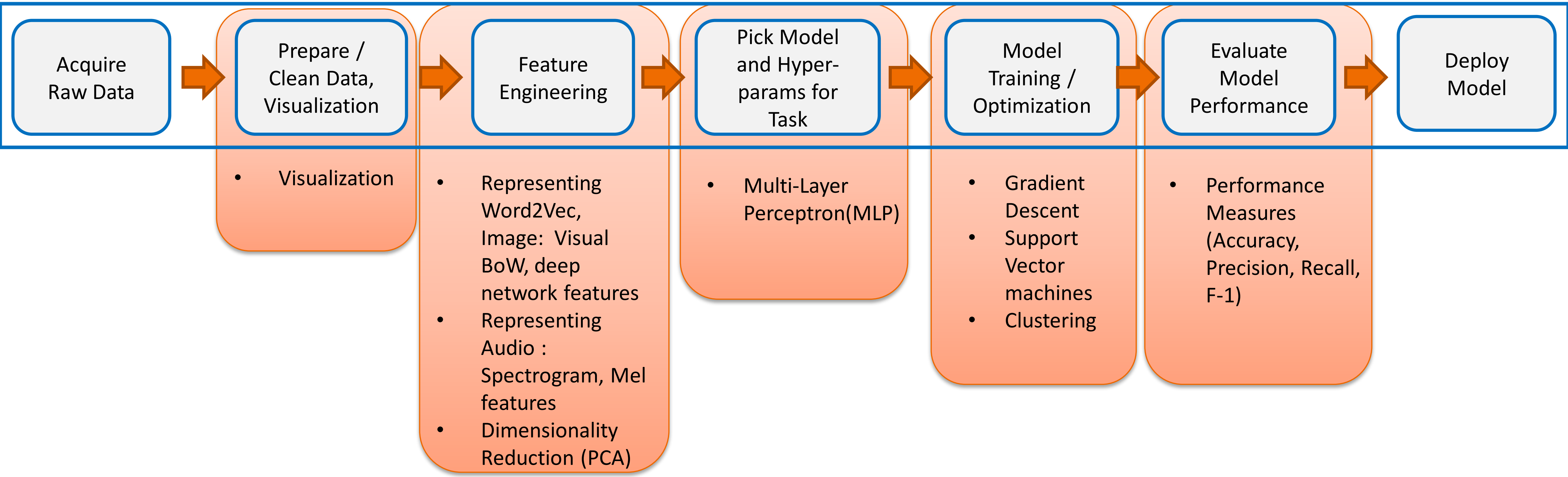
Focus of this lecture



Acquire Raw Data

Prepare / Clean Data, Visualization

- Visualization

Feature Engineering

- Representing Word2Vec, Image: Visual BoW, deep network features
- Representing Audio : Spectrogram, Mel features
- Dimensionality Reduction (PCA)

Pick Model and Hyper-params for Task

- Multi-Layer Perceptron(MLP)

Model Training / Optimization

- Gradient Descent
- Support Vector machines
- Clustering

Evaluate Model Performance

- Performance Measures (Accuracy, Precision, Recall, F-1)

Deploy Model

Today’s Plan

Session 1:

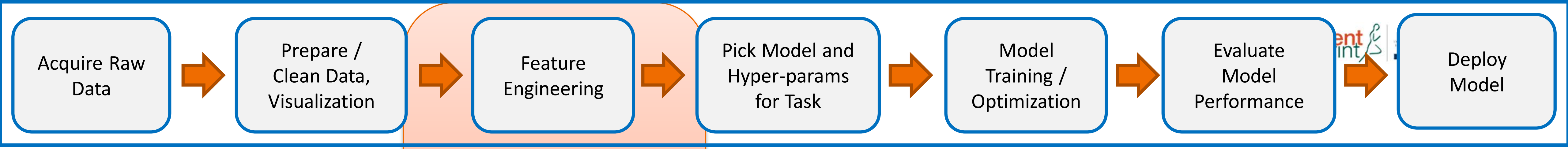
1. Representation
   - Word2Vec
   - Image: Visual BoW, deep network features
   - Audio : Spectrogram, Mel features
2. Dimensionality Reduction (PCA)
3. Visualization

Session 2:

5. Performance Metrics
6. Gradient Descent
7. MLP
8. SVM with Kernels
9. Clustering

Session 3:

- Paper Reading



Acquire Raw Data

Prepare / Clean Data, Visualization

Feature Engineering

Pick Model and Hyper-params for Task

Model Training / Optimization

Evaluate Model Performance

Deploy Model

- Representing Image: Word2Vec, Visual BoW, deep network features
- Representing Audio : Spectrogram, Mel features
- Dimensionality Reduction (PCA)

Feature Engineering

Word2vec - Intuition

- Marco saw a furry little wampimuk hiding in the tree

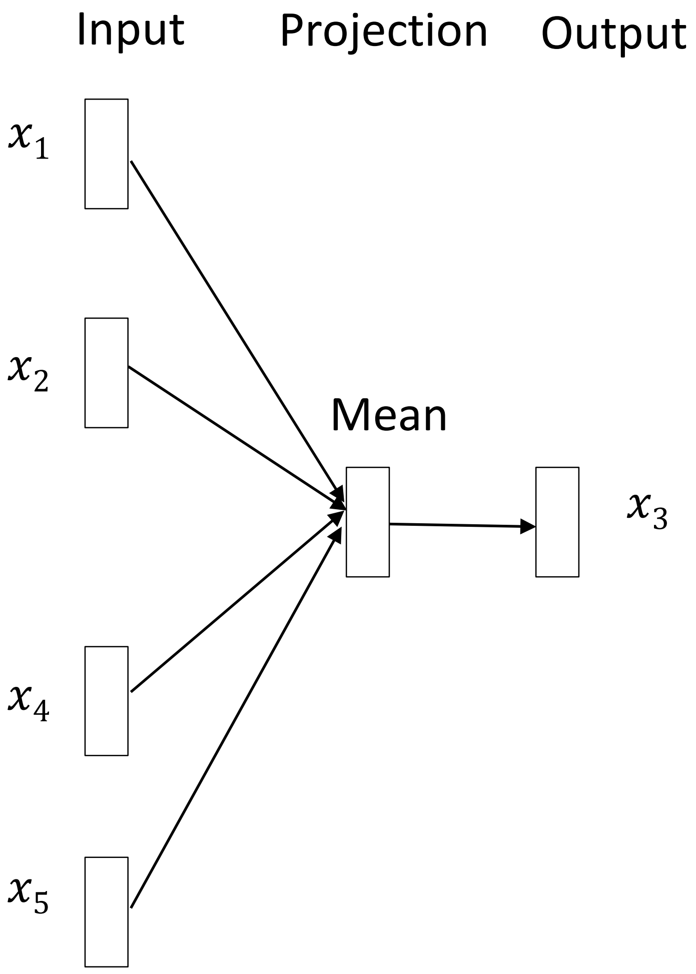
- **What is Marco?**
- **What is wampimuk?**

A person

Animal

Variants

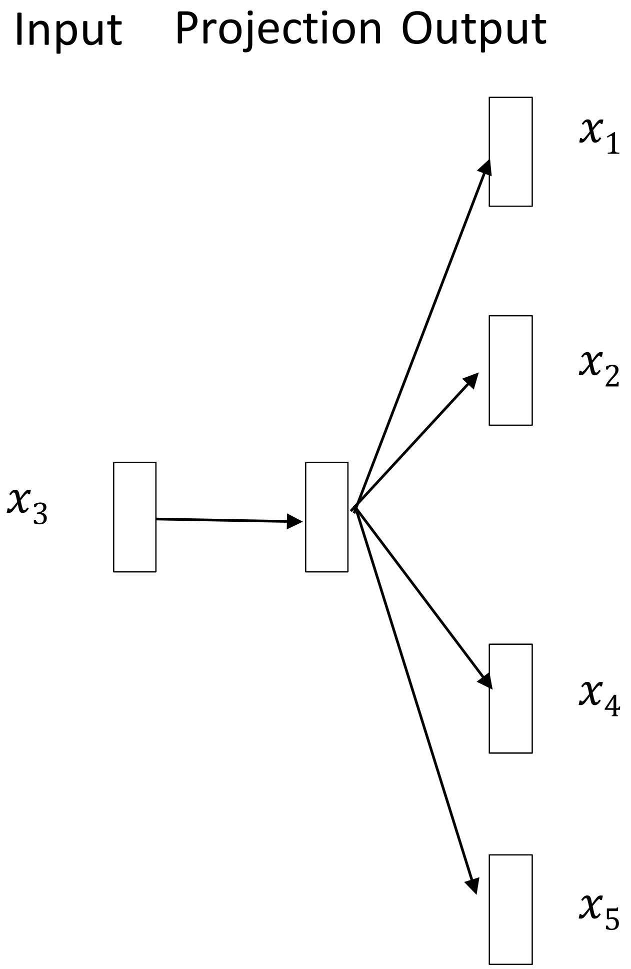
- Continuous Bag of Words (CBOW):
  - – use a window of words to predict the missing word



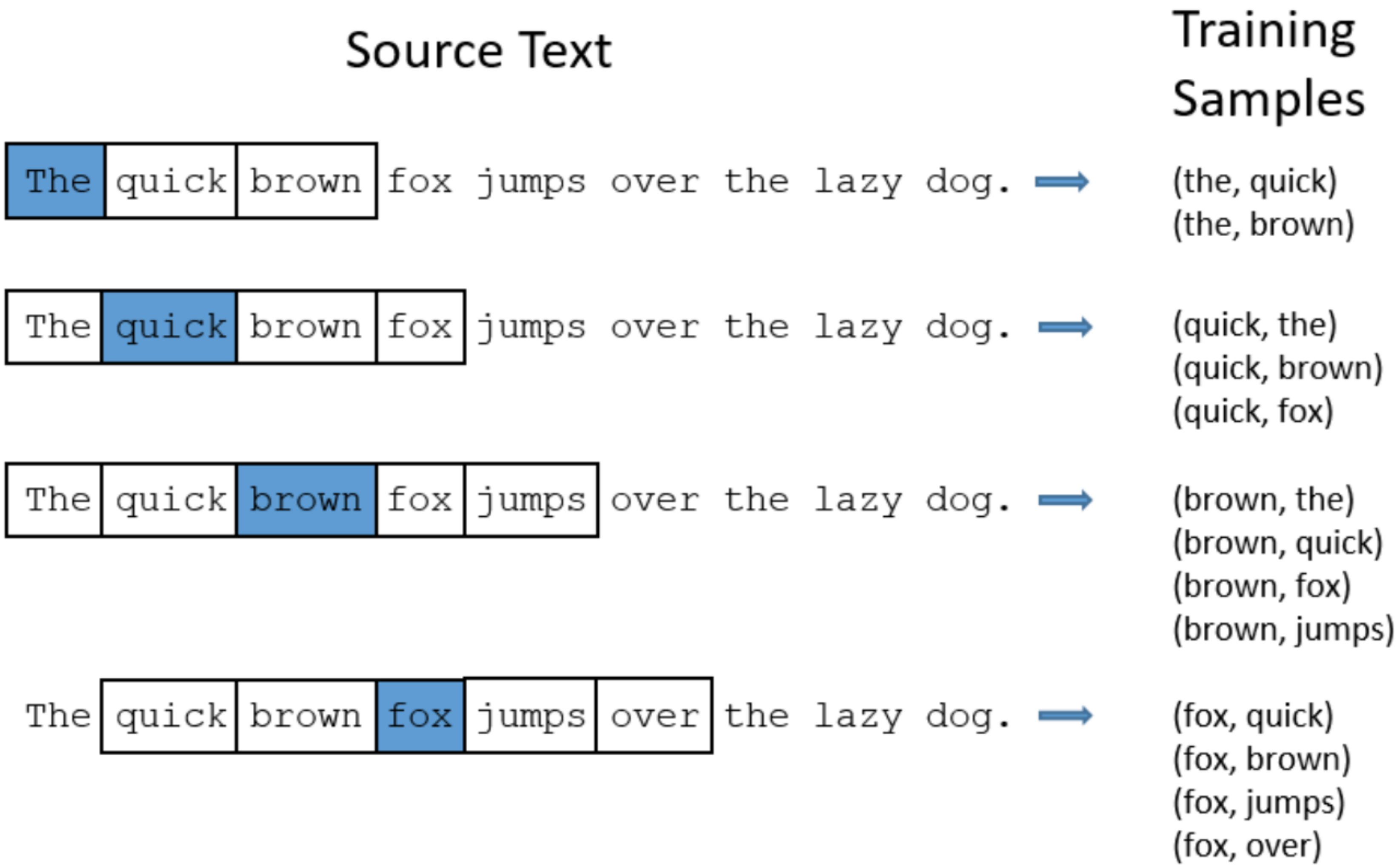
CBOW

Variants

- Skip-gram (SG):
  - use a word to predict the surrounding ones in the specified window.



Word2vec – skip gram examples



Source Text

Training Samples

The quick brown fox jumps over the lazy dog. →

(the, quick)
(the, brown)

The quick brown fox jumps over the lazy dog. →

(quick, the)
(quick, brown)
(quick, fox)

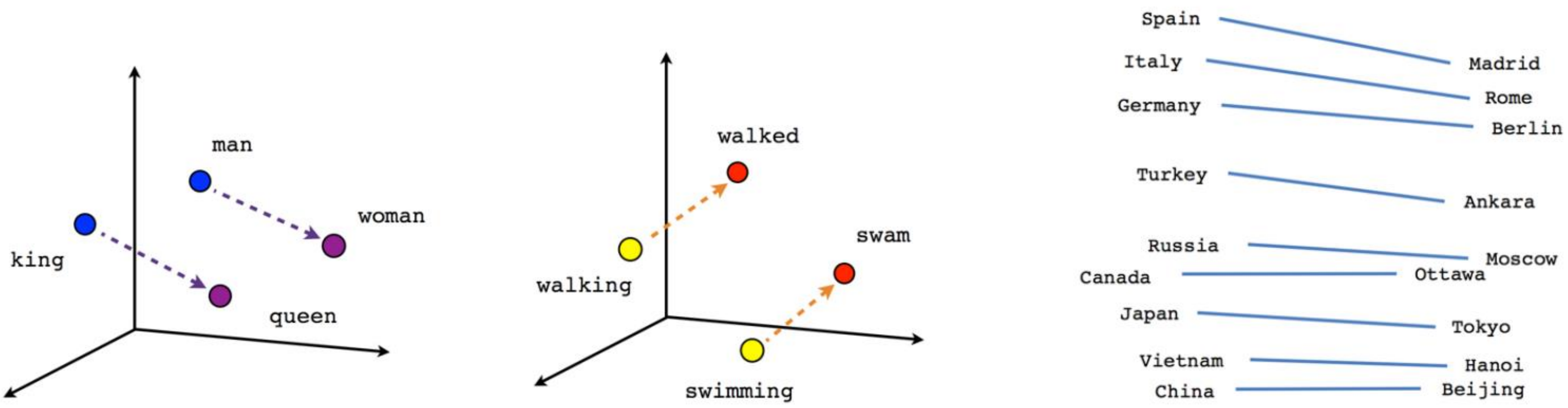
The quick brown fox jumps over the lazy dog. →

(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)

The quick brown fox jumps over the lazy dog. →

(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

Examples



Male-Female

Verb tense

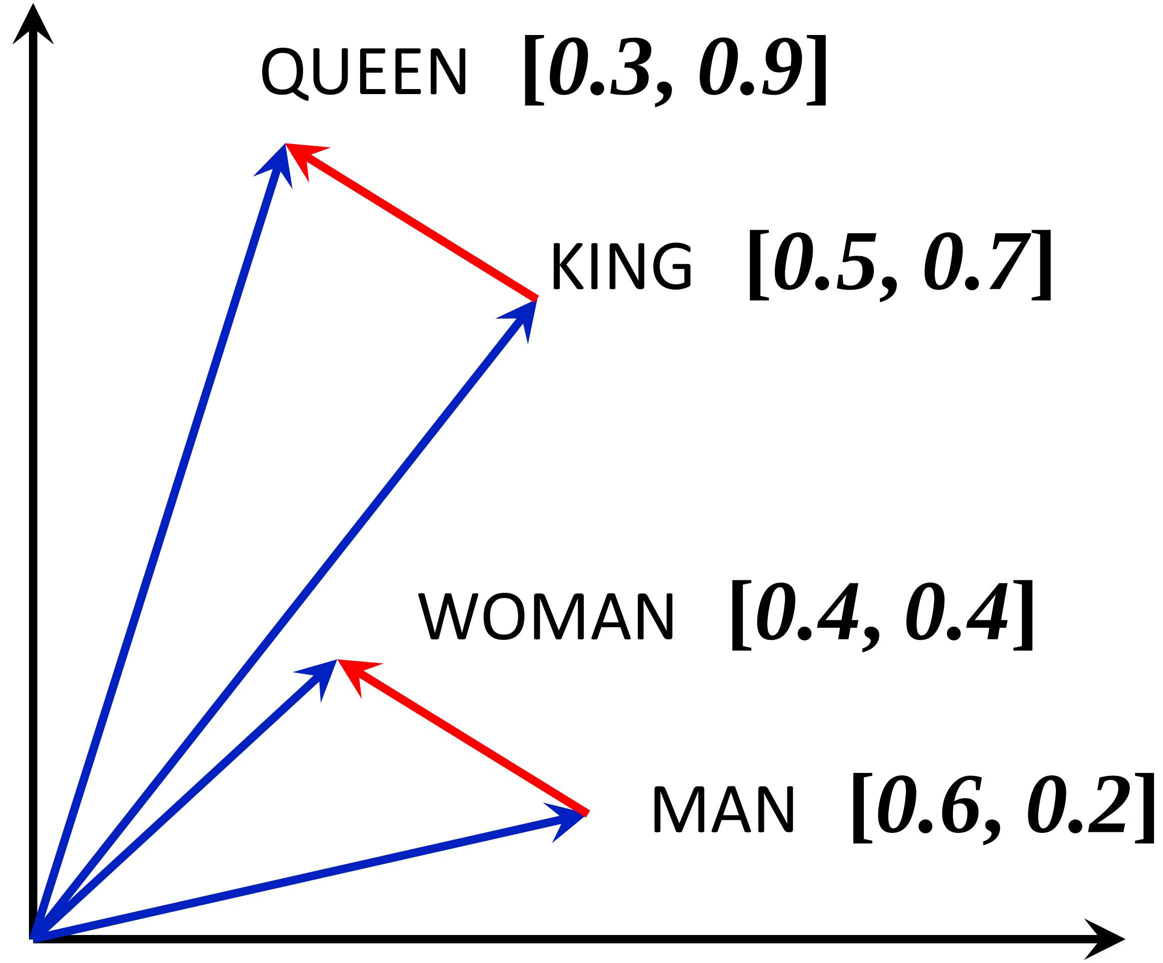
Country-Capital

https://www.tensorflow.org/images/linear-relationships.png

Visualising “Word-Arithmetic”

$$\boldsymbol{Vec}(\boldsymbol{Queen}) = \boldsymbol{Vec}(\boldsymbol{King}) - \boldsymbol{Vec}(\boldsymbol{Man}) + \boldsymbol{Vec}(\boldsymbol{Woman})$$

- These fancy arithmetic were imagined (and shown) earlier also; but became popular with Word2Vec.



