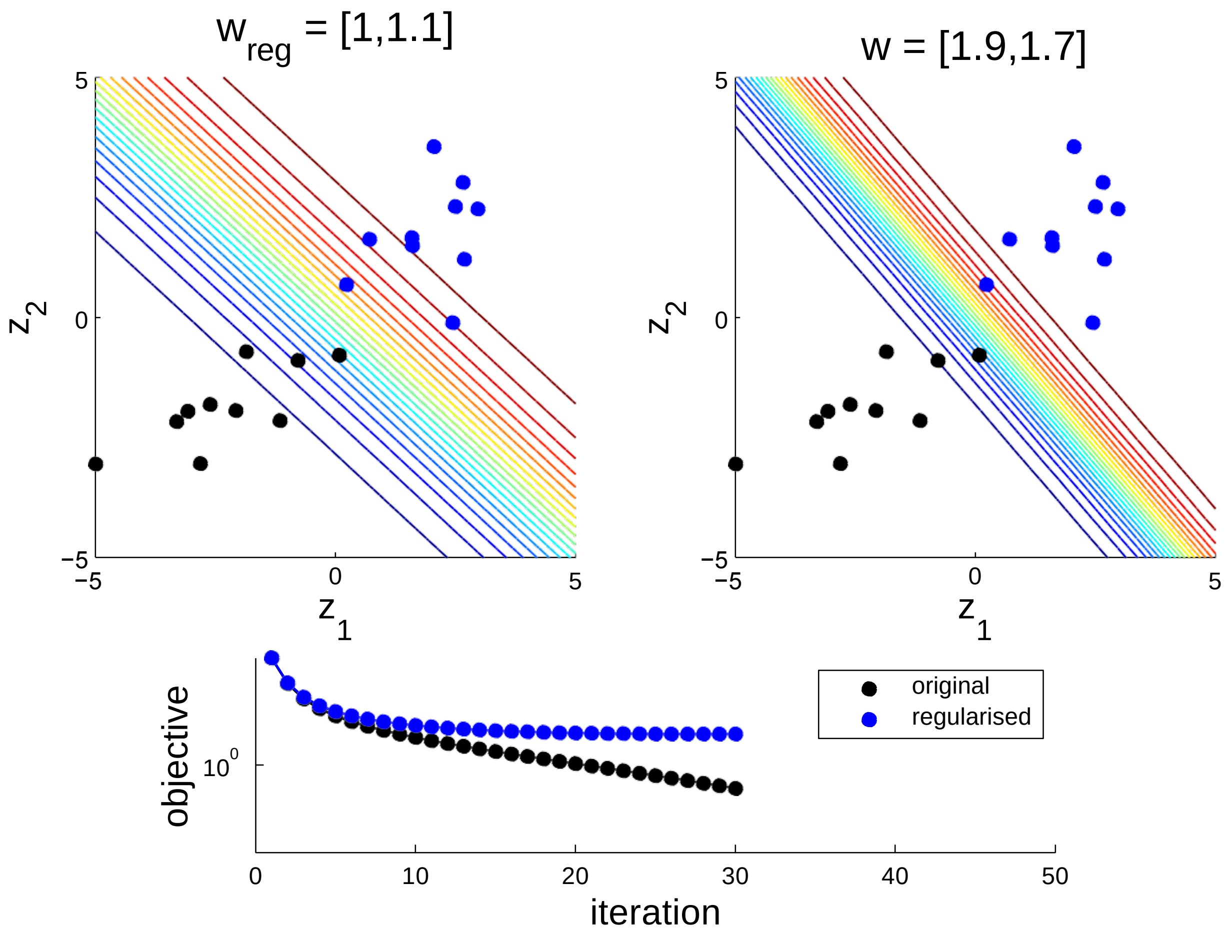
Training a Single Neuron (cont'd)



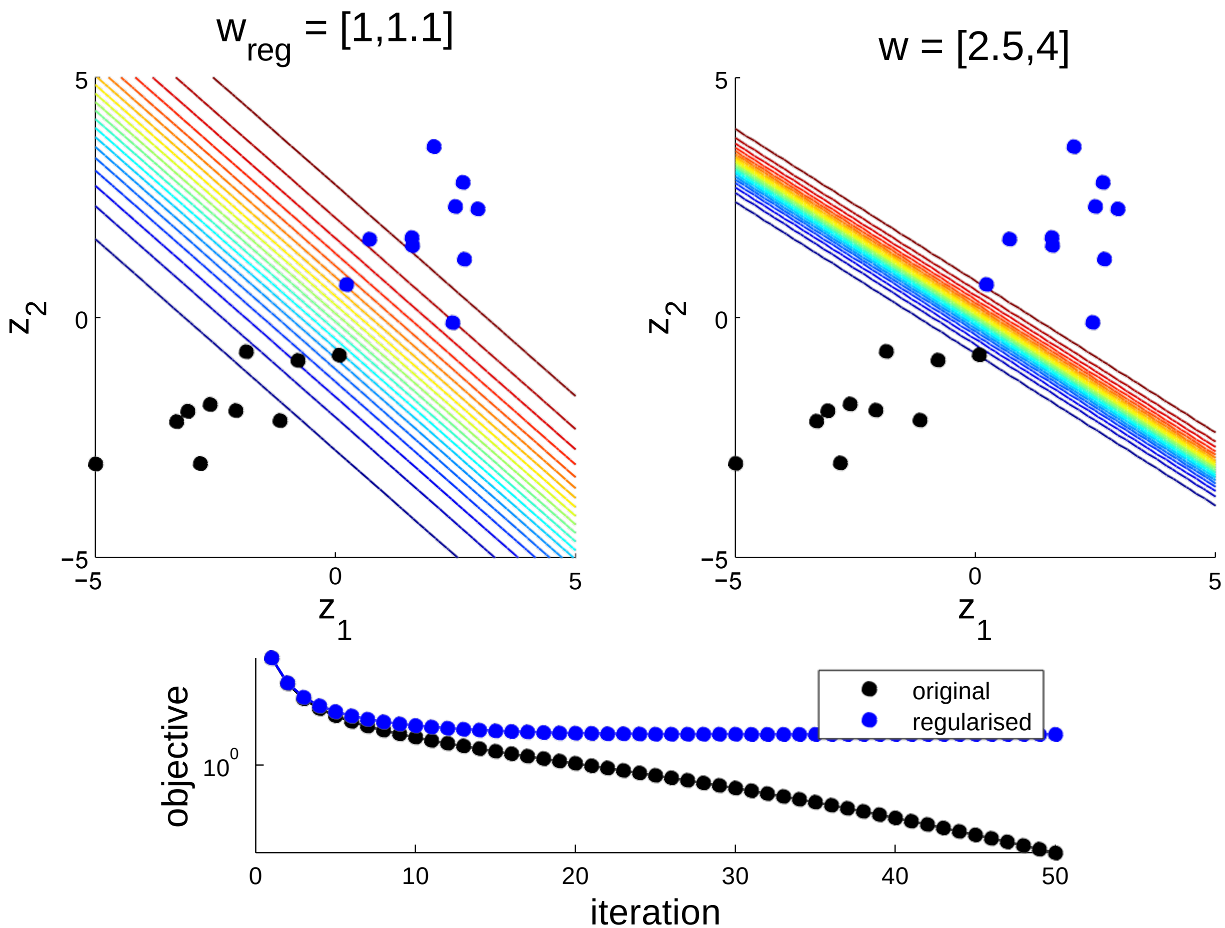
Training a Single Neuron (cont'd)



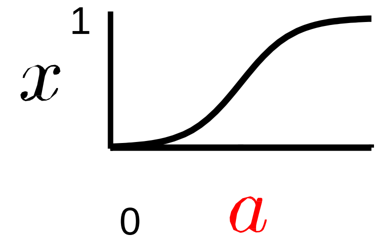
Training a Single Neuron (cont'd)