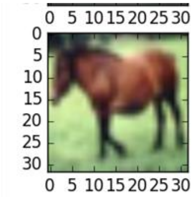
More Examples

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

What words have embeddings closest to a given word? From Collobert et al. (2011) (https://arxiv.org/pdf/1103.0398v1.pdf)

AI and Problem of Perception

Possible Features: Handcrafting



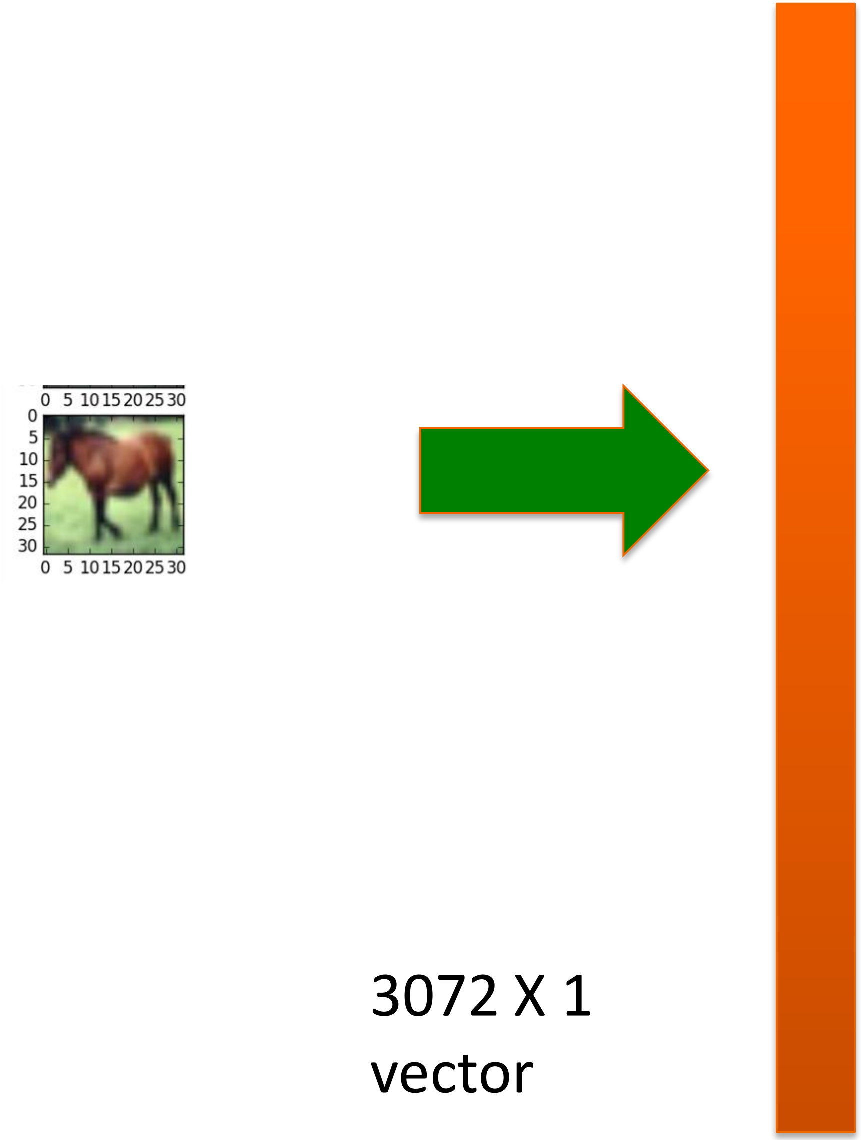
MIN RED
MAX RED
MEAN RED
MIN GREEN
MAX GREEN
MEAN GREEN
MIN BLUE
MAX BLUE
MEAN BLUE

9 X 1
FEATURE VECTOR
PER IMAGE

Concerns:

- **Too naïve to capture the visual content?**
- **Too small to represent information?**

Possible Features: Raw Data Itself



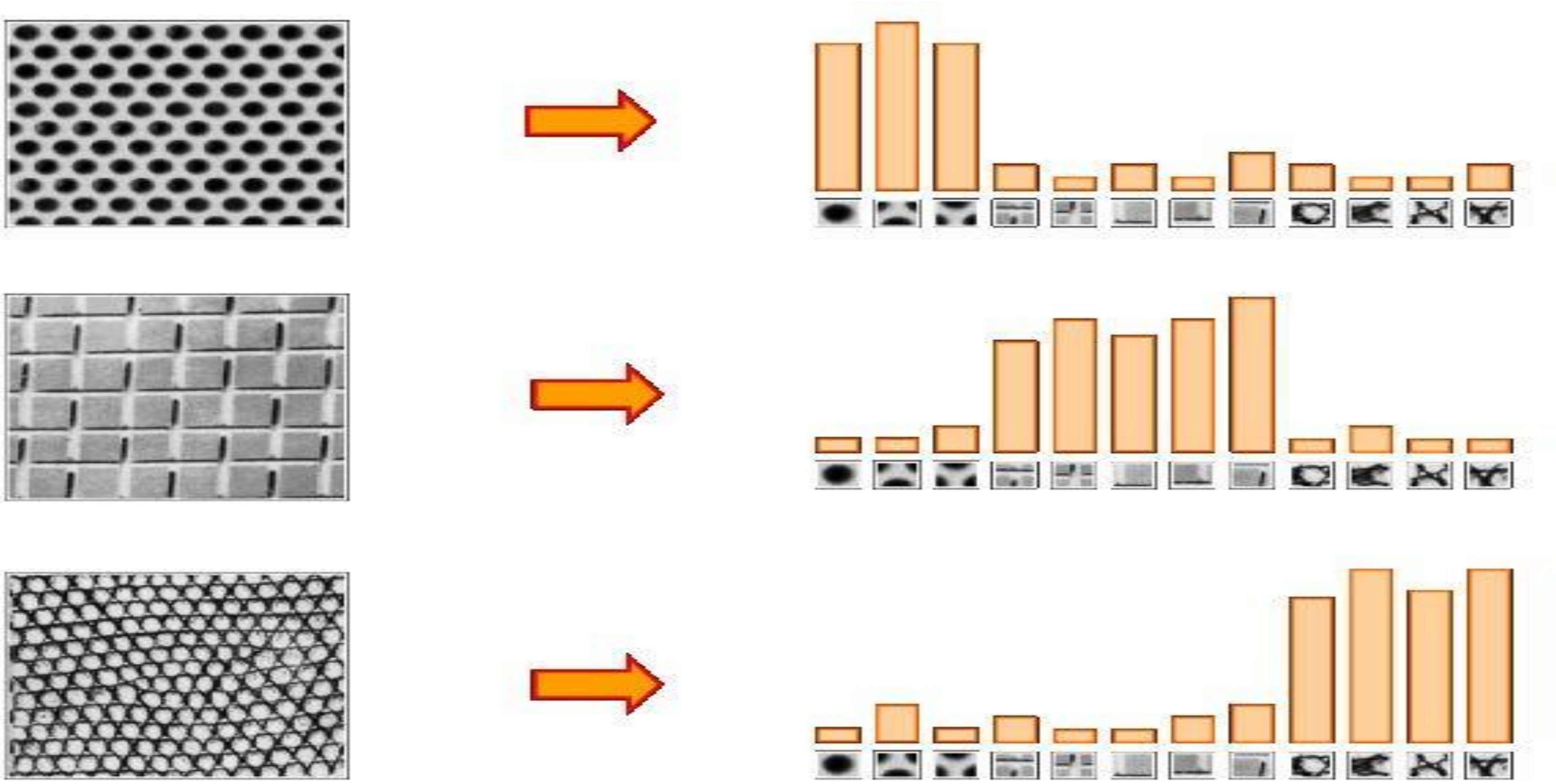
3072 X 1
vector

FEATURE VECTOR
32 X 32 X 3 = 3072
DIMENSION
PER IMAGE (d = 3072)

CONCERNS:

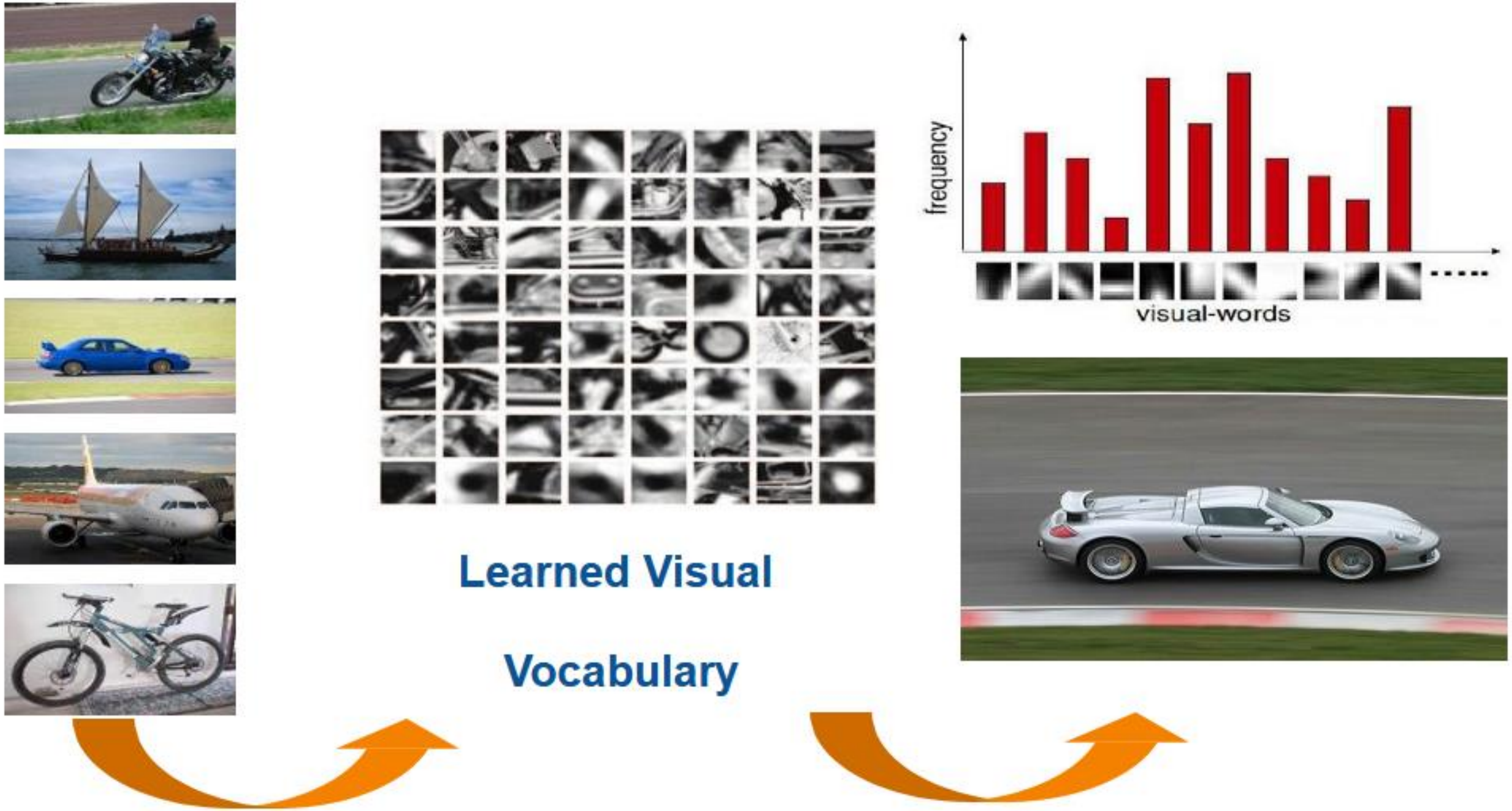
- Too big ?
- May be redundancy ?
- Too rigid?