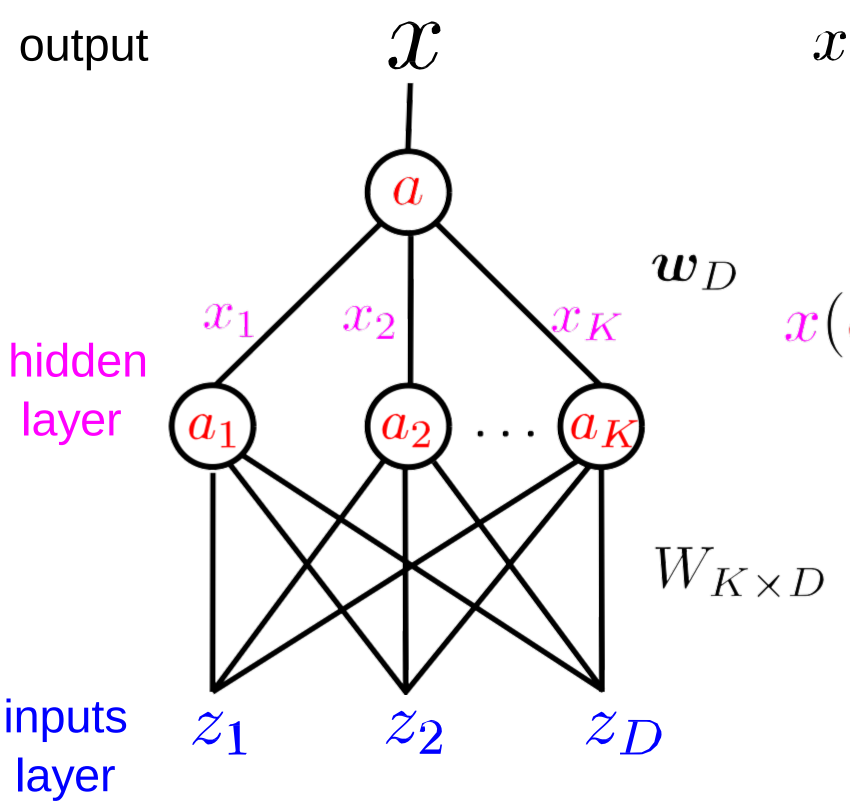


Training a Single Neuron (cont'd)



Single Hidden Layer Neural Networks





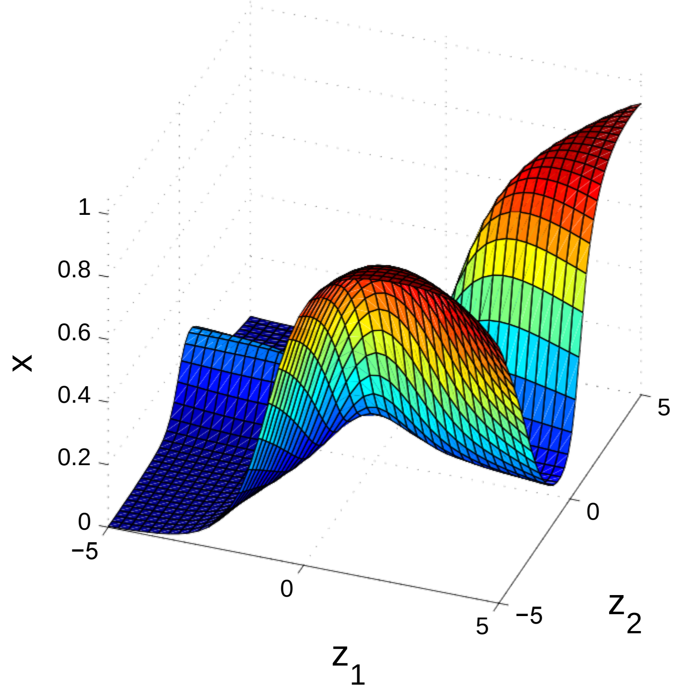
$$x(a) = \frac{1}{1+\exp(-a)}$$

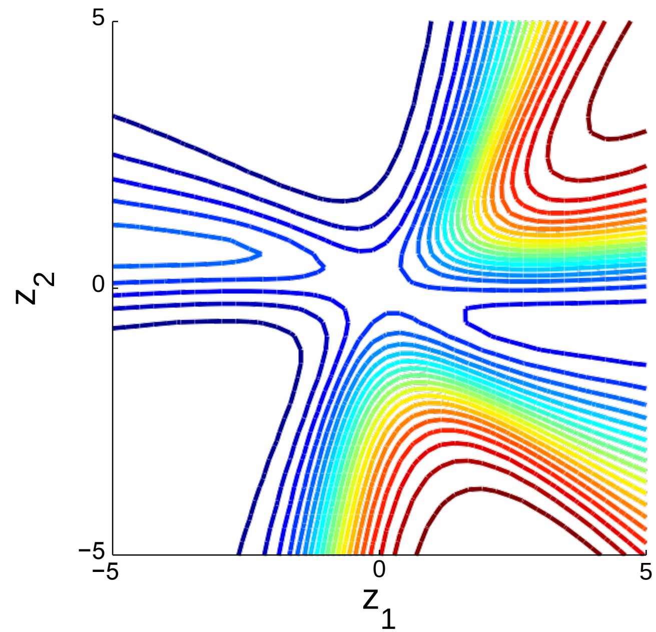
$$a = \sum_{k=1}^{K} w_k x_k$$

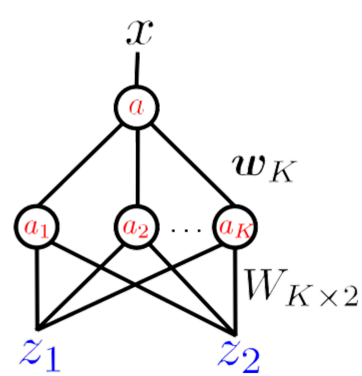
$$x(a_k) = \frac{1}{1+\exp(-a_k)}$$

$$a_k = \sum_{d=1}^{D} W_{k,d} z_d$$

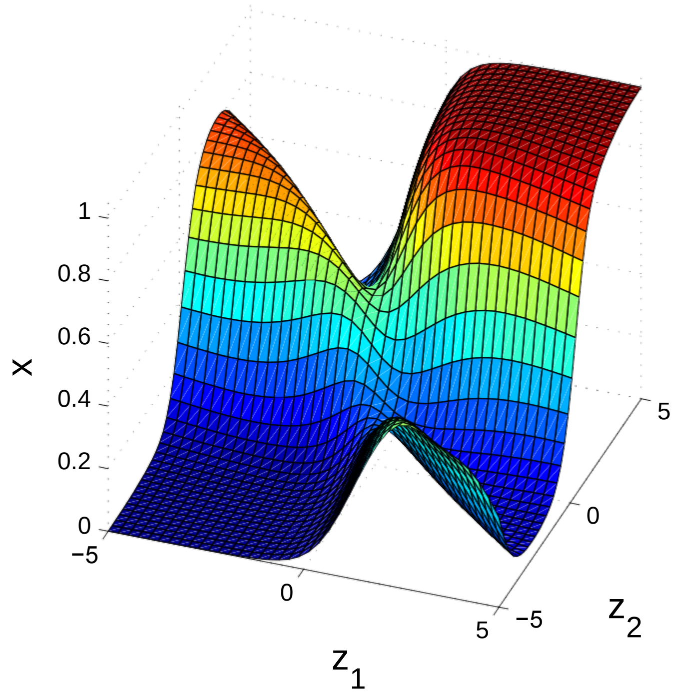
Sampling Random Neural Network Classifiers