Visual BoW: Basic Idea



Source: wordpress.com/2013/04/24/research-bag-of-features-for-visual-recognition

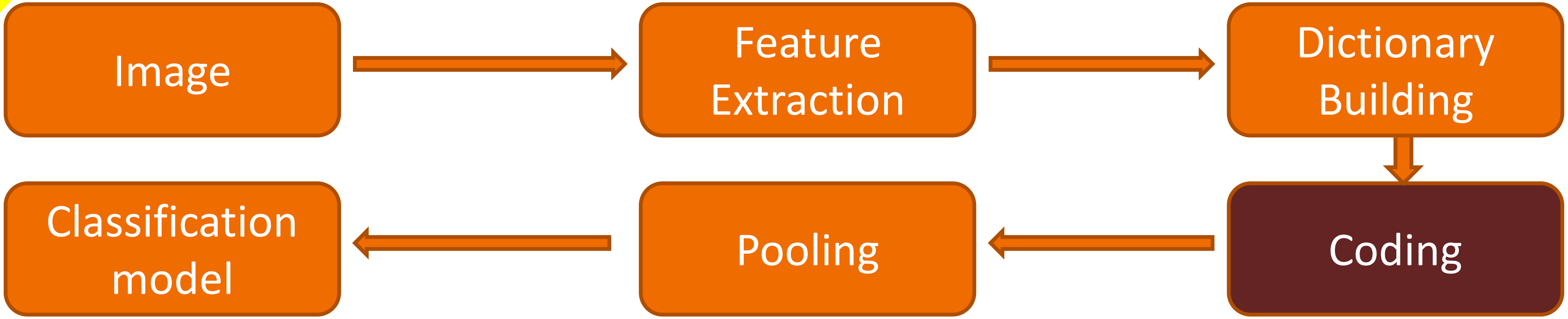
Bag of Visual Words



Learned Visual

Vocabulary

Code Available



Image

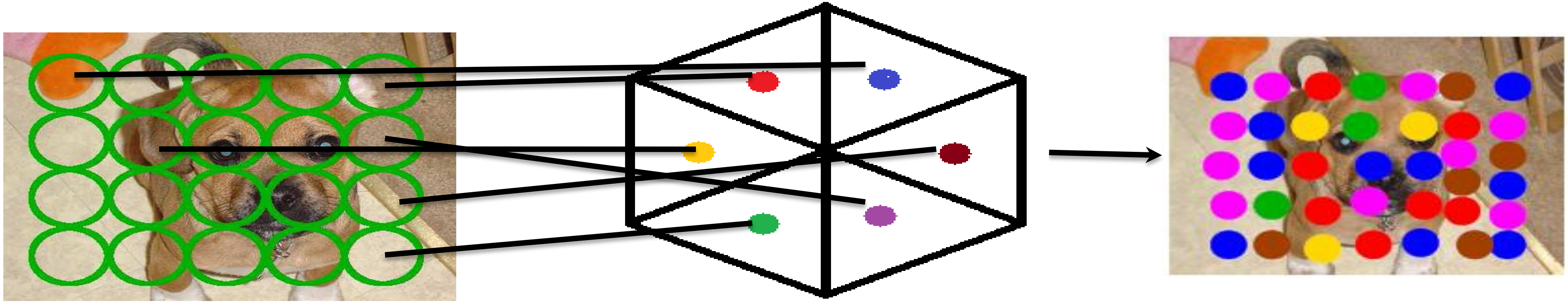
Feature Extraction

Dictionary Building

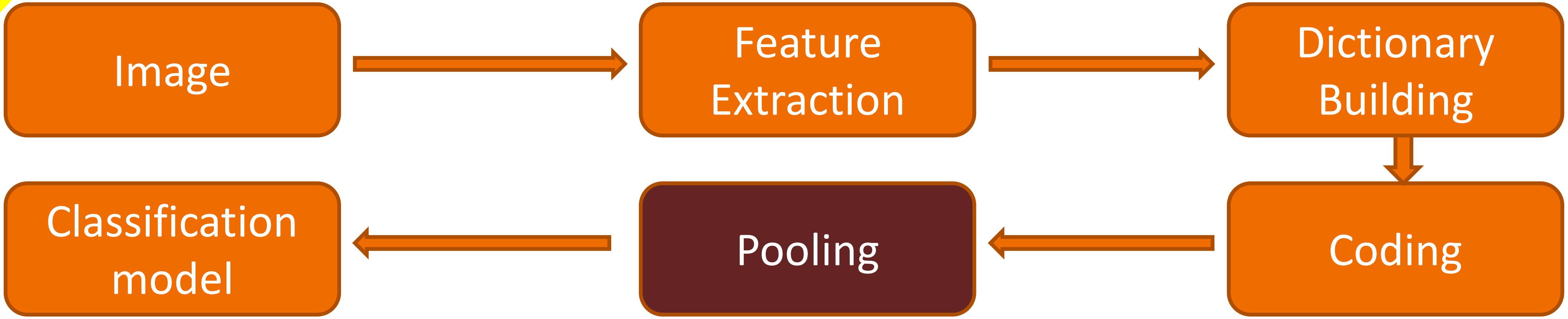
Coding

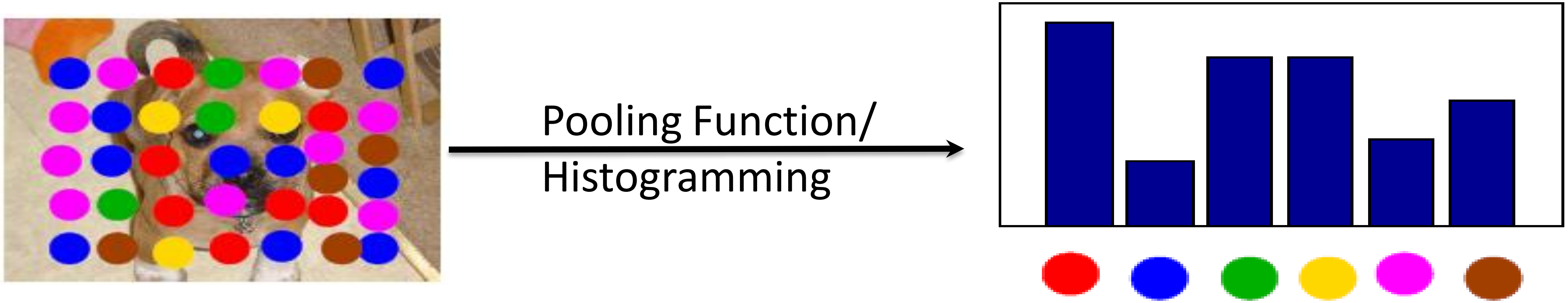
Pooling

Classification model


Dictionary/Codebook

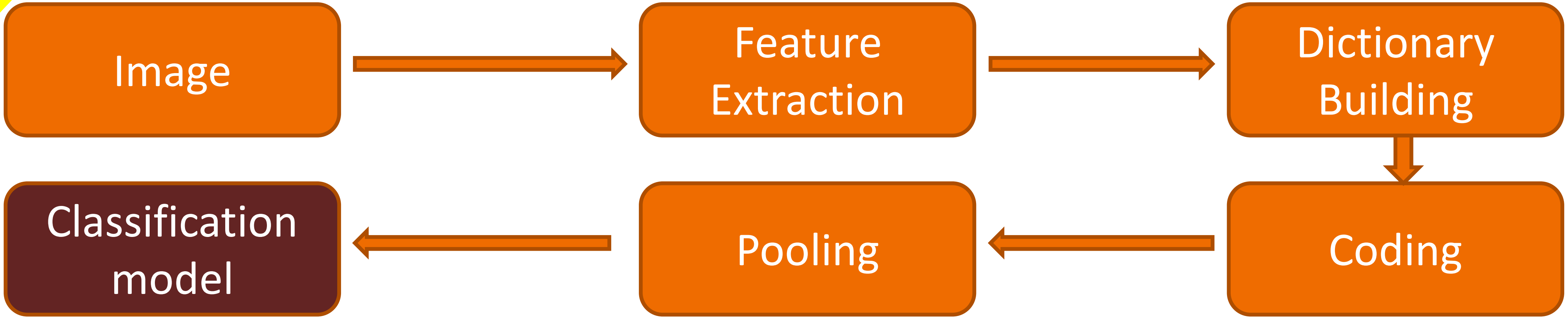
Code Available





Histogram of Visual Words / Image Representation

Code Available



Image

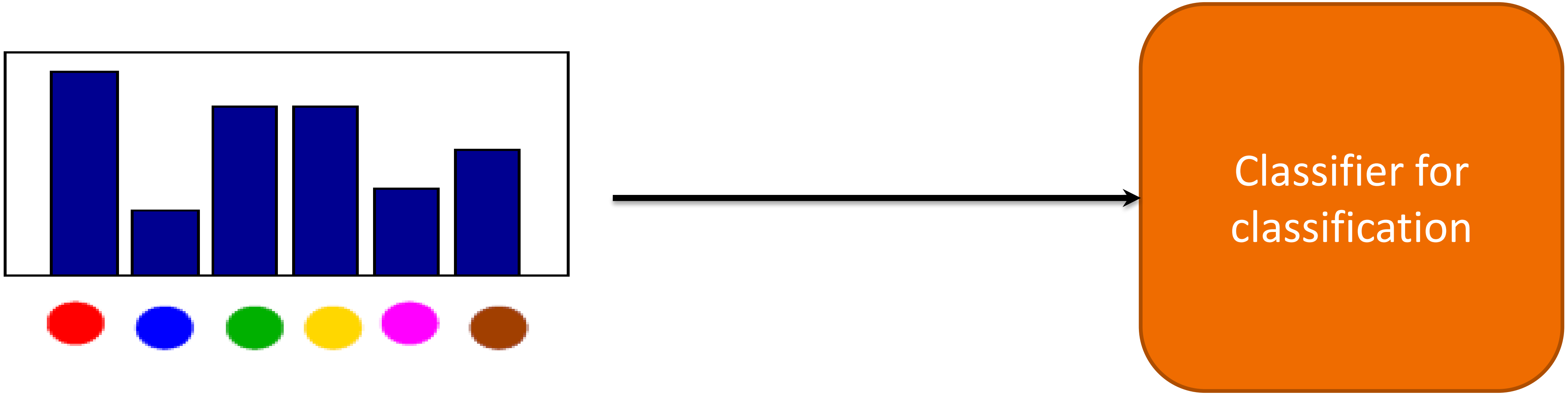
Feature Extraction

Dictionary Building

Coding

Pooling

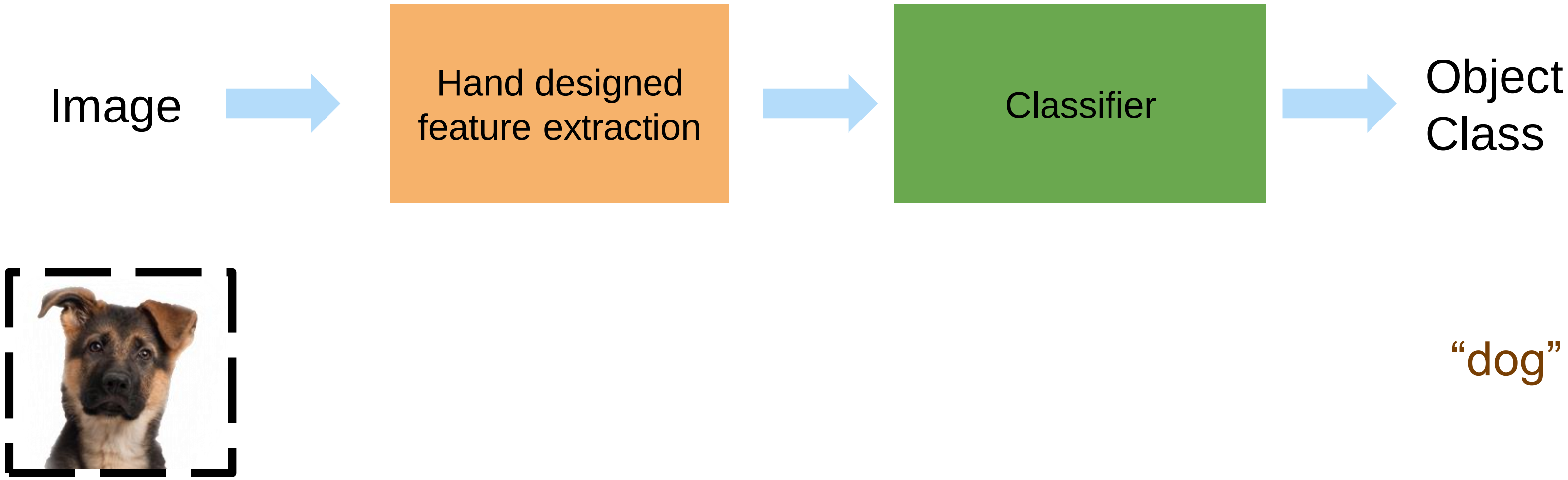
Classification model



Classifier for classification

Histogram of Visual Words / Image Representation

Object-recognition: conventional approach



Image

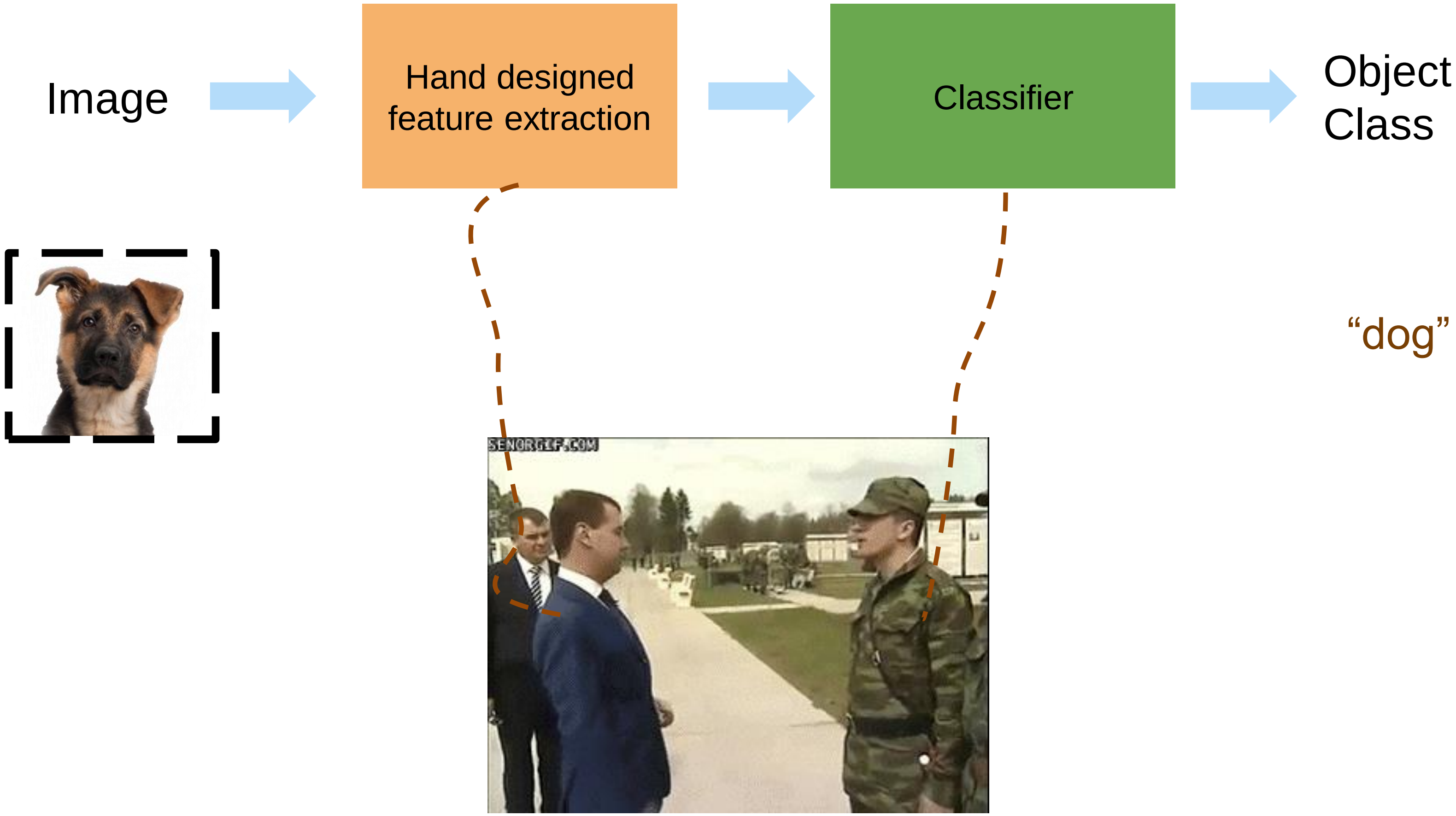
Hand designed feature extraction

Classifier

Object Class

“dog”

Object-recognition: conventional approach



Image

Hand designed feature extraction

Classifier

Object Class

“dog”

Object Recognition: Deep Neural Networks

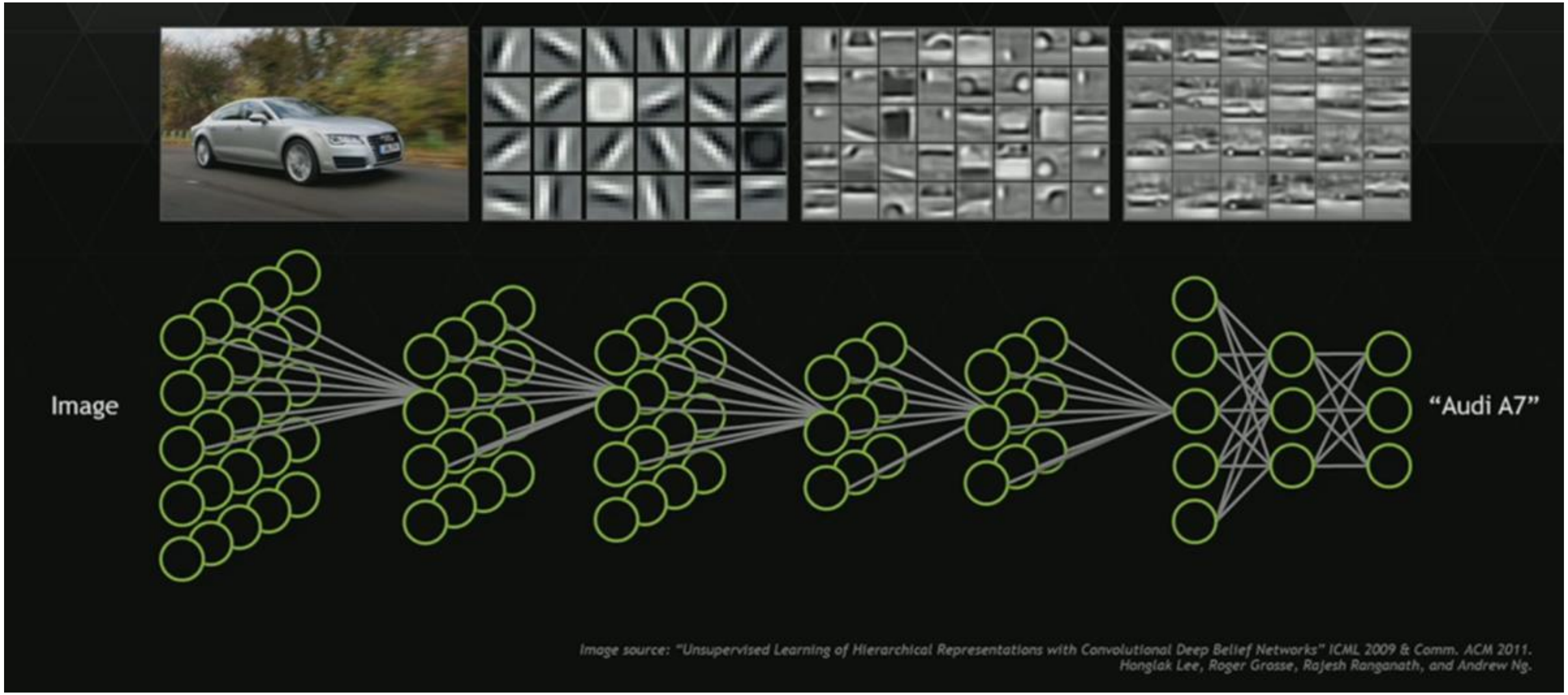


*Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICML 2009 & Comm. ACM 2011.
Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng.*

Data-driven, End-to-End learning, Task-specific feature hierarchy

Summary

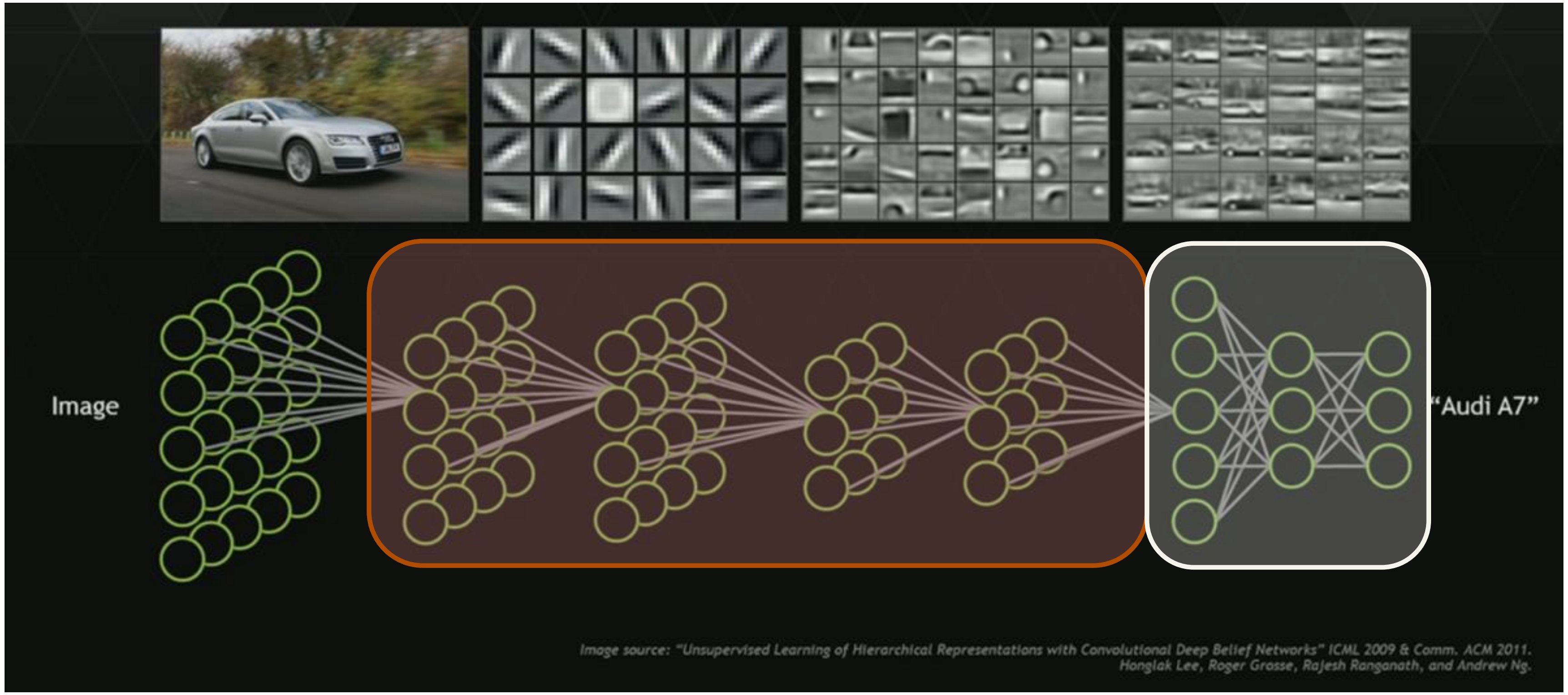
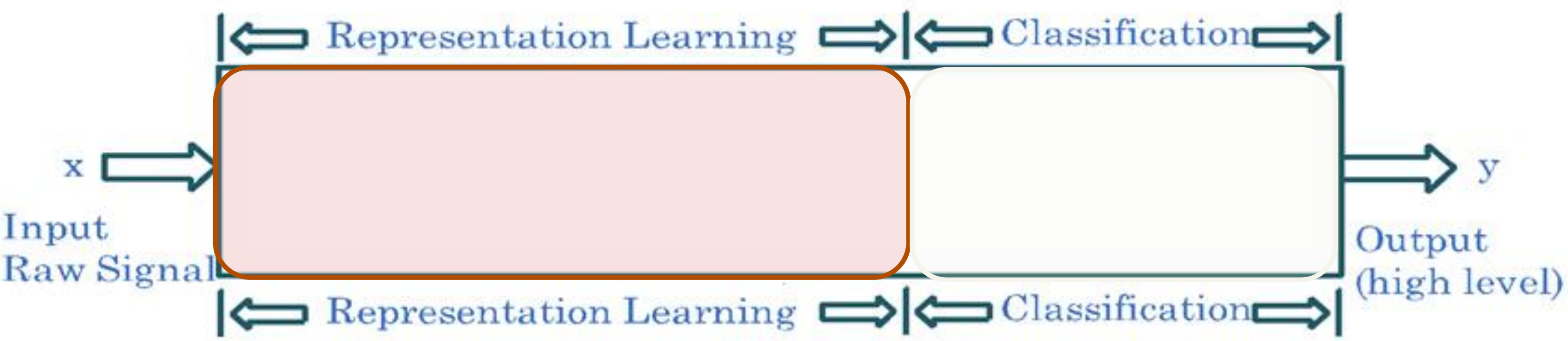
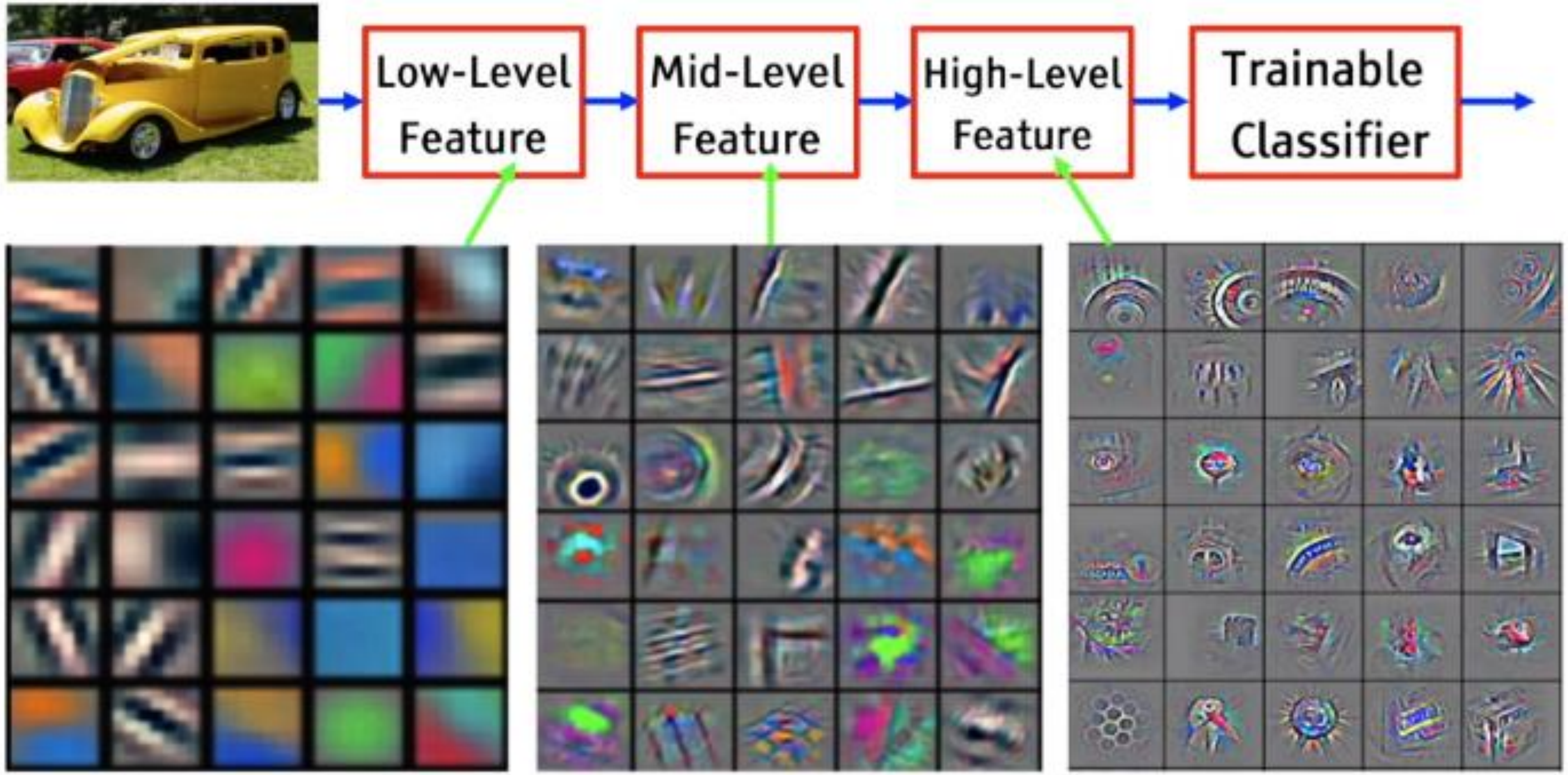


Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICML 2009 & Comm. ACM 2011. Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng.

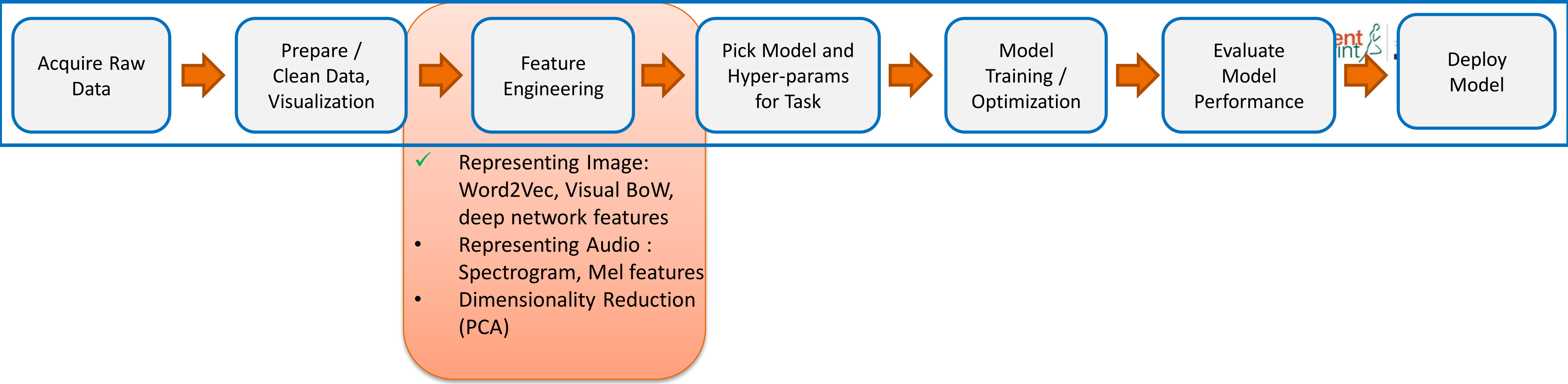


Deep Learnt Features (2013-XXX)

- It's deep if it has more than one stage of non-linear feature transformation.



Source: Yann LeCun



Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

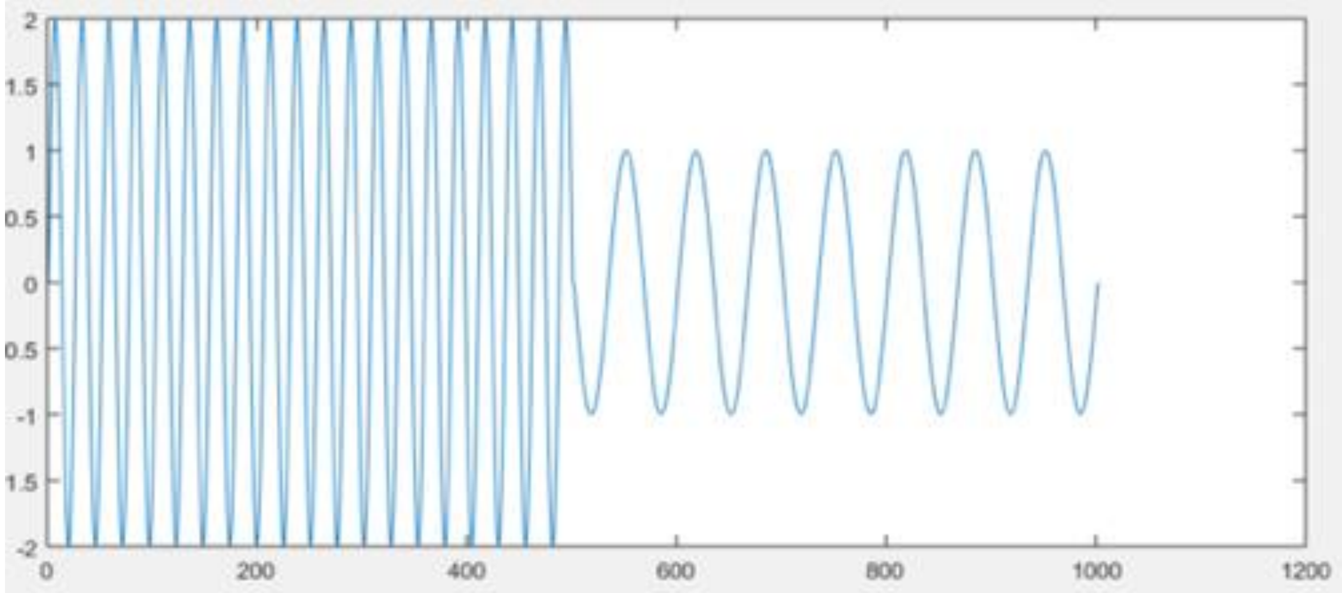
- ✓ Representing Image: Word2Vec, Visual BoW, deep network features
- Representing Audio : Spectrogram, Mel features
- Dimensionality Reduction (PCA)

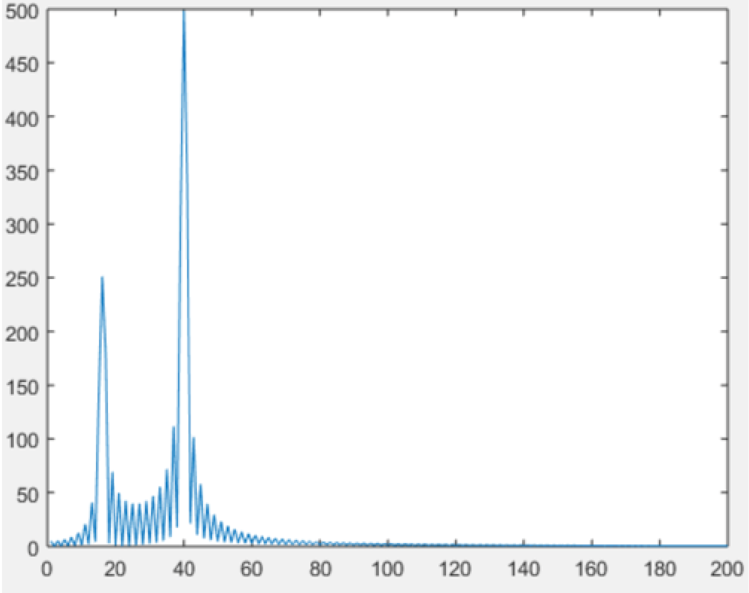
Speech

(Brief Explanation)

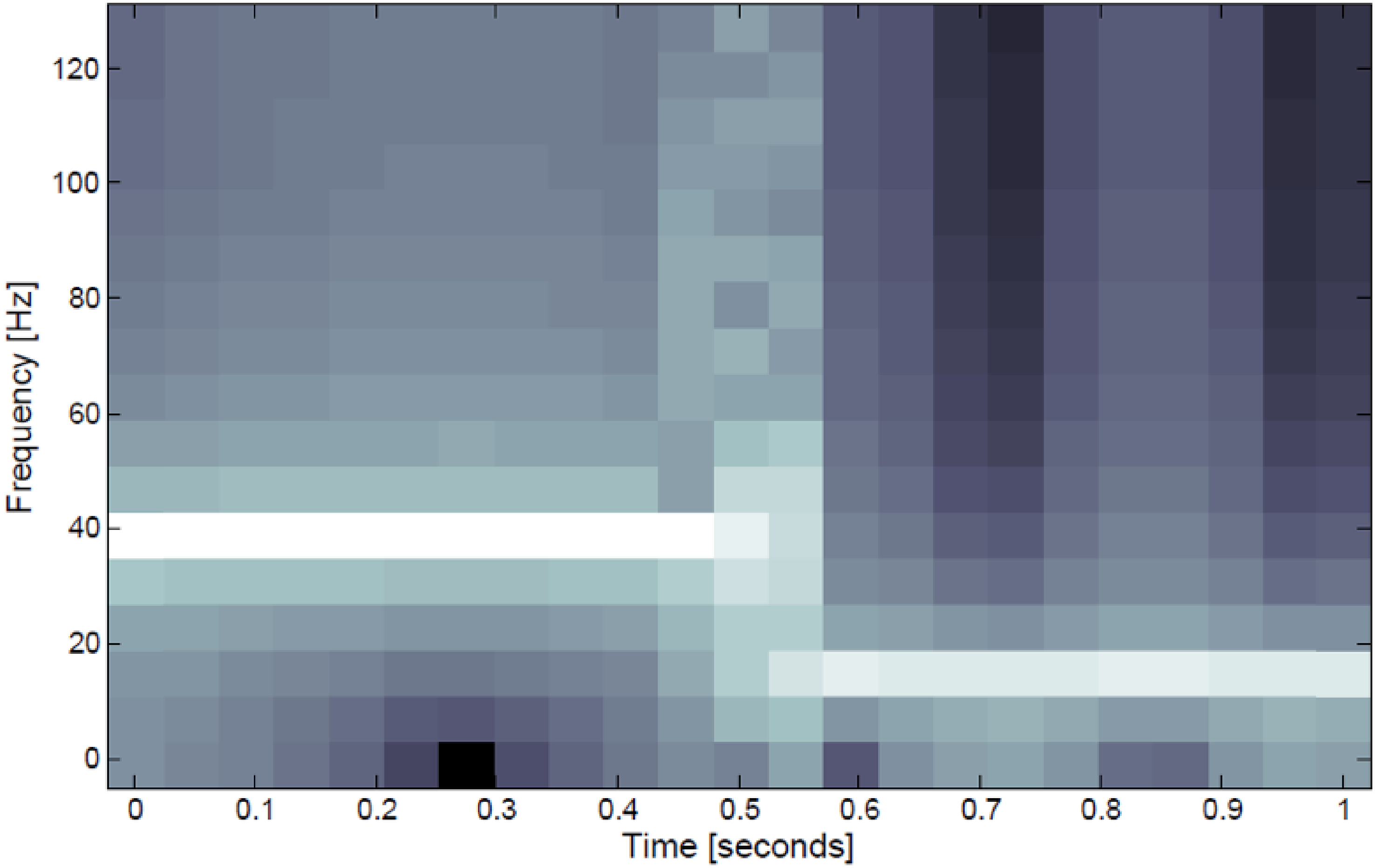
Example Sound Signal

$$g(t) = \begin{cases} 2 * \sin(2\pi \cdot 39t)\,, 0 \leq t \leq 1/2 \\ \sin(2\pi \cdot 15t)\,, 1/2 < t \leq 1 \end{cases}$$





Spectrogram



Spectrogram of a piecewise monochromatic signal.

Lighter color ⍰ greater DFT magnitude

Representations

A: MFCC (Signal processing based; Classical)

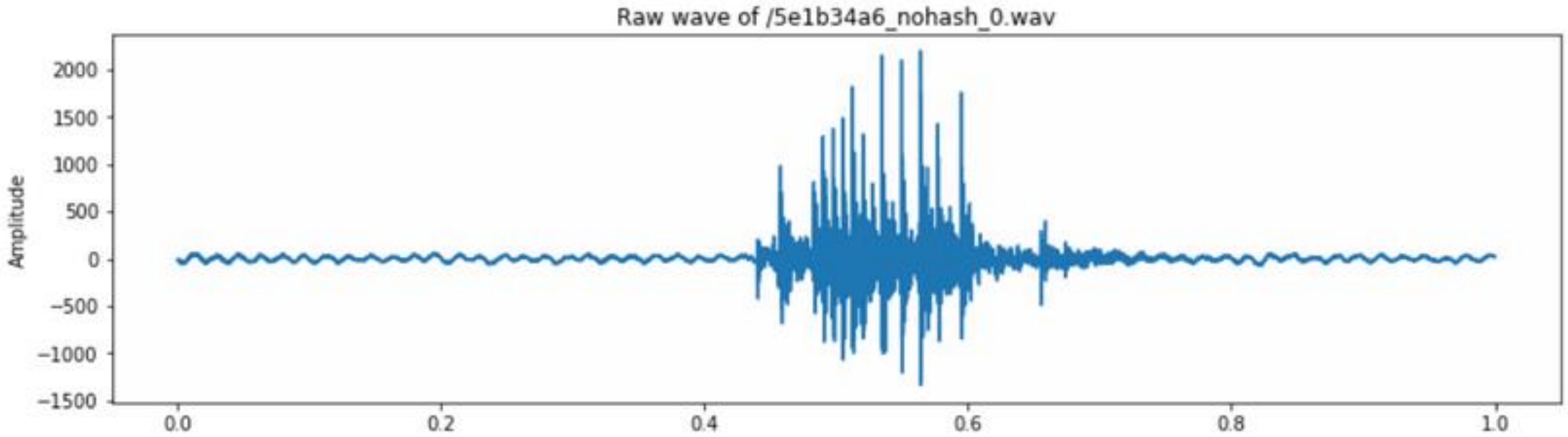
- Mel Frequency Cepstral Coefficients

B: CNN Based (Modern)

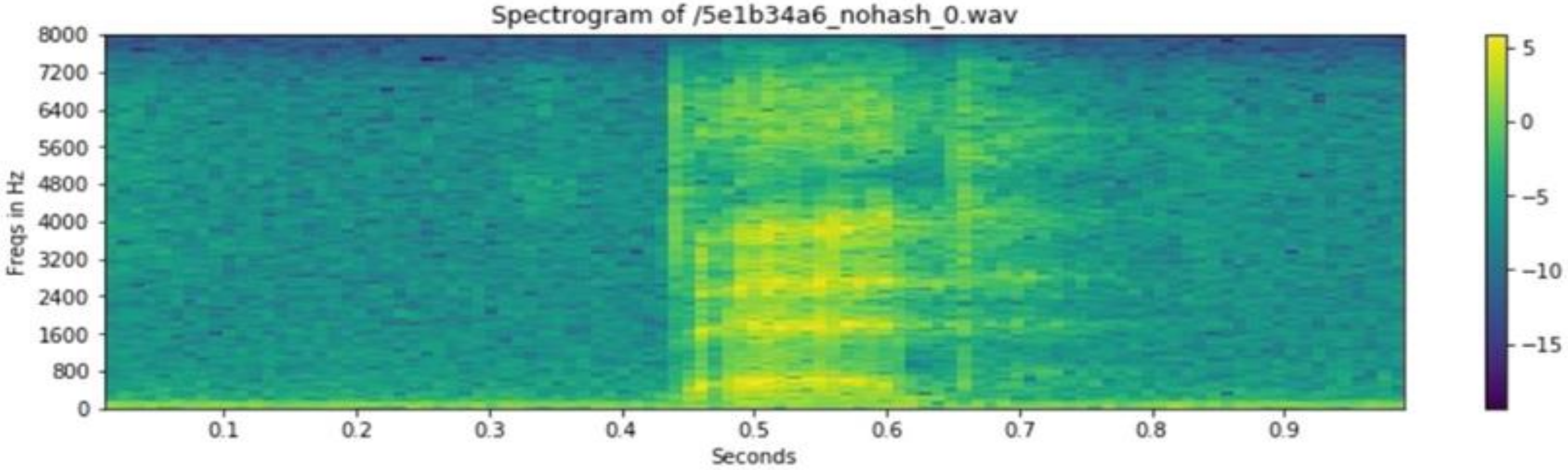
- VGG Features on the Mel Spectrogram

http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

Classical Feature (MFCC)