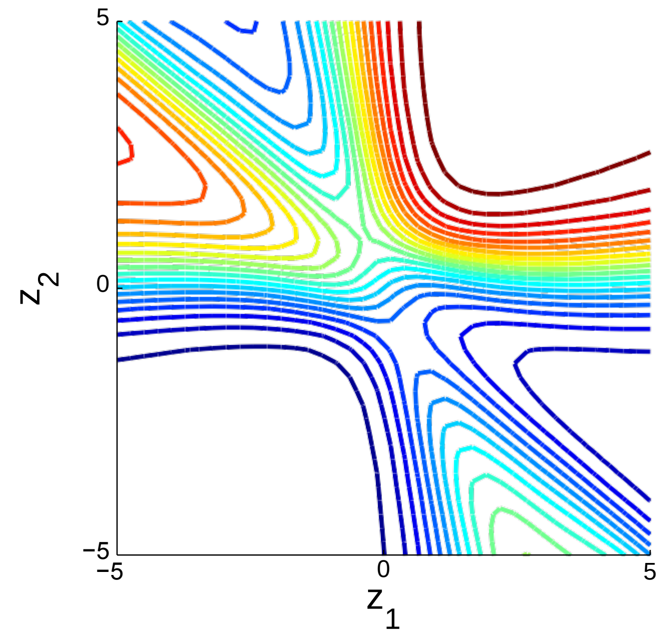


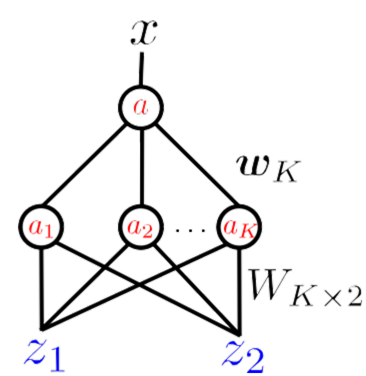




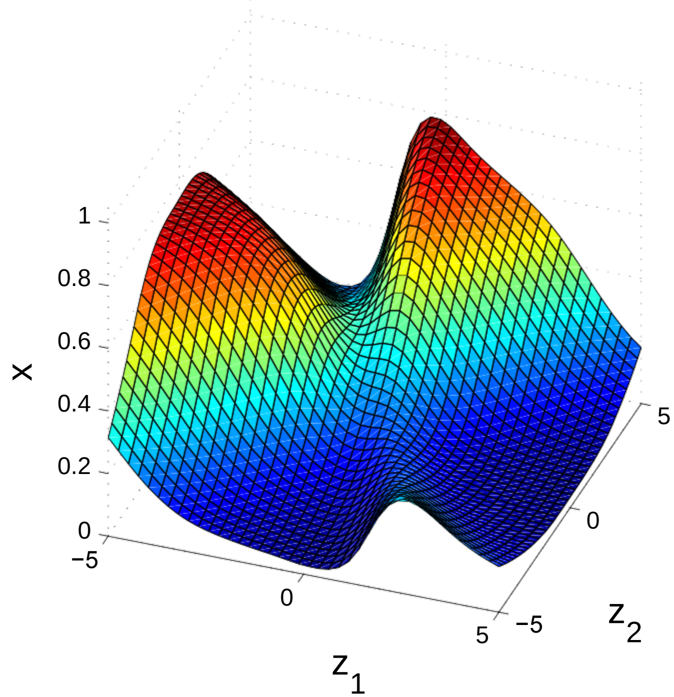
Sampling Random Neural Network Classifiers

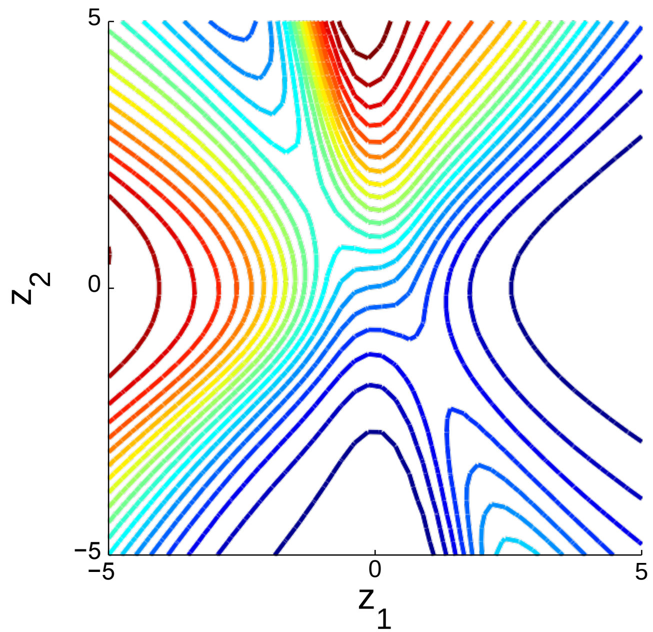


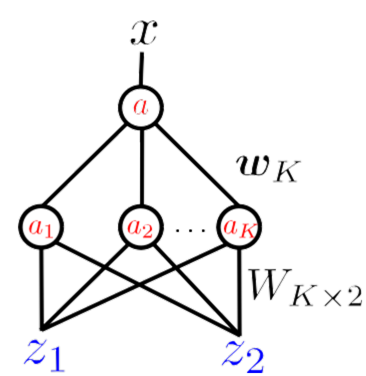




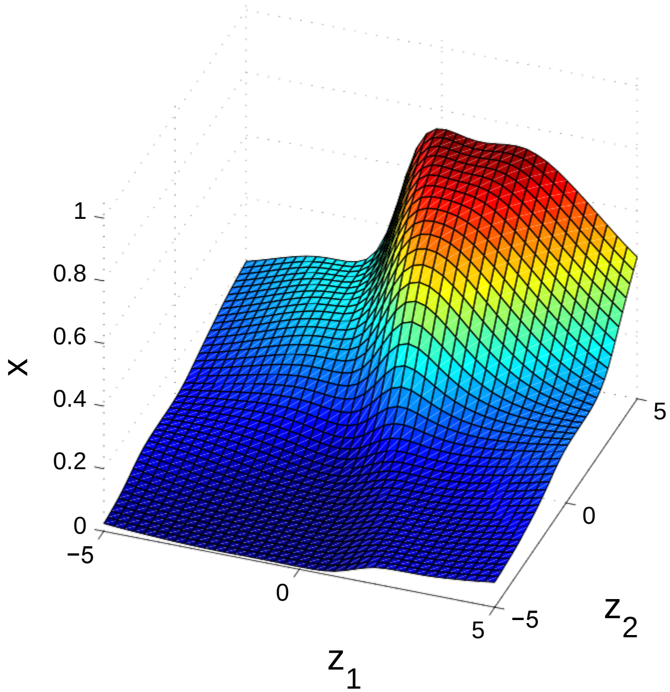
Sampling Random Neural Network Classifiers

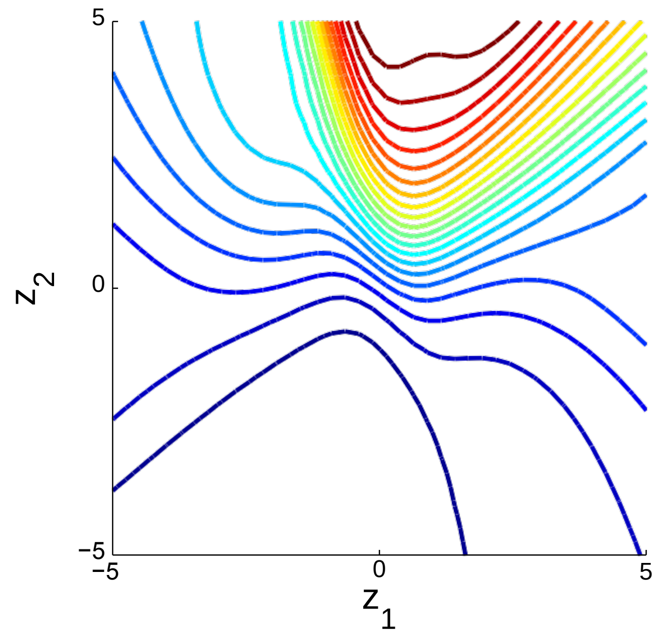


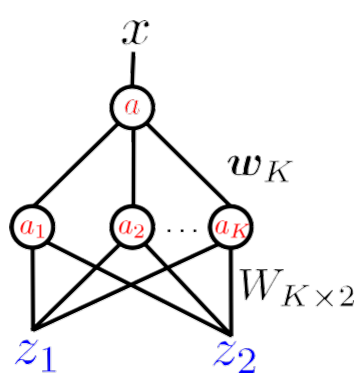




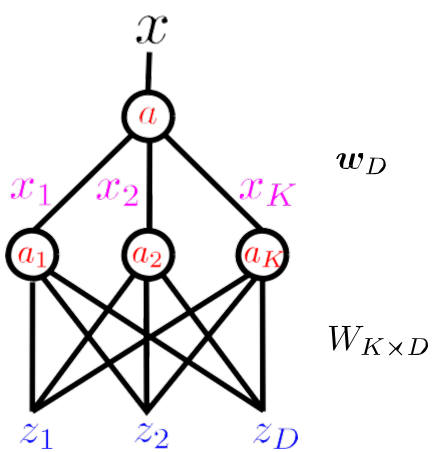
Sampling Random Neural Network Classifiers







Training a Neural Network with a Single Hidden Layer



$$x(a) = \frac{1}{1+\exp(-a)}$$

$$a = \sum_{k=1}^{K} w_k x_k$$

$$x(a_k) = \frac{1}{1+\exp(-a_k)}$$

$$a_k = \sum_{d=1}^{D} W_{k,d} z_d$$

objective function:

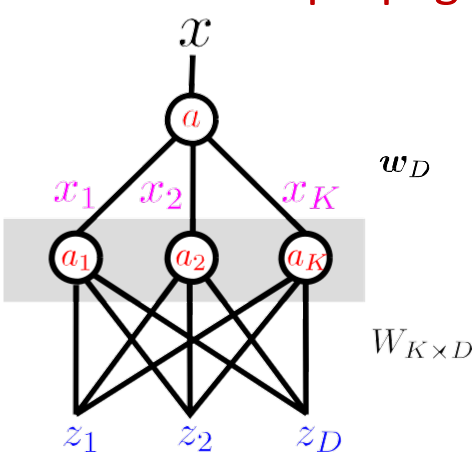
$$G(W, \boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}^{(n)} + (1 - t^n) \log\left(1 - \mathrm{x}^{(n)}\right)\right]$$ likelihood same as before

$$E(W, \boldsymbol{w}) = \tfrac{1}{2} \sum_i w_i^2 + \tfrac{1}{2} \sum_{ij} W_{ij}^2$$ regulariser discourages extreme weights

$$\{W, \boldsymbol{w}^*\} = \underset{W, \boldsymbol{w}}{\arg\min}\, M(W, \boldsymbol{w}) = \underset{W, \boldsymbol{w}}{\arg\min} \left[G(W, \boldsymbol{w}) + \alpha E(W, \boldsymbol{w})\right]$$

Training a Neural Network with a Single Hidden Layer

Networks with hidden layers can be fit using gradient descent using an algorithm called back-propagation.



$$x(a) = \frac{1}{1+\exp(-a)}$$

$$a = \sum_{k=1}^{K} w_k x_k$$

$$x(a_k) = \frac{1}{1+\exp(-a_k)}$$

$$a_k = \sum_{d=1}^{D} W_{k,d} z_d$$

objective function:

$$G(W, \boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}^{(n)} + (1 - t^n) \log \left(1 - \mathrm{x}^{(n)}\right)\right]$$ likelihood same as before

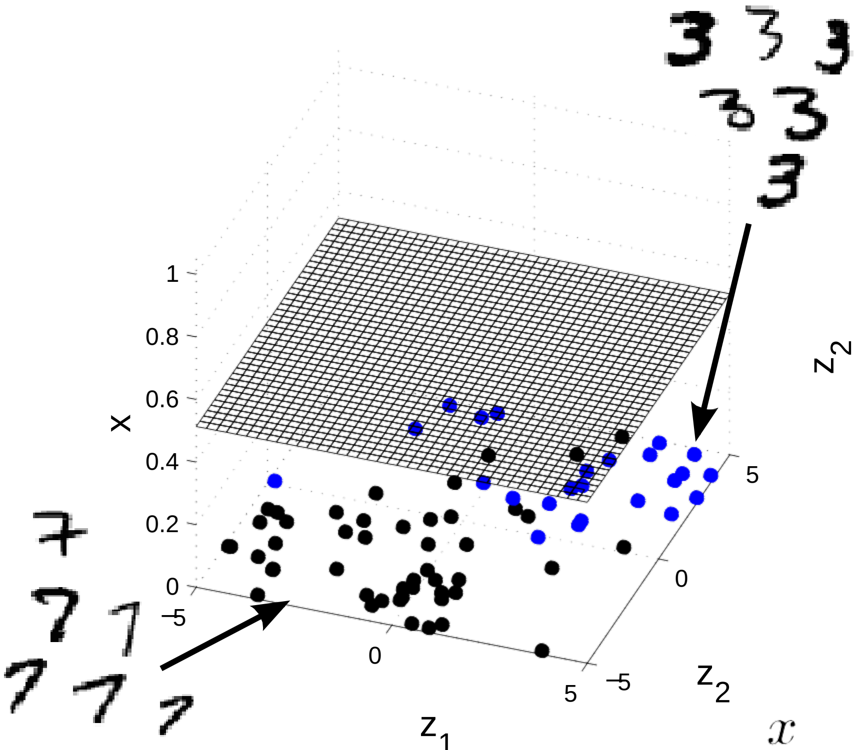
$$E(W, \boldsymbol{w}) = \frac{1}{2}\sum_i w_i^2 + \frac{1}{2}\sum_{ij} W_{ij}^2$$ regulariser discourages extreme weights

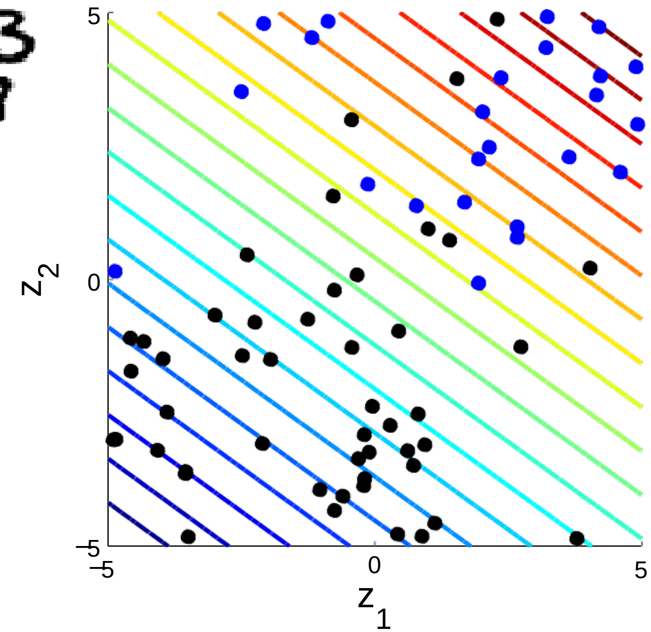
$$\{W, \boldsymbol{w}^*\} = \underset{W,\boldsymbol{w}}{\arg\min}\, M(W, \boldsymbol{w}) = \underset{W,\boldsymbol{w}}{\arg\min}\left[G(W, \boldsymbol{w}) + \alpha E(W, \boldsymbol{w})\right]$$