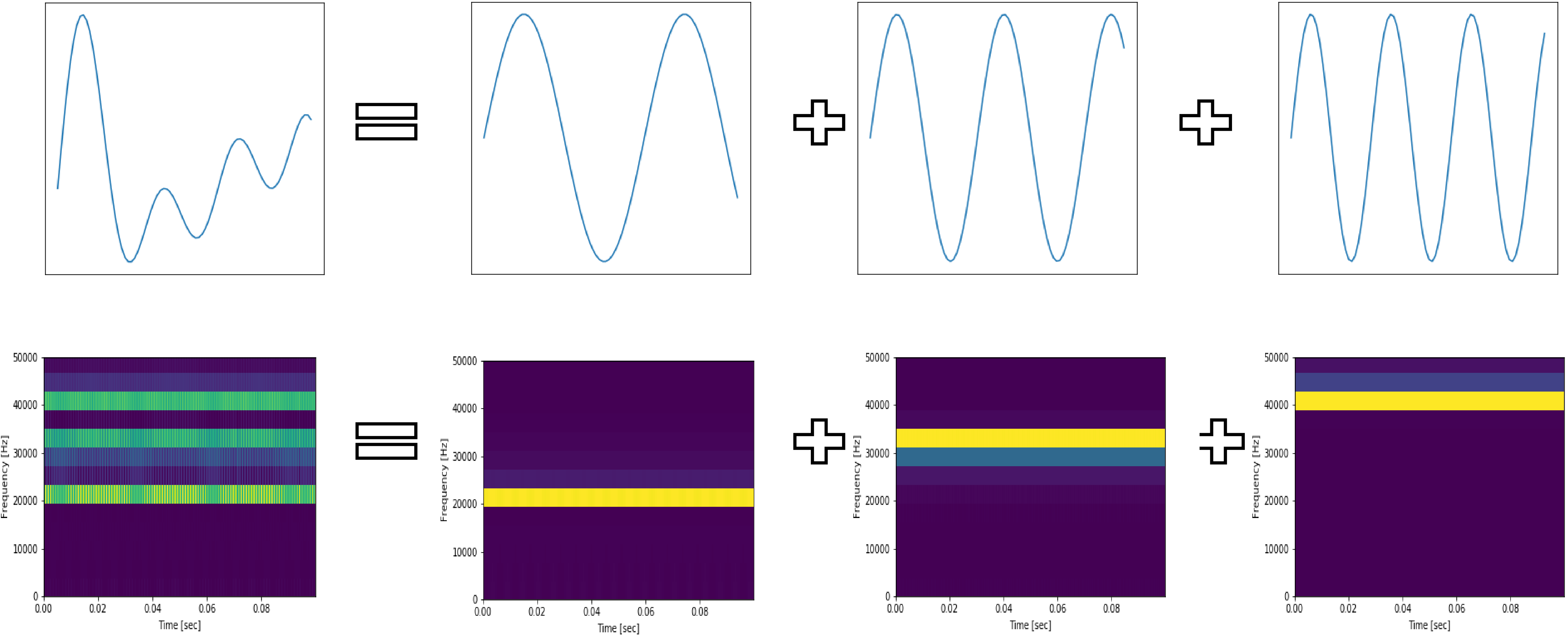


Amplitude
Vs
Time

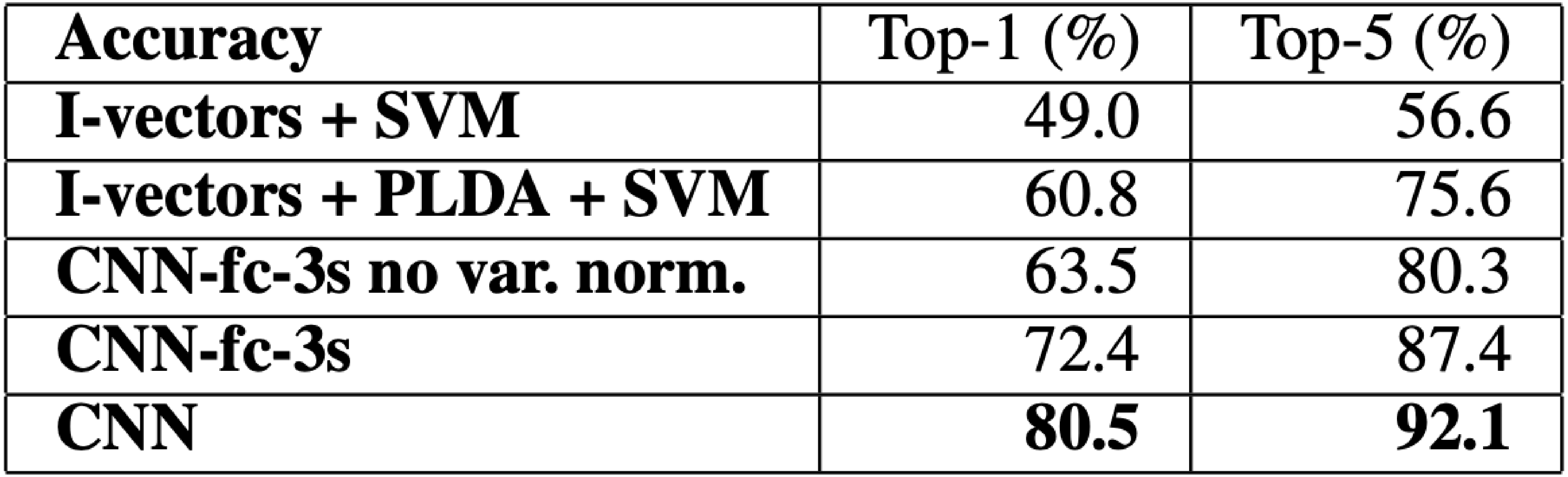


Frequency
Vs
Time

Any wave is a combination of many sine waves

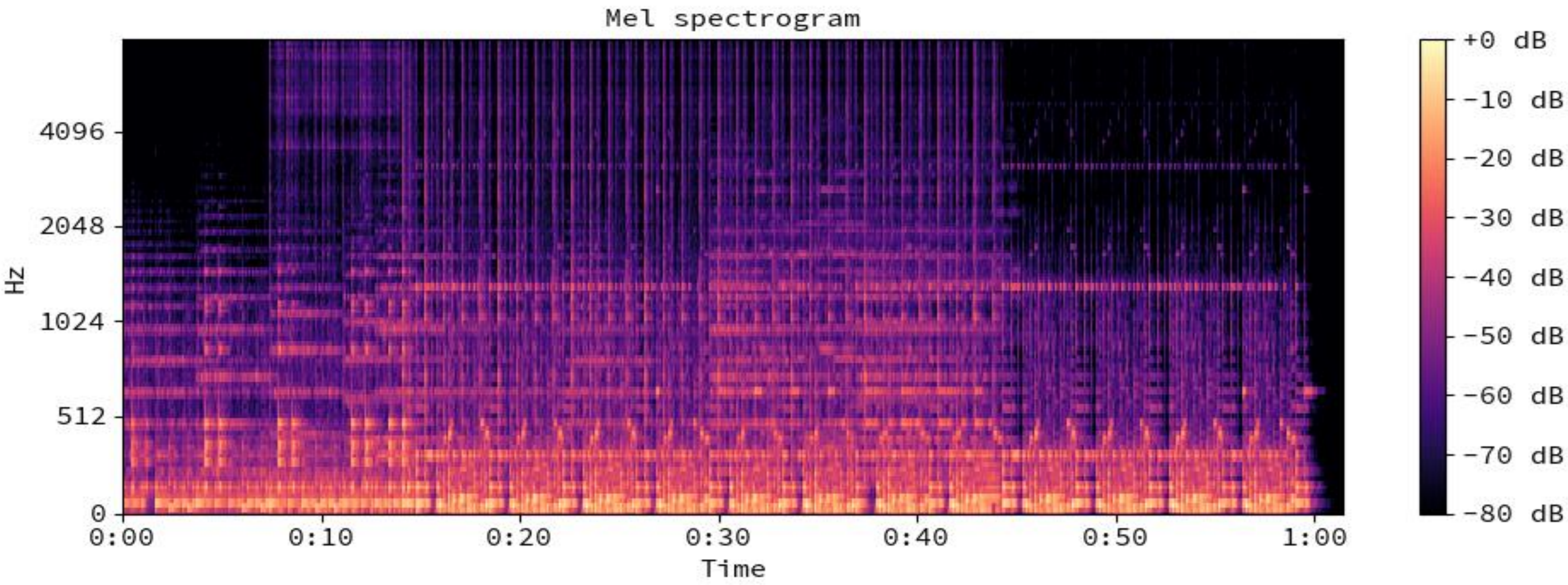


Performance on VoxCeleb



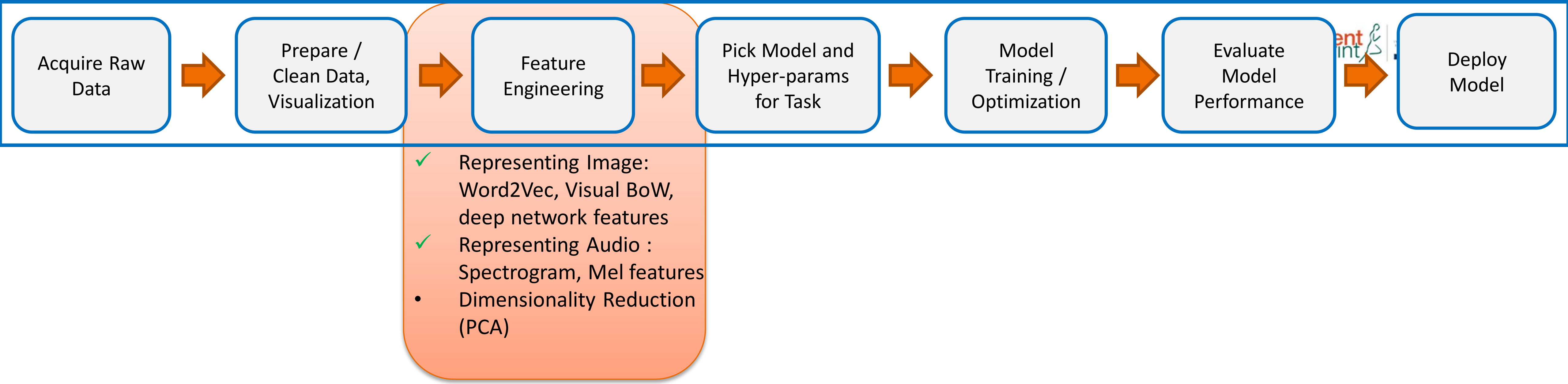
Accuracy	Top-1 (%)	Top-5 (%)
I-vectors + SVM	49.0	56.6
I-vectors + PLDA + SVM	60.8	75.6
CNN-fc-3s no var. norm.	63.5	80.3
CNN-fc-3s	72.4	87.4
CNN	**80.5**	**92.1**

Features from Mel Spectrogram



MFCC
(Hand coded Classic Features)

http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/



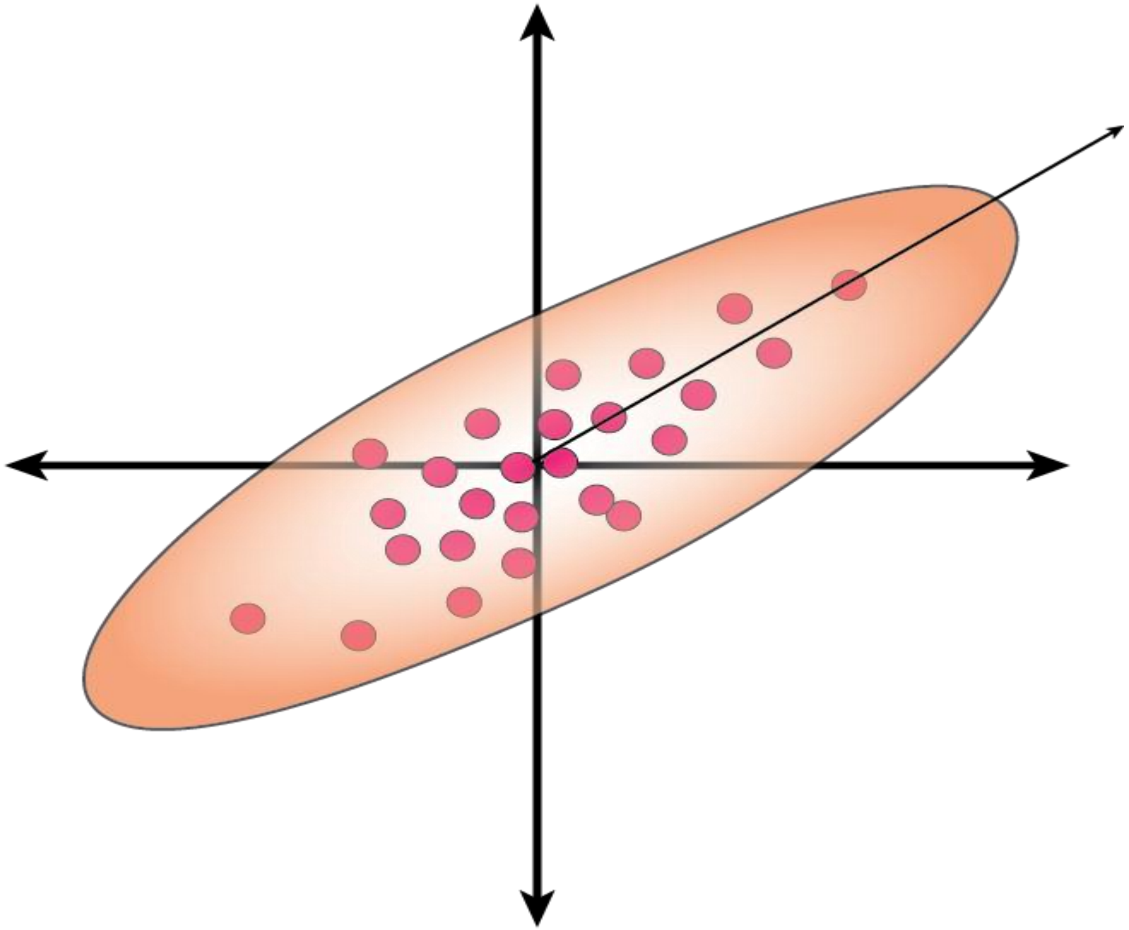
Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- ✓ Representing Image: Word2Vec, Visual BoW, deep network features
- ✓ Representing Audio : Spectrogram, Mel features
- Dimensionality Reduction (PCA)

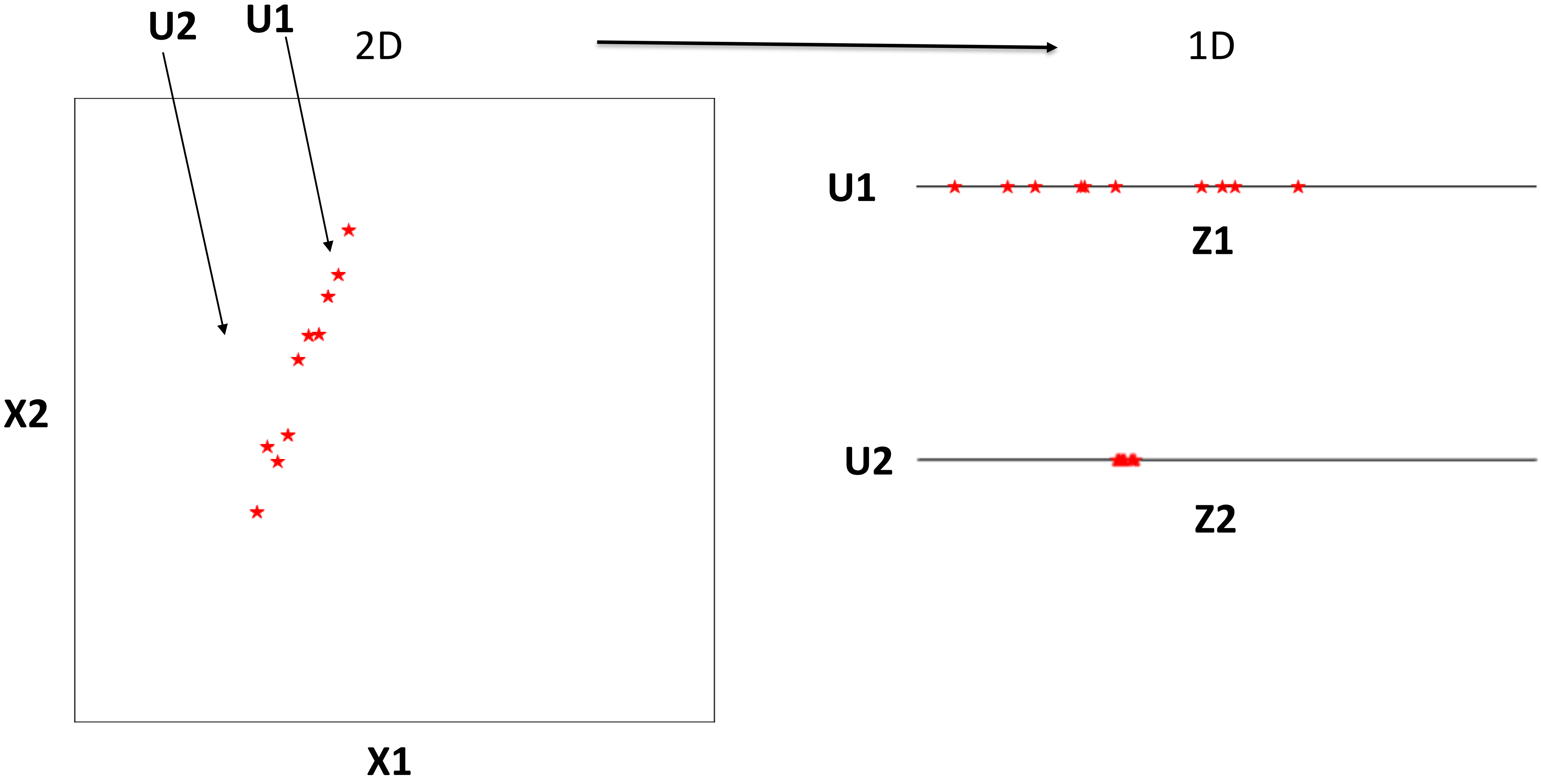
Principal Component Analysis

Simplifying Representations

PCA



Dimensionality and Representation



PCA

$$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\widehat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N} (x_i - \widehat{\mu})(x_i - \widehat{\mu})^T$$

$$\begin{bmatrix} V_a & C_{a,b} & C_{a,c} & C_{a,d} & C_{a,e} \\ C_{a,b} & V_b & C_{b,c} & C_{b,d} & C_{b,e} \\ C_{a,c} & C_{b,c} & V_c & C_{c,d} & C_{c,e} \\ C_{a,d} & C_{b,d} & C_{c,d} & V_d & C_{d,e} \\ C_{a,e} & C_{b,e} & C_{c,e} & C_{d,e} & V_e \end{bmatrix}$$

Review:Eigenvector and Eigenvalue

$$Ax = \lambda x$$

A: Square Matrix

λ: Eigenvalue or characteristic value

x: Eigenvector or characteristic vector

- *A is d X d*
- *x is d x 1*
- *Lambda is scalar*
- *Max of d nonzero lambda*
- *Min (N,d)*

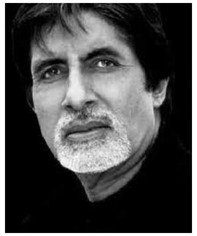
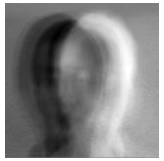
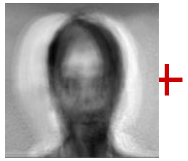
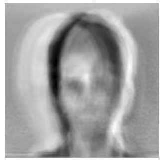
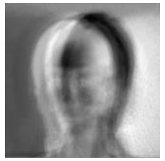
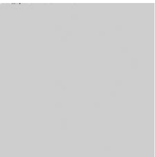
$$\begin{bmatrix} 3 & 4 & -2 \\ 1 & 4 & -1 \\ 2 & 6 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 6 \end{bmatrix}$$