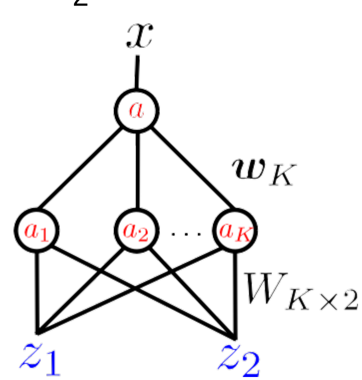
$$\frac{\mathrm{d}G(W, \boldsymbol{w})}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W, \boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W, \boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}a^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}W_{ij}}$$

$$= \sum_n \frac{\mathrm{d}G(W, \boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}a^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}x_i^{(n)}} \frac{\mathrm{d}x_i^{(n)}}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W, \boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}a^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}x_i^{(n)}} \frac{\mathrm{d}x_i^{(n)}}{\mathrm{d}a_i^{(n)}} \frac{\mathrm{d}a_i^{(n)}}{\mathrm{d}W_{ij}}$$

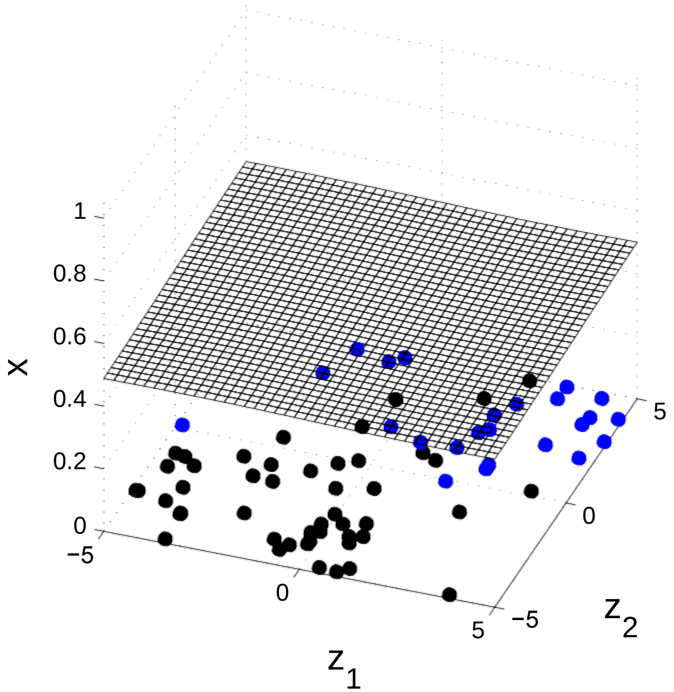
Training a Neural Network with a Single Hidden Layer

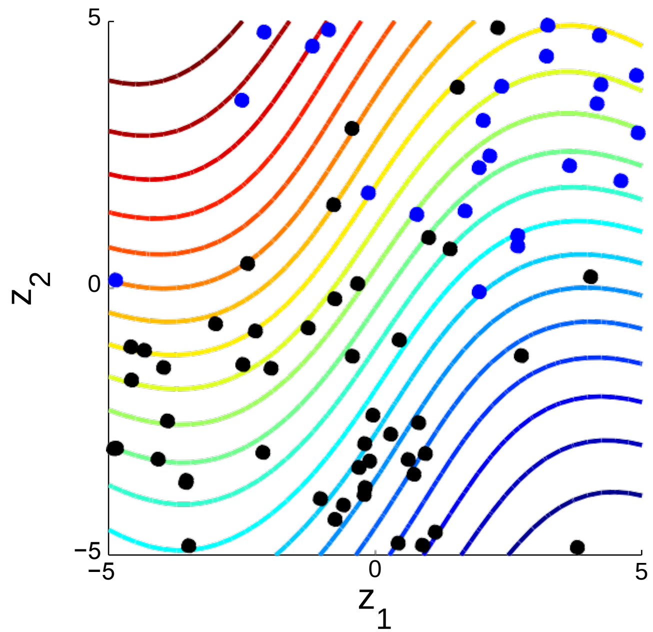


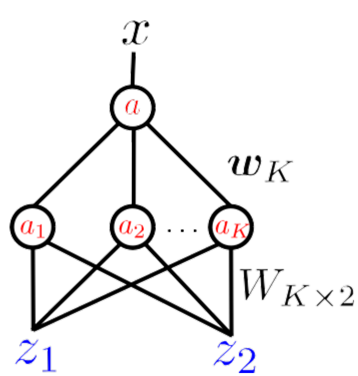




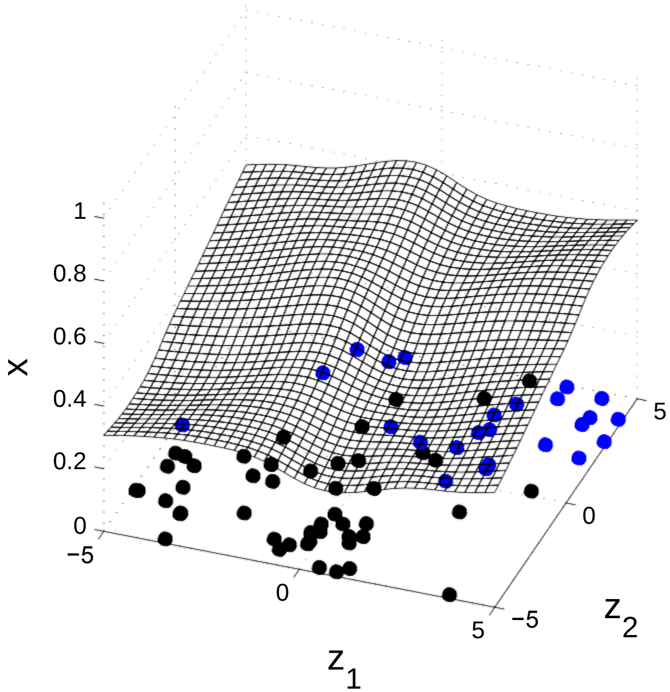
Training a Neural Network with a Single Hidden Layer

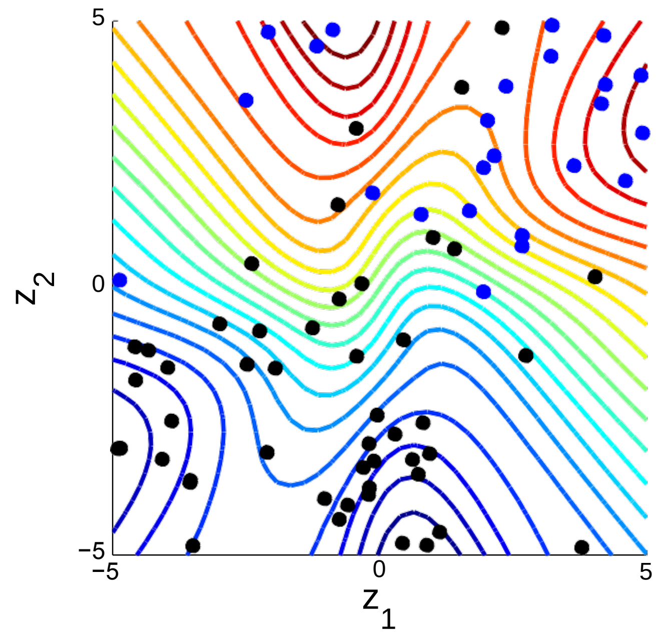


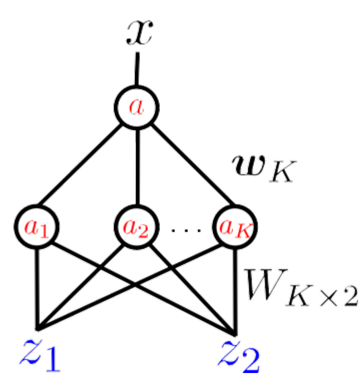




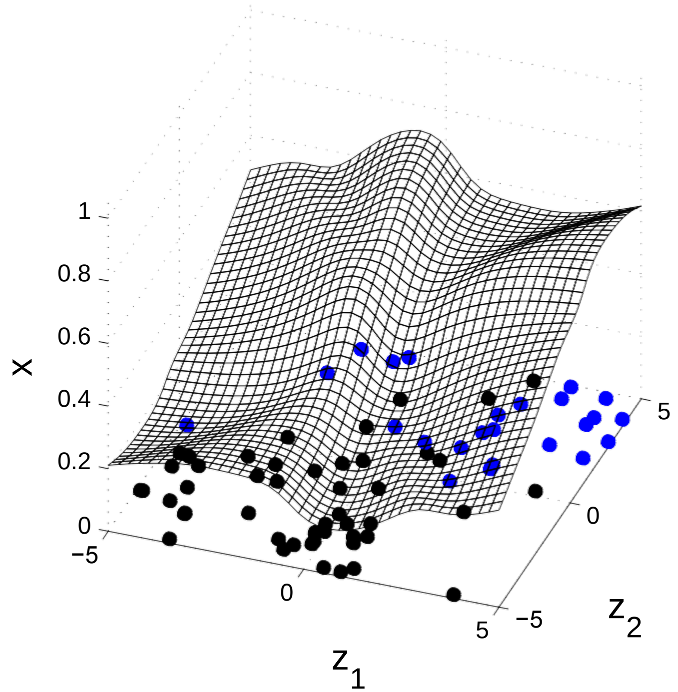
Training a Neural Network with a Single Hidden Layer

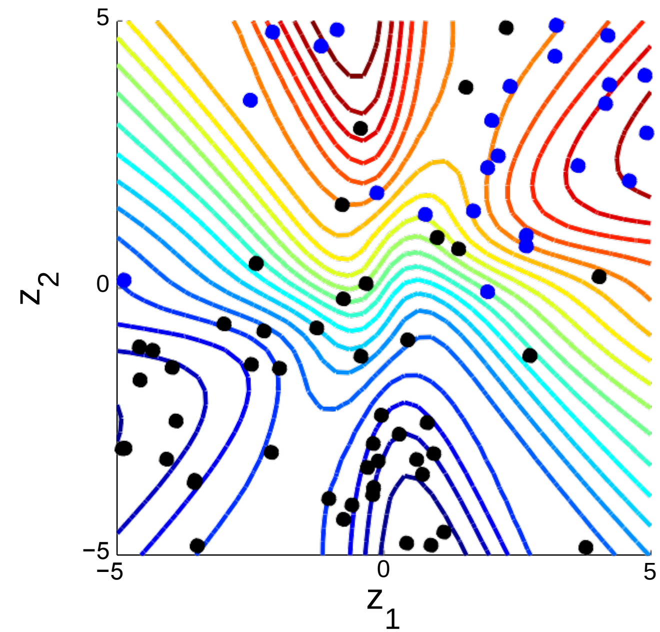


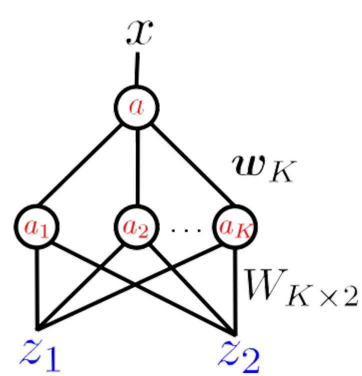




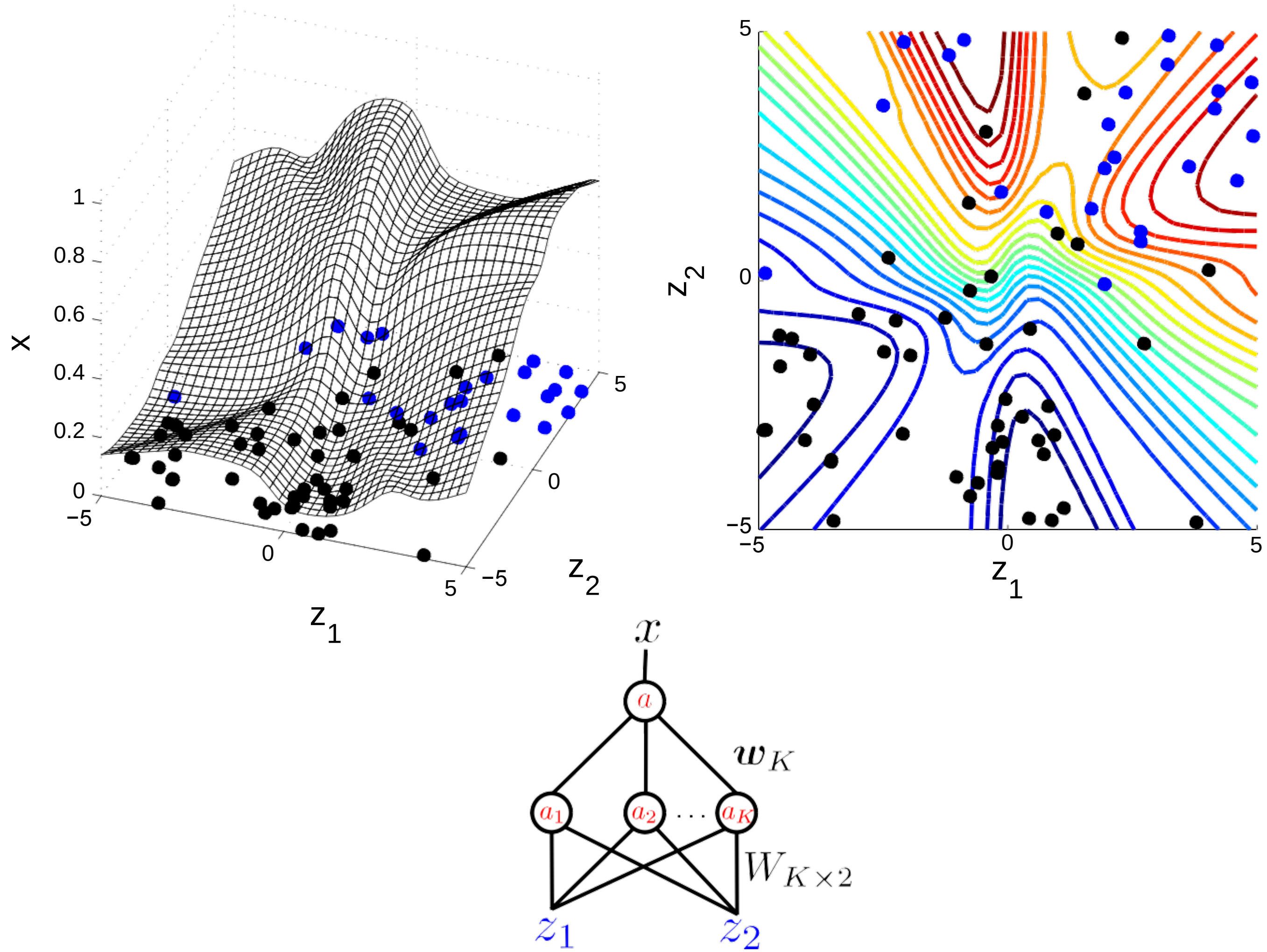
Training a Neural Network with a Single Hidden Layer