PCA based Feature Extraction

$$\underset{Z}{\overset{r \times 1}{\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ . \\ . \\ z_r \end{bmatrix}}} = \underset{U}{\overset{r \times d}{\begin{bmatrix} u_{11} & u_{12} & u_{13} & . & . & . & . & . & u_{1d} \\ u_{21} & u_{22} & u_{23} & . & . & . & . & . & u_{2d} \\ u_{31} & u_{32} & u_{33} & . & . & . & . & . & u_{3d} \\ . & & & & & & & & \\ u_{r1} & u_{r2} & u_{r3} & . & . & . & . & . & u_{rd} \end{bmatrix}}} \underset{X}{\overset{d \times 1}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ . \\ . \\ . \\ . \\ x_d \end{bmatrix}}}$$

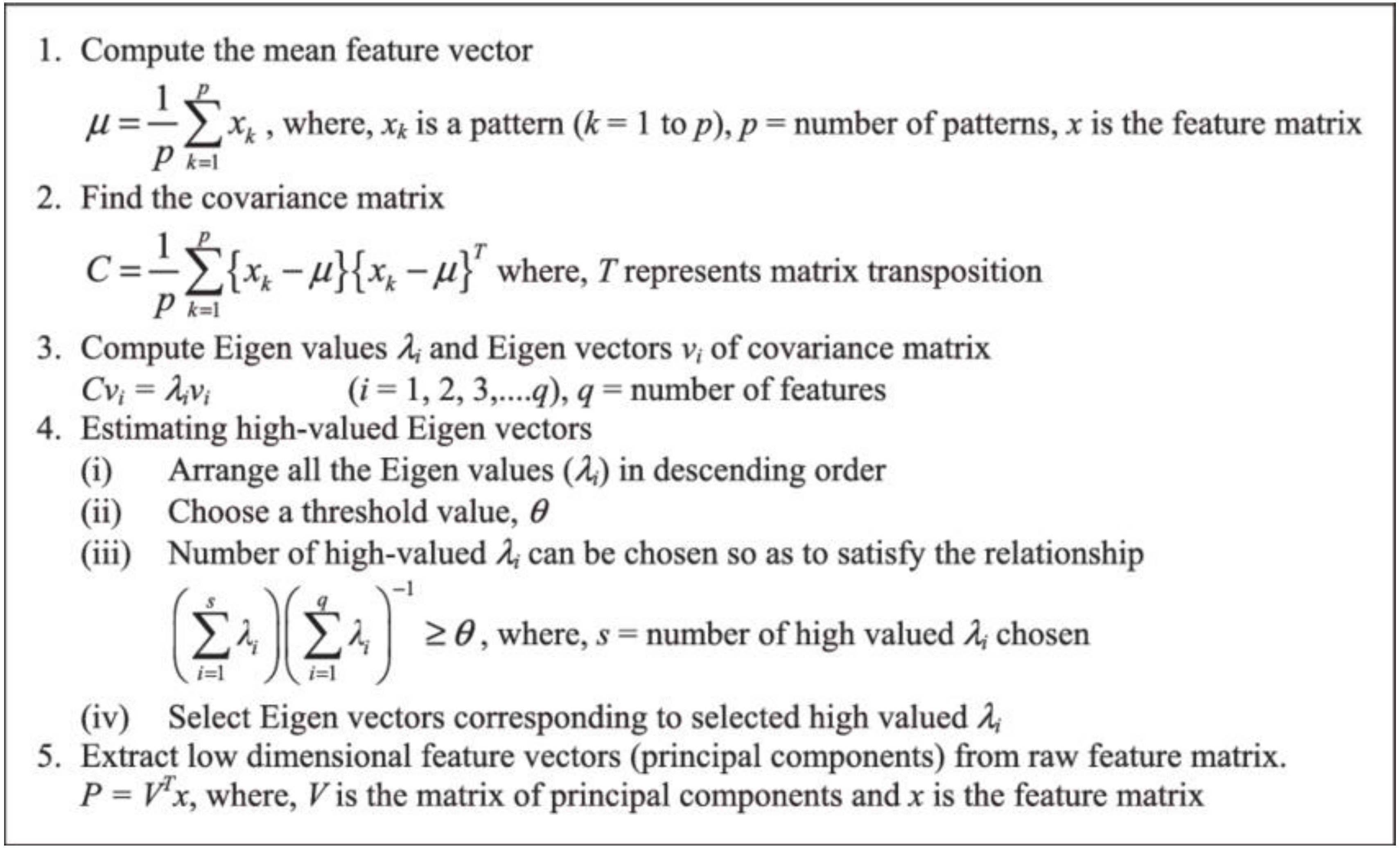
Each row in U is an Eigen vector of co-variance Matrix

Representing with EigenFaces

 = 6.8 *  + 3.5 *  + 2.89 *  -1.2 *  ……..+0.0002 * 

Any face in the database can be represented as a linear combination of the eigen faces.

PCA/Eigen Face Algorithm: Detail



1. Compute the mean feature vector

   $$\mu = \frac{1}{p}\sum_{k=1}^{p} x_k$$, where, x_k is a pattern ($k = 1$ to p), p = number of patterns, x is the feature matrix

2. Find the covariance matrix

   $$C = \frac{1}{p}\sum_{k=1}^{p}\{x_k - \mu\}\{x_k - \mu\}^T$$ where, T represents matrix transposition

3. Compute Eigen values λ_i and Eigen vectors v_i of covariance matrix

   $Cv_i = \lambda_i v_i$ (i = 1, 2, 3,....q), q = number of features

4. Estimating high-valued Eigen vectors

   (i) Arrange all the Eigen values (λ_i) in descending order

   (ii) Choose a threshold value, θ

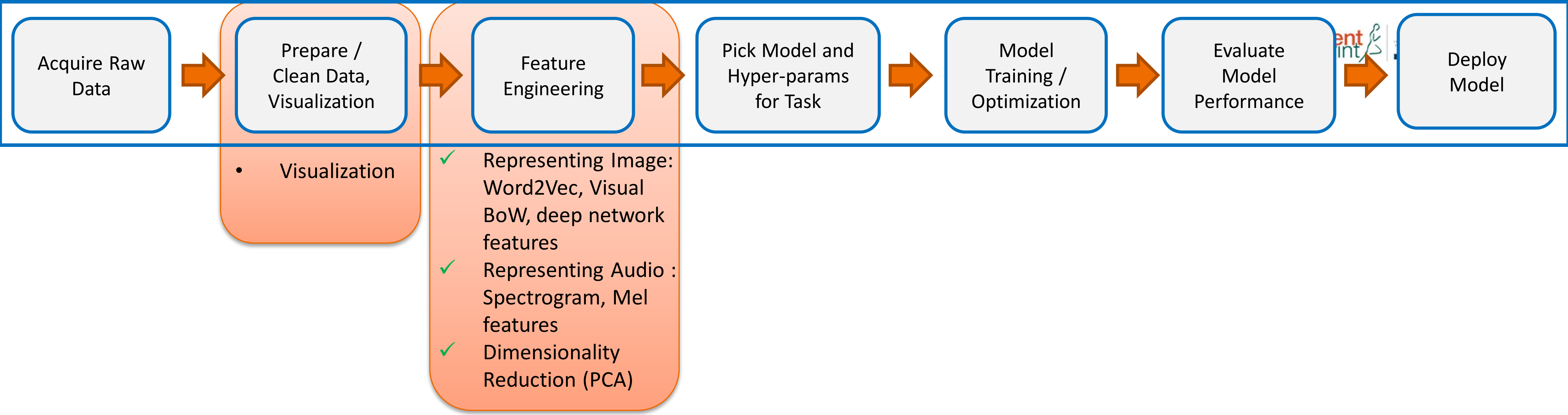
   (iii) Number of high-valued λ_i can be chosen so as to satisfy the relationship

   $$\left(\sum_{i=1}^{s}\lambda_i\right)\left(\sum_{i=1}^{q}\lambda_i\right)^{-1} \geq \theta$$, where, s = number of high valued λ_i chosen

   (iv) Select Eigen vectors corresponding to selected high valued λ_i

5. Extract low dimensional feature vectors (principal components) from raw feature matrix.

   $P = V^T x$, where, V is the matrix of principal components and x is the feature matrix



Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- Visualization

✓ Representing Image: Word2Vec, Visual BoW, deep network features

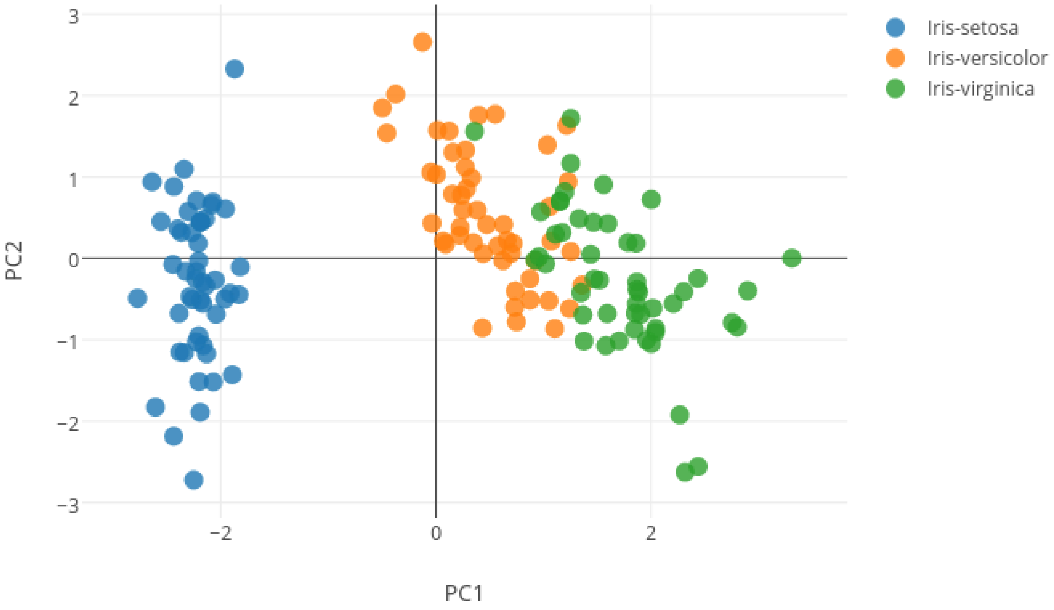
✓ Representing Audio : Spectrogram, Mel features

✓ Dimensionality Reduction (PCA)

Data Visualization

When Data is High-Dimensional

PCA



PLOT ON 2 PRINCIPLE COMPONENTS

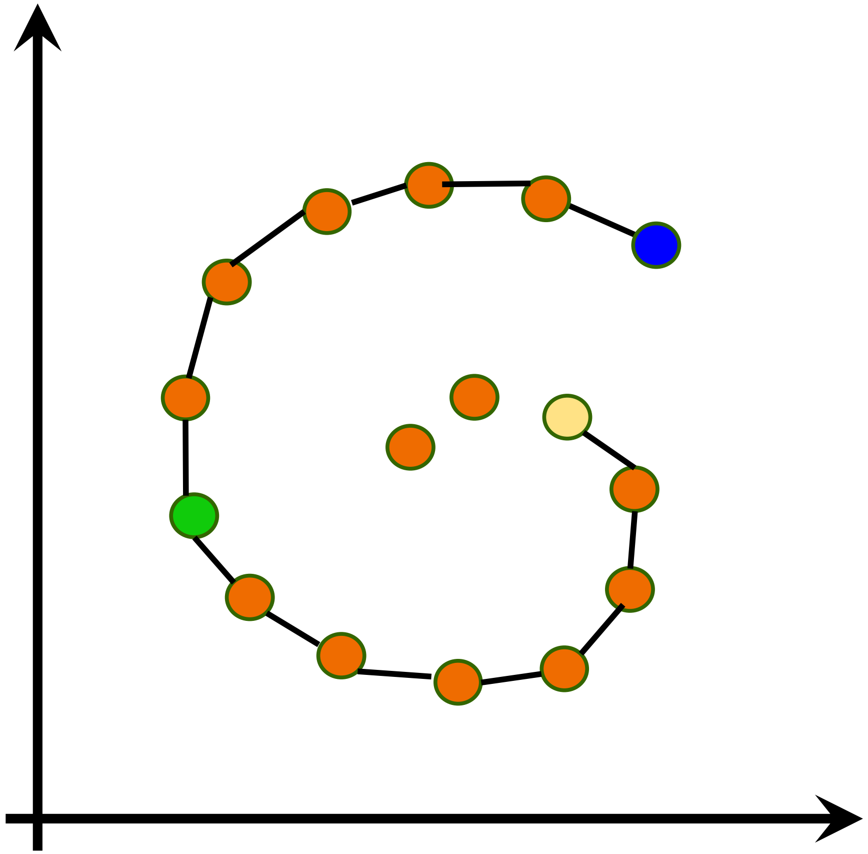
MDS (Multidimensional scaling)

- Minimize an objective function that measures the discrepancy between similarities in the data and similarities in the map.

- Distance between samples in “high” dimension and “low’’ dimension is same (or D-d) is minimized.

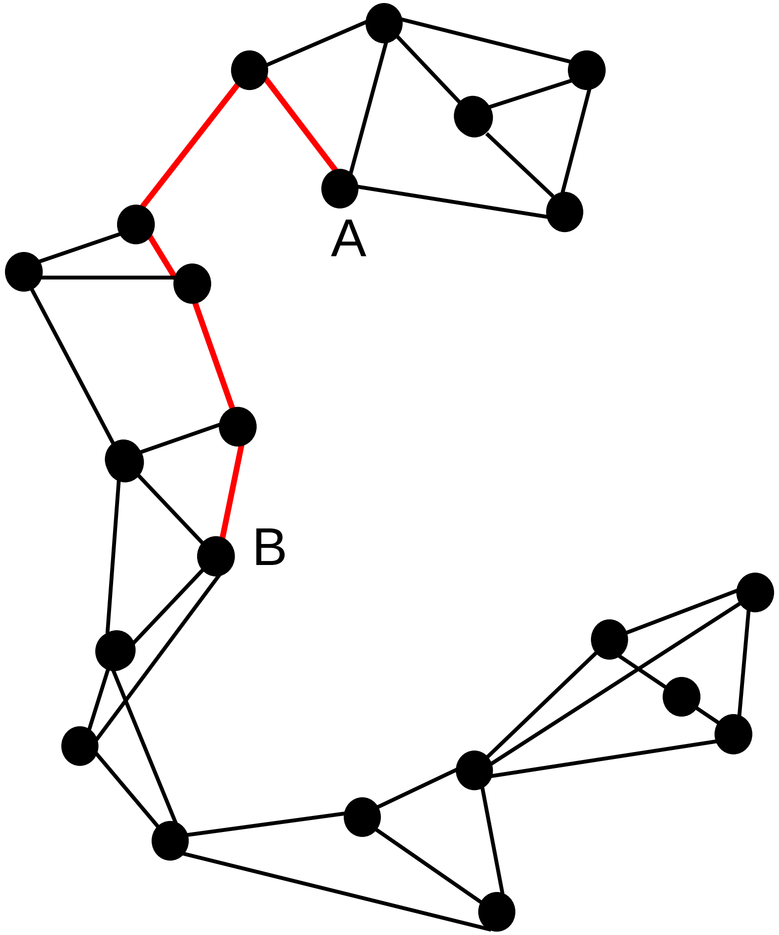
ISOMAP (Isometric Mapping)

- d(🔵,🟡) > d(🔵,🟢)
- Is Euclidean metric the right distance metric?
- How to robustly measure distances along the manifold?



ISOMAP

- How does ISOMAP measure the MD?
- Connect each data point to its K nearest neighbors in the high-dimensional space.
- Link weights: True Euclidean distances.
- $MD(A, B) = ShortestPath(A, B)$ in this ***neighborhood graph***.
- Compute the low-dimensional embedding as in Metric MDS.



LLE: Locally Linear Embedding

- Idea: Preserve the structure of local neighbourhood

$$\mathbf{x}_i \approx \sum_j w_{ij}\mathbf{x}_j$$

- Approach:
  - —Represent each point as a weighted combination of its Neighbours in HD. Remember the $w_{ij}s$.
  - —Find a LD representation that minimize the representation error:

$$Cost = \sum_i \| \mathbf{y}_i - \sum_{j \varepsilon N(i)} w_{ij}\mathbf{y}_j \|^2$$

- – The weights w_{ij} refer to the amount of contribution the point x_i has while reconstructing the point x_i. The cost function is minimized under two constraints: (a) Each data point x_i is reconstructed only from its neighbors, thus enforcing w_{ij} to be zero if point x_j is not a neighbor of the point x_i and (b) The sum of every row of the weight matrix equals 1.
- – Also ys should have unit variance across each dimension.

SNE and t-SNE

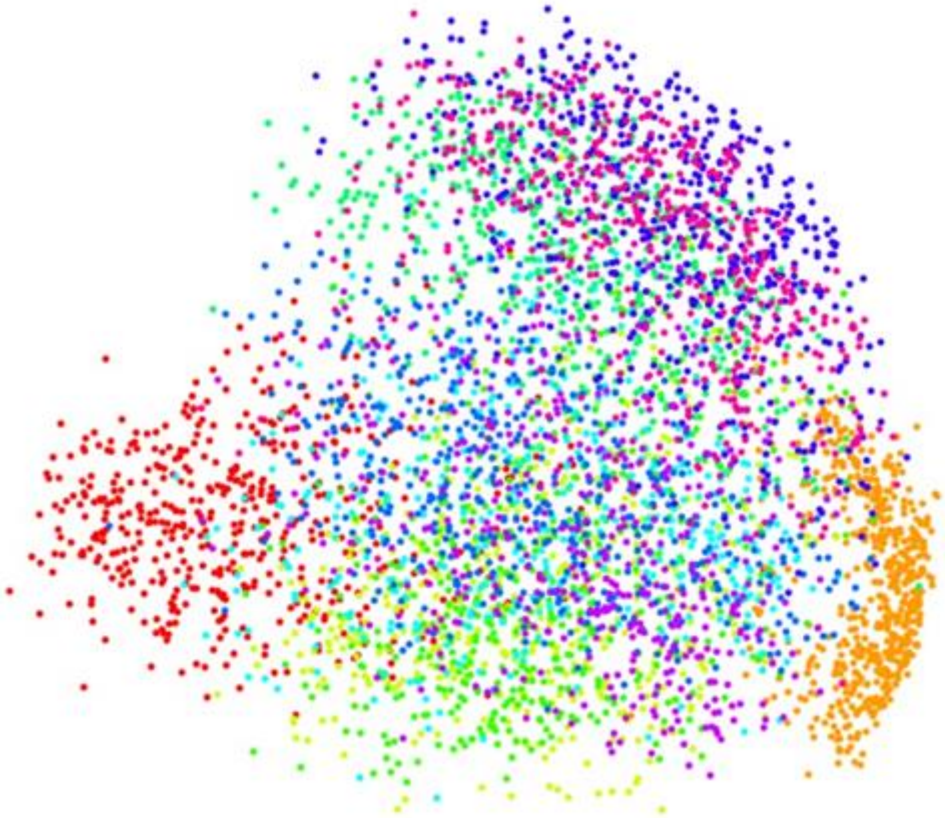
- Idea is simple: Instead of distance think about probabilities. P_{ij} as the probability of j in the neighborhood of i.
- For each point, we have now a probability vector (of size N).
  - SNE uses Gaussian. T-SNE uses another t-distribution (with 1 degree of freedom).
- We want these prob vectors to be the same in low dimensional.
- Optimize using gradient descent.