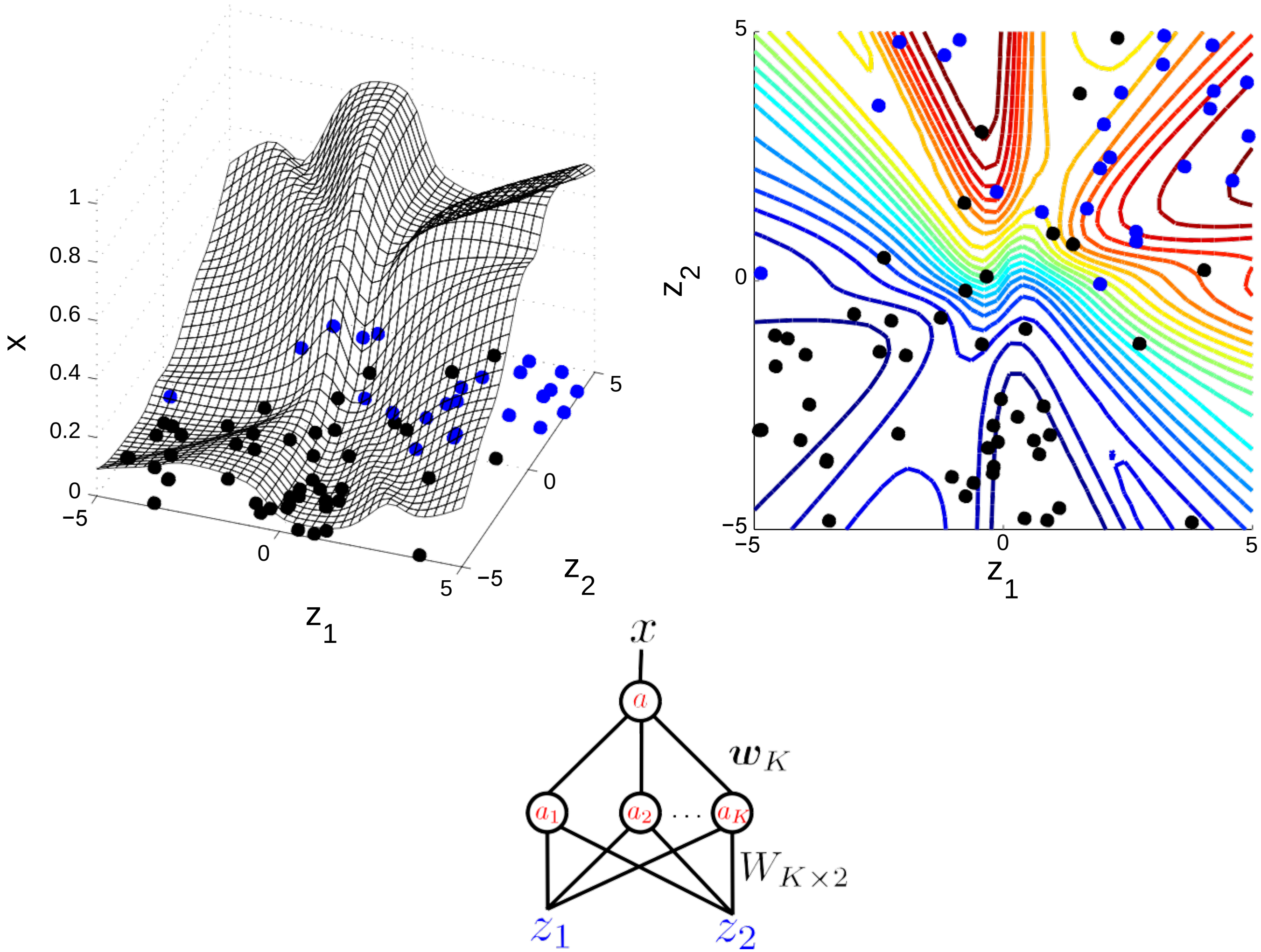




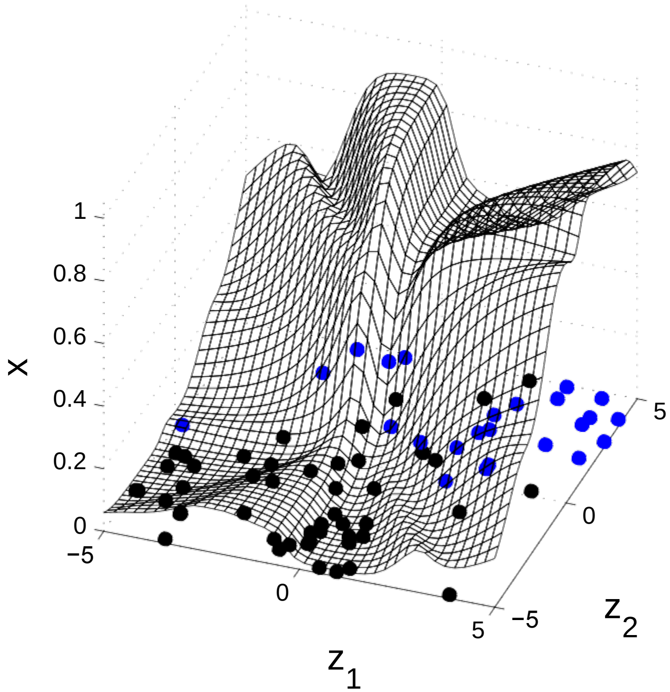
Training a Neural Network with a Single Hidden Layer

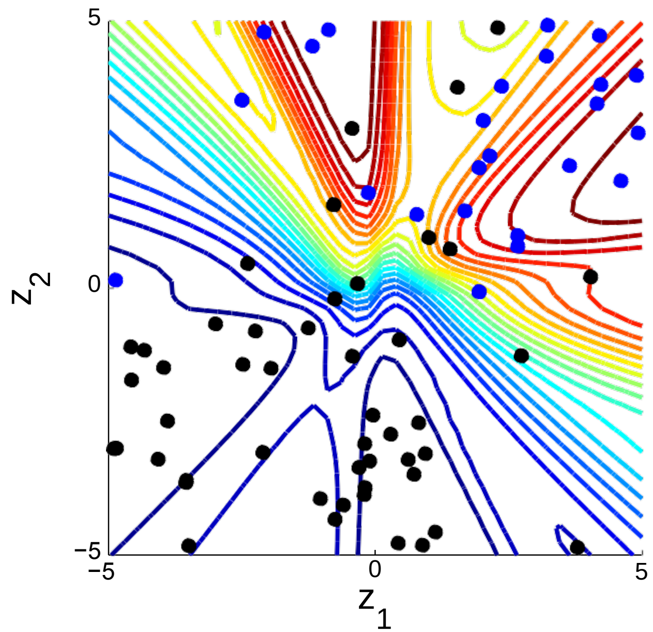


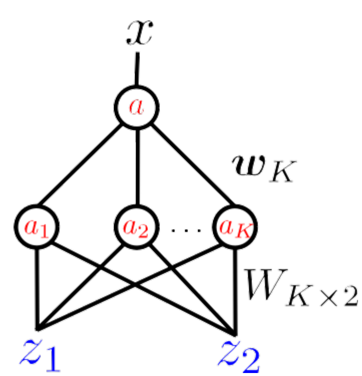
Training a Neural Network with a Single Hidden Layer