Computing the LD Embedding

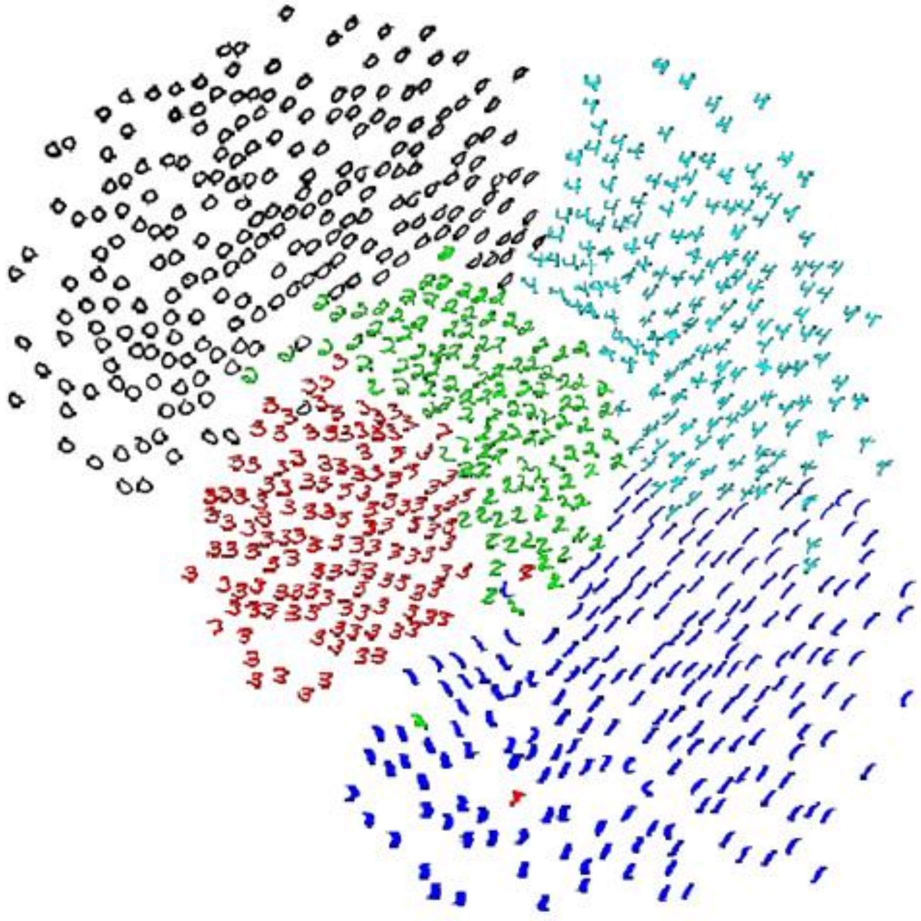
$$Cost = \sum_i KL(P_j \parallel Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- For points where p_{ij} is large and q_{ij} is small we lose a lot.
  - – Nearby points in high-D really want to be nearby in low-D

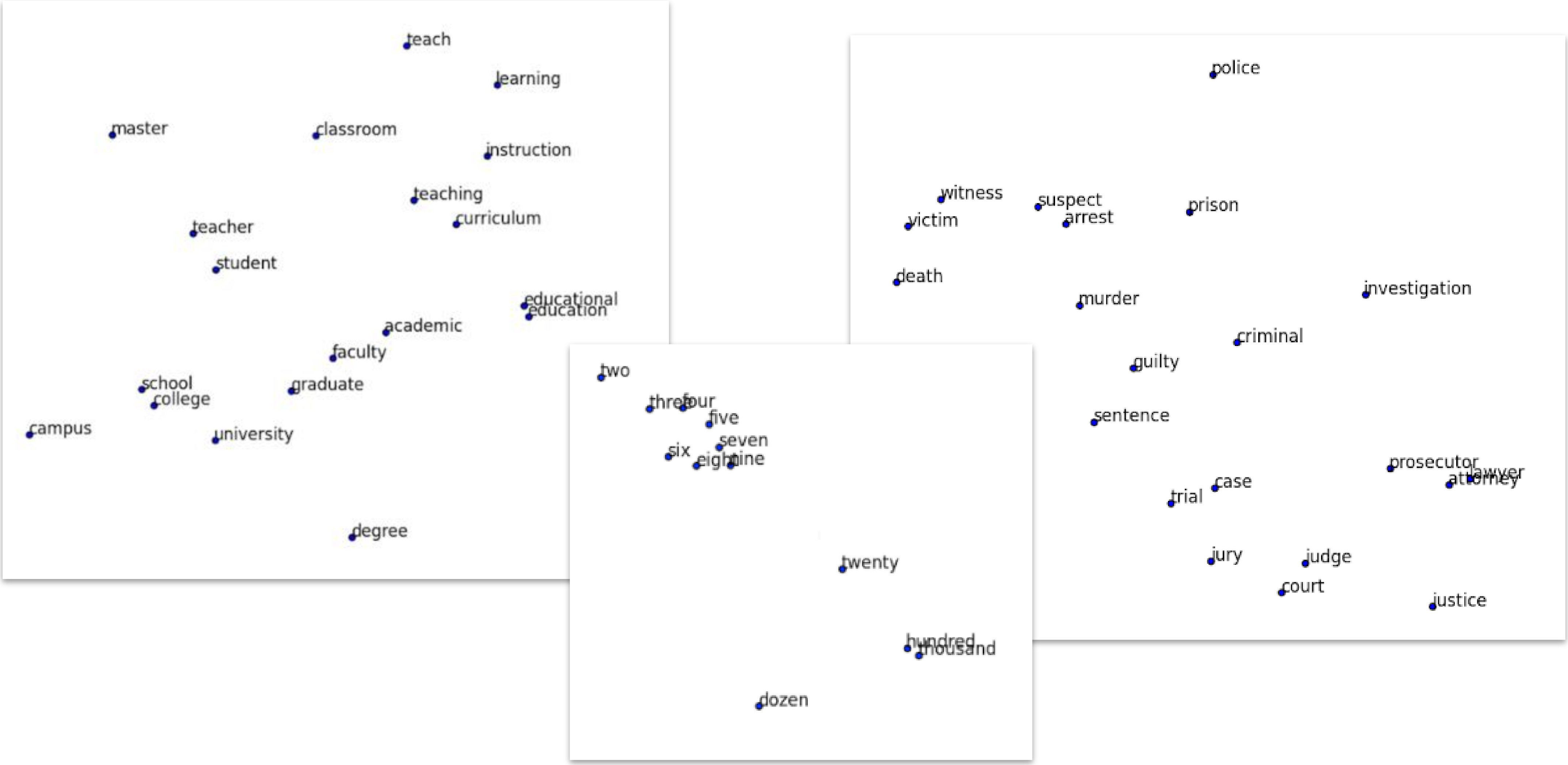
PCA on MNIST *(0-9)*



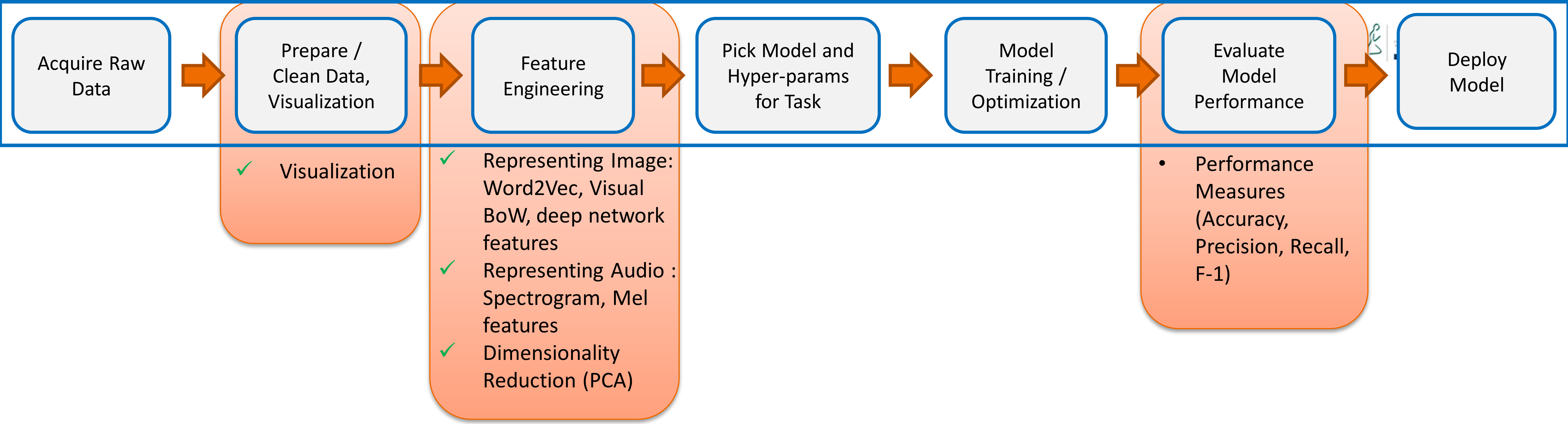
SNE on MNIST *(0-5)*



Word2vec



http://nlp.yvespeirsman.be/blog/visualizing-word-embeddings-with-tsne/



Acquire Raw Data

Prepare / Clean Data, Visualization

Feature Engineering

Pick Model and Hyper-params for Task

Model Training / Optimization

Evaluate Model Performance

Deploy Model

- ✓ Visualization

- ✓ Representing Image: Word2Vec, Visual BoW, deep network features
- ✓ Representing Audio : Spectrogram, Mel features
- ✓ Dimensionality Reduction (PCA)

- Performance Measures (Accuracy, Precision, Recall, F-1)

Performance Metrics

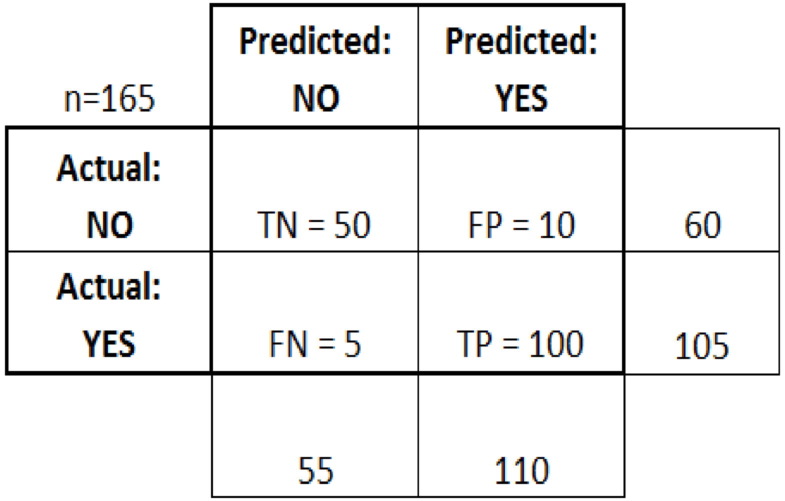
Key accuracy measures and terminologies

- Classification Error = $\frac{errors}{total}$

$$= \frac{FP + FN}{TP + TN + FP + FN}$$

- Accuracy = 1 - Error = $\frac{correct}{total}$

$$= \frac{TP + TN}{TP + TN + FP + FN}$$



n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Revisiting scenarios where metrics are appropriate

- When you do cancer screening what do you care?
  - High TP and Low FN

- When you classify between “apple” and “orange”
  - High Accuracy

- Automatic Firing on detecting a violation.
  - Very low FP

Precision and Recall

$$Precision = \frac{TP}{(TP + FP)}$$

$$Recall = \frac{TP}{(TP + FN)}$$

Precision and Recall – examples

- A system which needs to launch a missile at a terrorist hideout located in a dense urban area.
  - Precision not 100% ➡ civilian casualties
- A system which needs to identify cancer-risk patients
  - Recall not 100% ➡ some patients will die of cancer

F-measure: Combines Precision and Recall

- What to do when one classifier has better Precision but worse Recall, while other classifier behaves exactly opposite?
  - F-measure (Information Retrieval)

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

F-measure

- What to do when one classifier has better Precision but worse Recall, while other classifier behaves exactly opposite?
  - F-measure (Information Retrieval)

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

$$Precision = \frac{TP}{(TP + FP)}$$

$$Recall = \frac{TP}{(TP + FN)}$$

- F1 measure punishes extreme values more !
- Definition of Recall and Precision have same numerator, different denominators. A sensible way to combine them is harmonic mean.

F-measure

- Use when
  - FP and FN are ‘equally costly’
  - You don’t expect results to change when more data is added
  - TN is high (e.g. face detector)

$$Precision = \frac{TP}{(TP+FP)}$$

$$Recall = \frac{TP}{(TP+FN)}$$

$$\frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

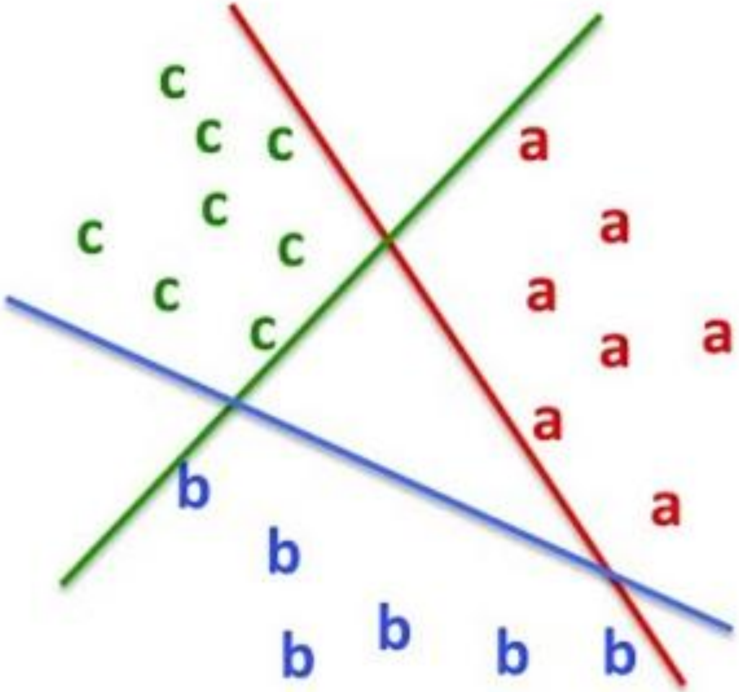
Utility and Cost

- Sometimes, there is a cost for each error
  - E.g. Earthquake prediction
    - False positive: Cost of preventive measures
    - False negative: Cost of recovery

- Detection Cost (Event detection) -Can be applied to example above
  - Cost = C_{FP} * FP + C_{FN} * FN

How to use 2-class measures for multi-class ?

- Convert into 2-class problem(s) !

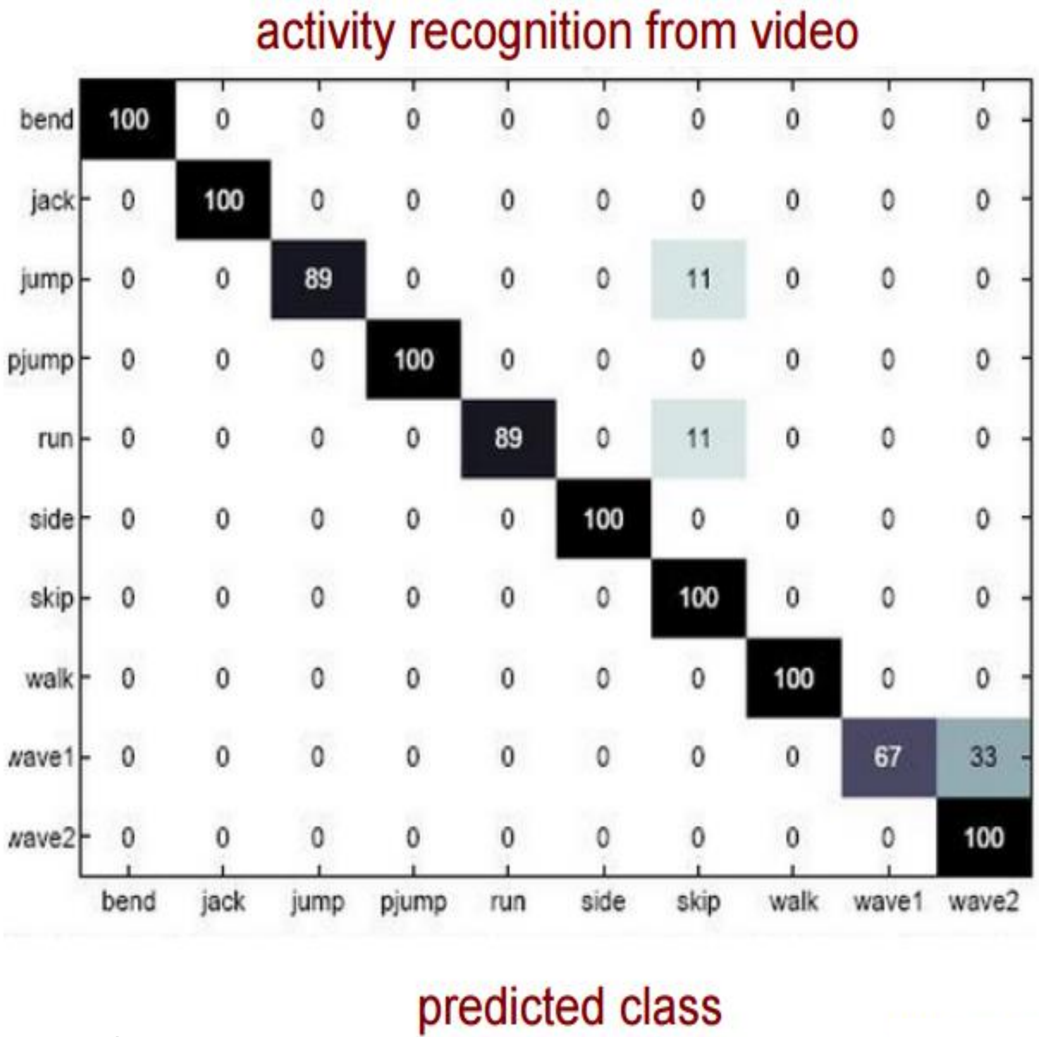


Multi-class problems - Confusion matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Avg. accuracy may not be very meaningful with imbalanced class label distribution