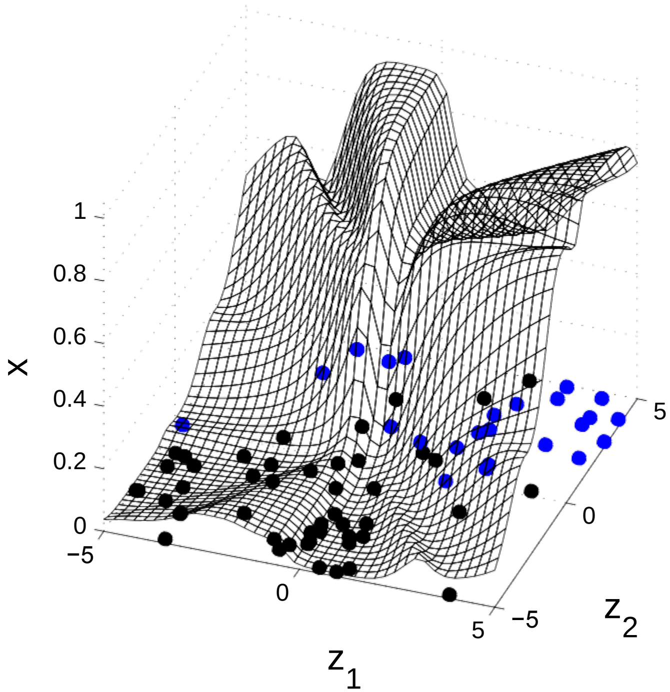
Training a Neural Network with a Single Hidden Layer

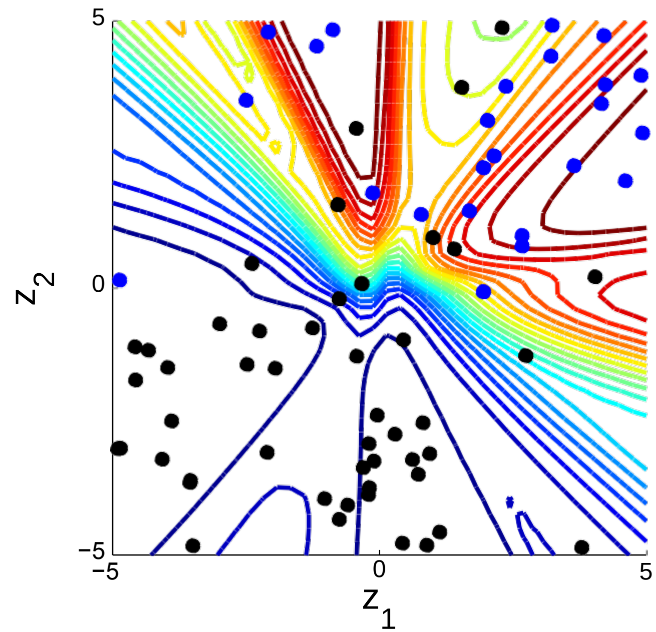


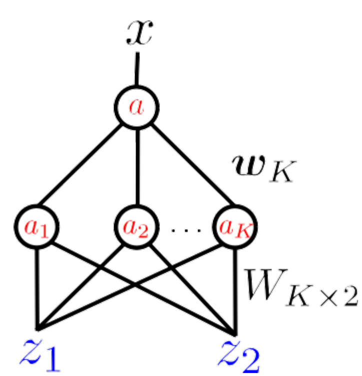




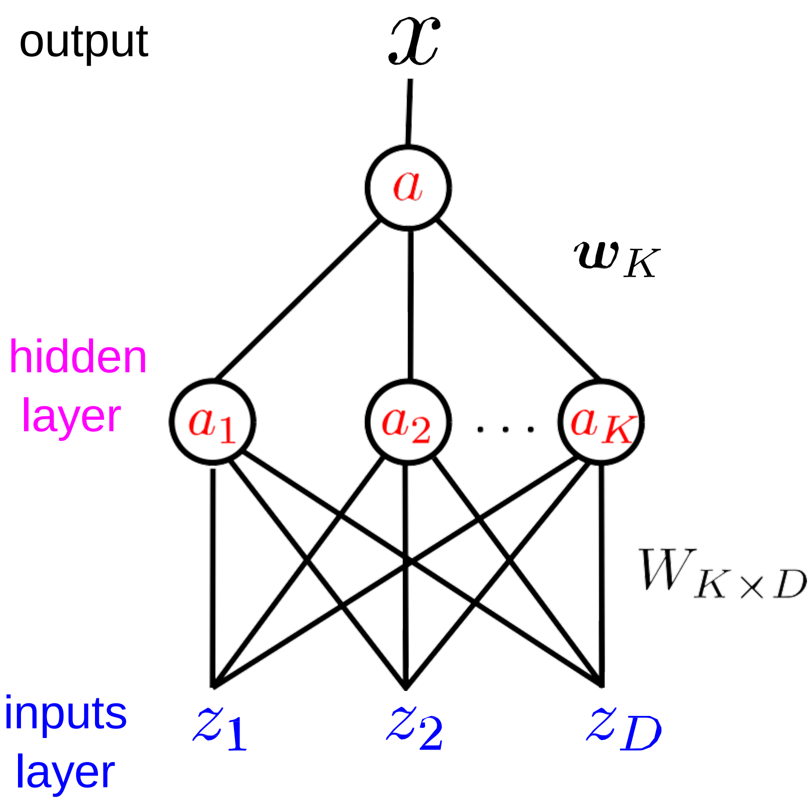
Training a Neural Network with a Single Hidden Layer

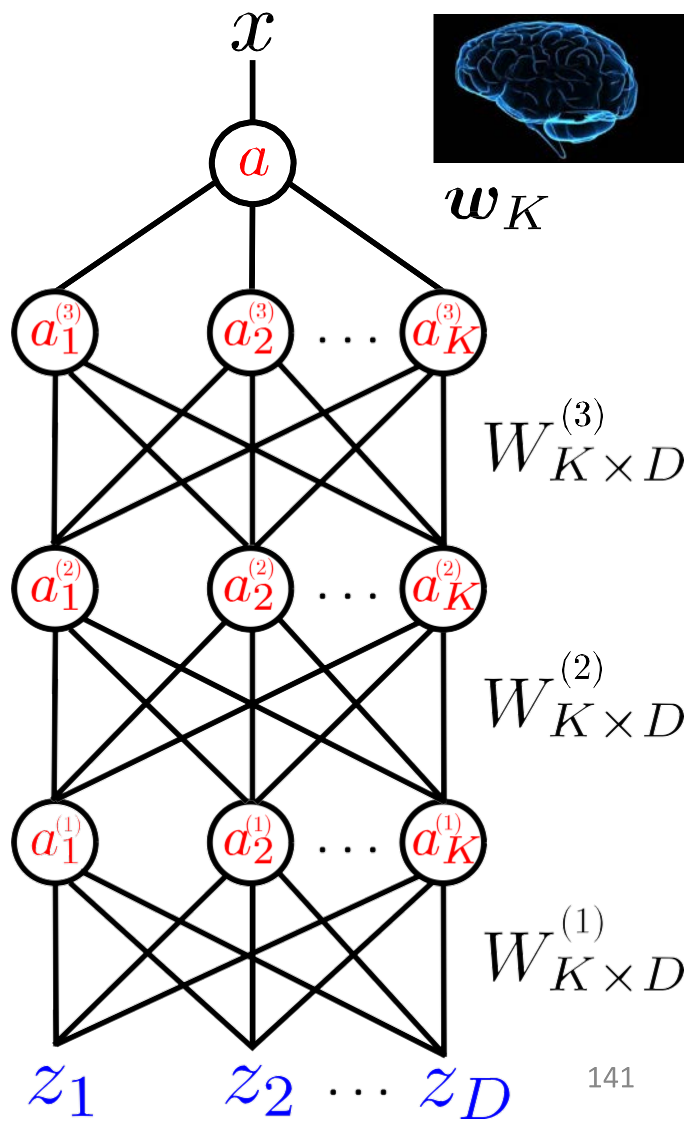






Hierarchical Models with Many Layers





What We Have Covered Today…

- Unsupervised vs. Supervised Learning
  - Clustering
  - Unsup. vs. Sup. Dimension Reduction
  - Training, testing, & validation
- Image Representation
  - Bag-of-Words Representation
  - Linear Classification
  - Intro to Neural Networks