actual class

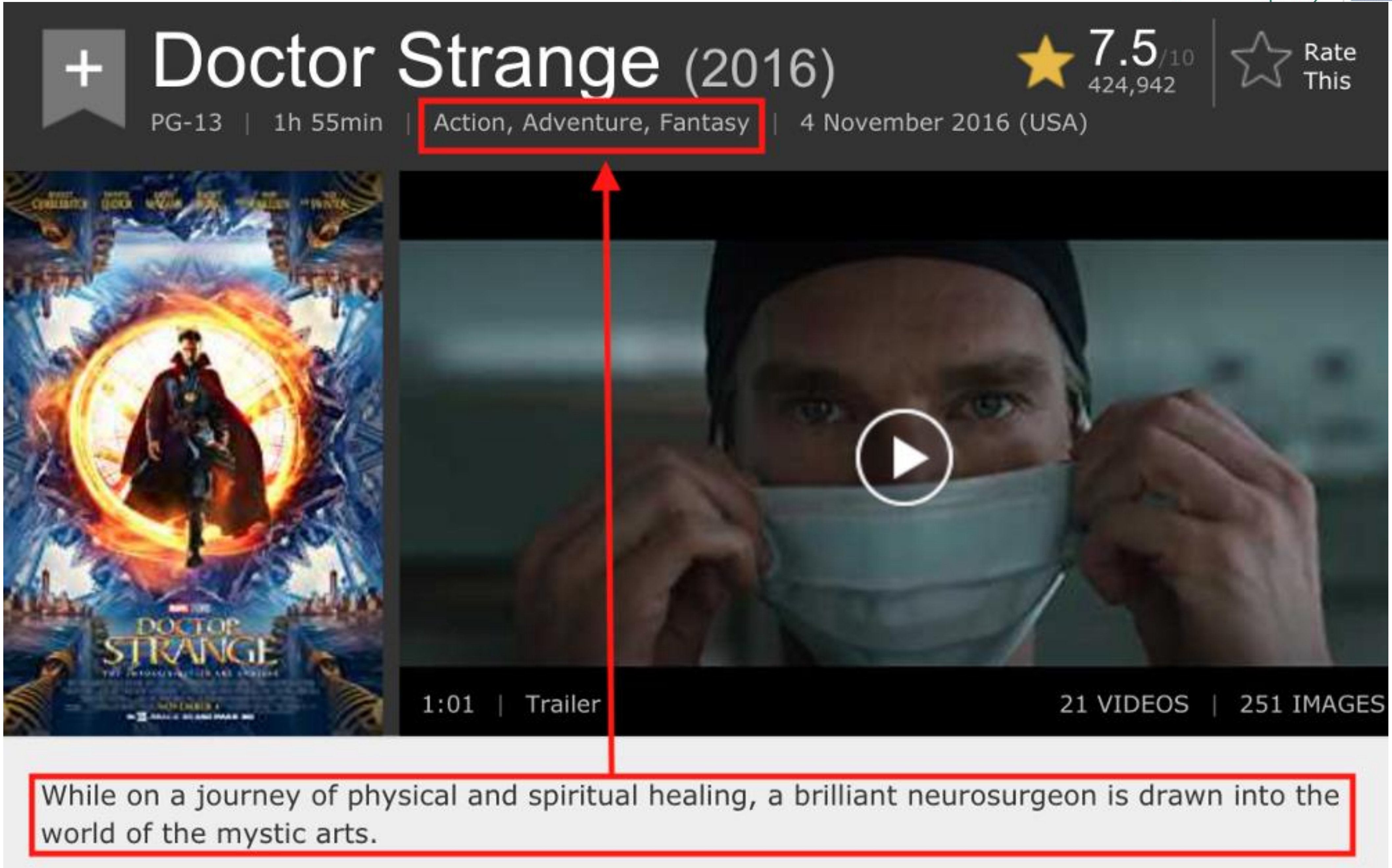


activity recognition from video

	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2
bend	100	0	0	0	0	0	0	0	0	0
jack	0	100	0	0	0	0	0	0	0	0
jump	0	0	89	0	0	0	11	0	0	0
pjump	0	0	0	100	0	0	0	0	0	0
run	0	0	0	0	89	0	11	0	0	0
side	0	0	0	0	0	100	0	0	0	0
skip	0	0	0	0	0	0	100	0	0	0
walk	0	0	0	0	0	0	0	100	0	0
wave1	0	0	0	0	0	0	0	0	67	33
wave2	0	0	0	0	0	0	0	0	0	100

predicted class

Courtesy:
vision.jhu.edu



+ Doctor Strange (2016)

7.5/10 424,942

Rate This

PG-13 | 1h 55min | Action, Adventure, Fantasy | 4 November 2016 (USA)

1:01 | Trailer

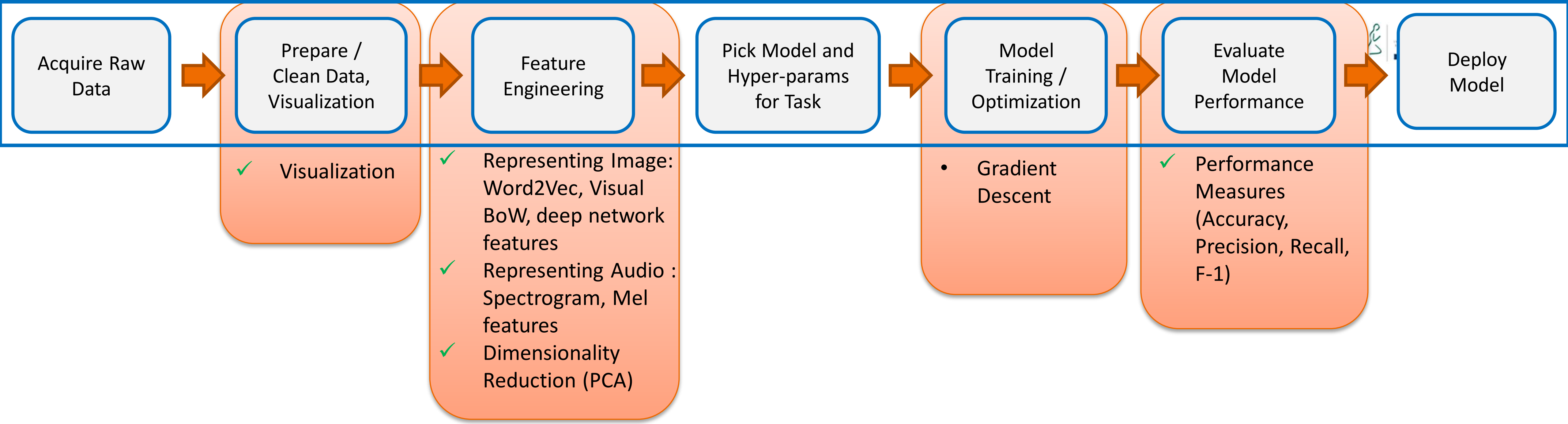
21 VIDEOS | 251 IMAGES

While on a journey of physical and spiritual healing, a brilliant neurosurgeon is drawn into the world of the mystic arts.

Two Metrics for Multi-label case

- Jaccard Distance: 1-intersection/union

- Hamming Loss: mismatches/total



Acquire Raw Data

Prepare / Clean Data, Visualization

- ✓ Visualization

Feature Engineering

- ✓ Representing Image: Word2Vec, Visual BoW, deep network features
- ✓ Representing Audio : Spectrogram, Mel features
- ✓ Dimensionality Reduction (PCA)

Pick Model and Hyper-params for Task

Model Training / Optimization

- Gradient Descent

Evaluate Model Performance

- ✓ Performance Measures (Accuracy, Precision, Recall, F-1)

Deploy Model

Gradient Descent

The Problem

- Find $\boldsymbol{w}$, given examples: $(x_i, y_i), i = 1,2, \ldots, n$
- Supervised situation: output label y; is available for all training n samples
- **Objective:** predict y values for a novel input x that we have not seen before
  - Called generalization in Machine Learning
- How do we find w? **Ans: Gradient descent!**

Loss Function

- **Error/Loss (L):** A function of the difference between the actual value or label (y_i) and the predicted value ($f(w, x_i)$)

- **Total Loss:**

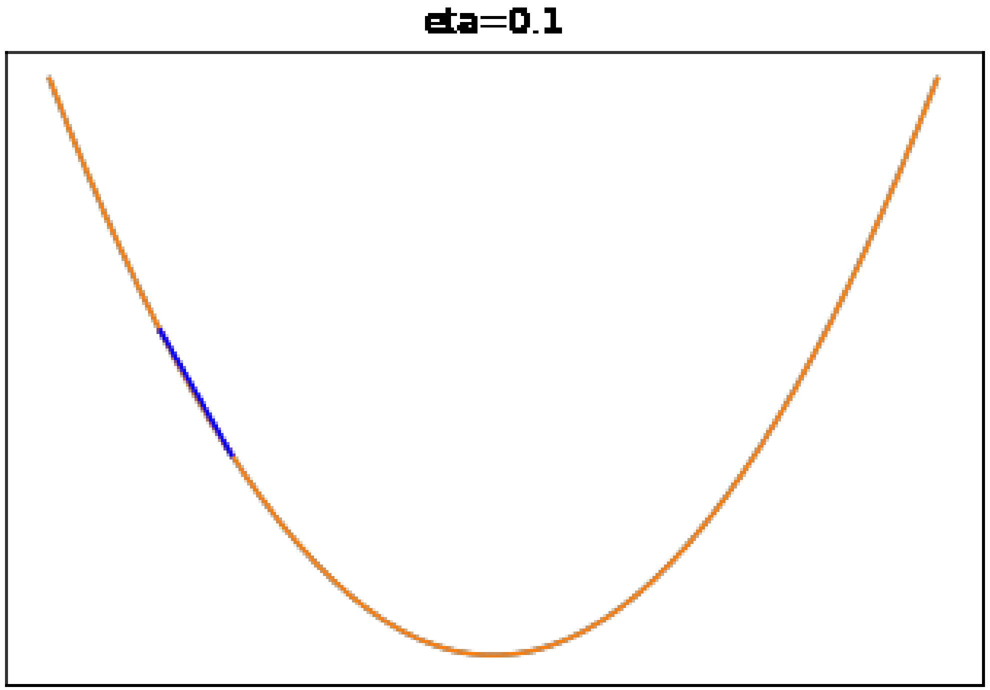
$$J = \sum_{i=1}^{n} L(y_i, f(w, x_i))$$

the sum of loss over n labelled training samples: (x_i, y_i)

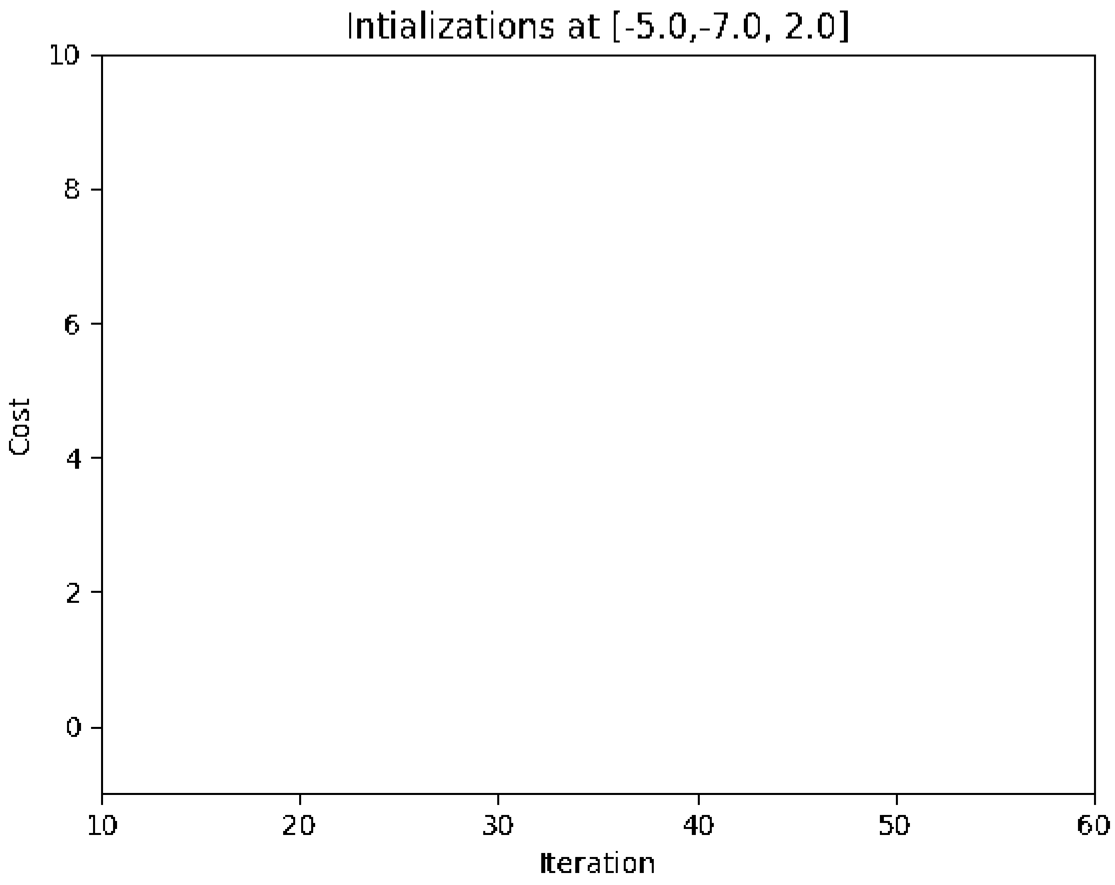
- **Strategy:** Start with some initial values for w and bring the predicted values closer to the corresponding labels (or minimize J) by adjusting w.

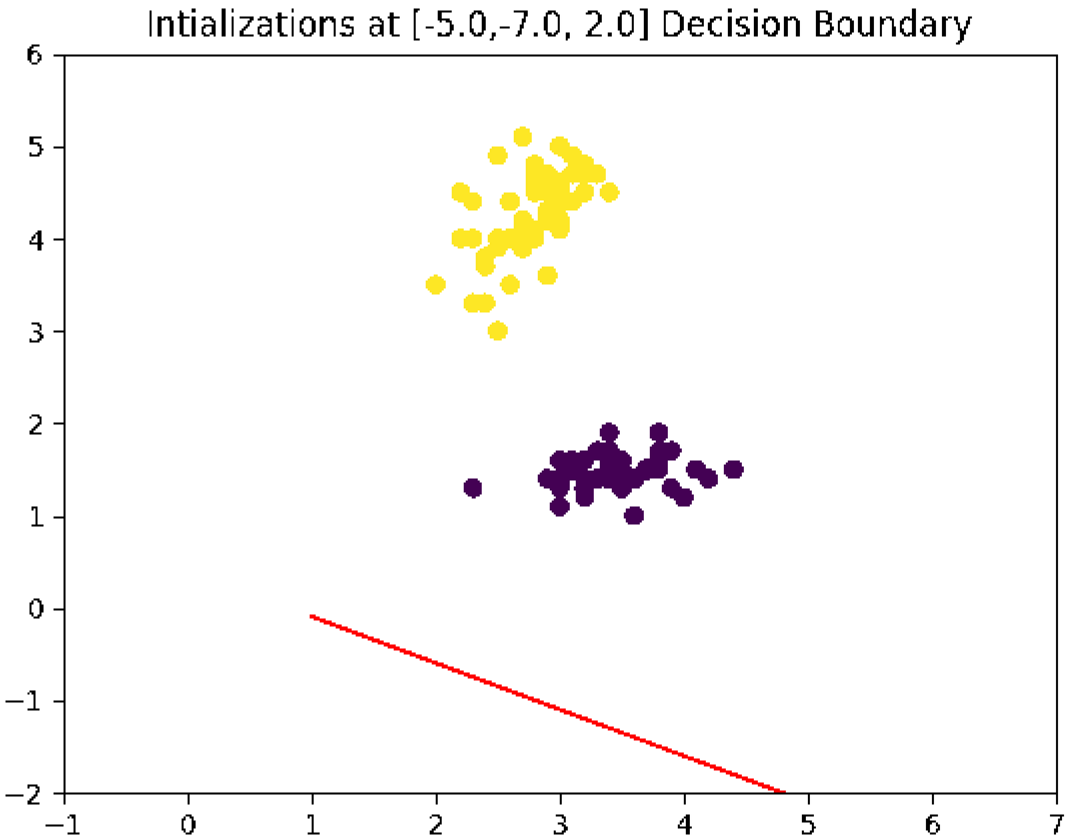
Gradient Descent

- Minimum of the function lies in the opposite direction of gradient
- Start with a guess
- Take a step against gradient:
- $\boldsymbol{w' = w - \eta \nabla J(w)}$

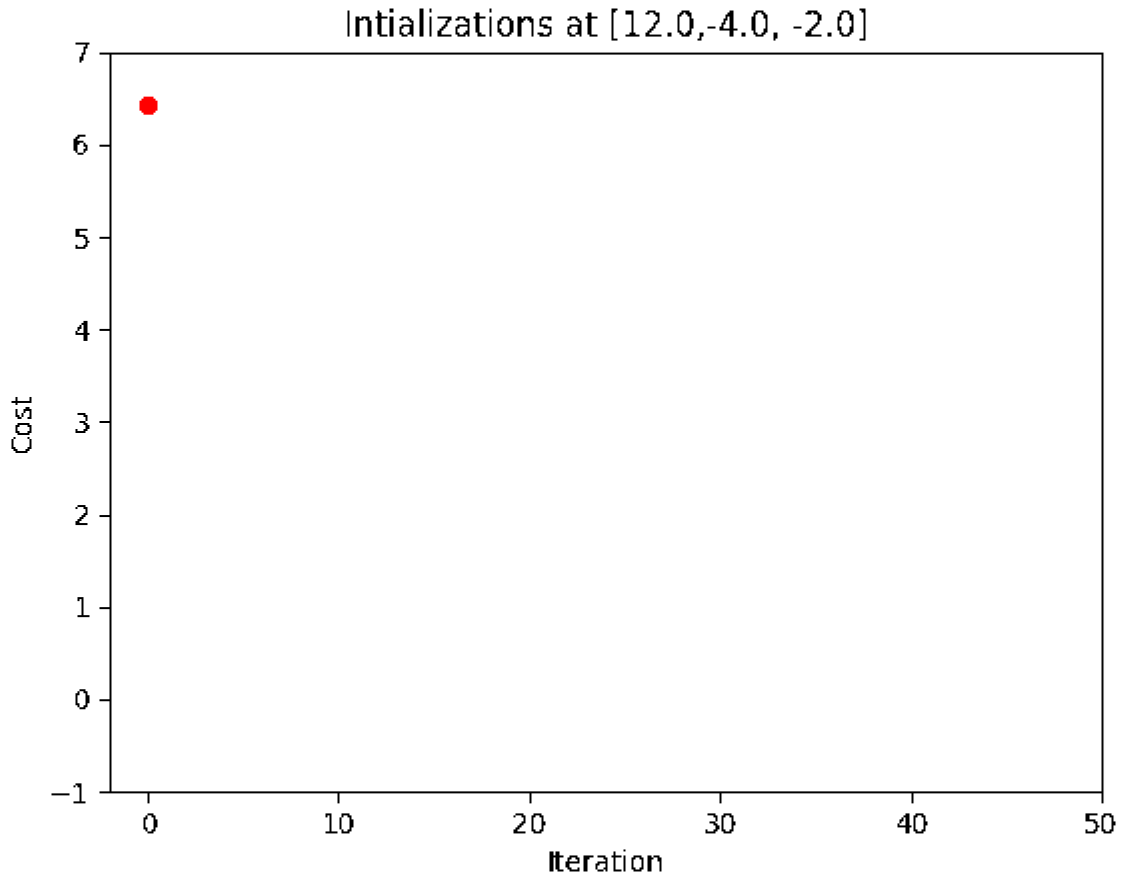


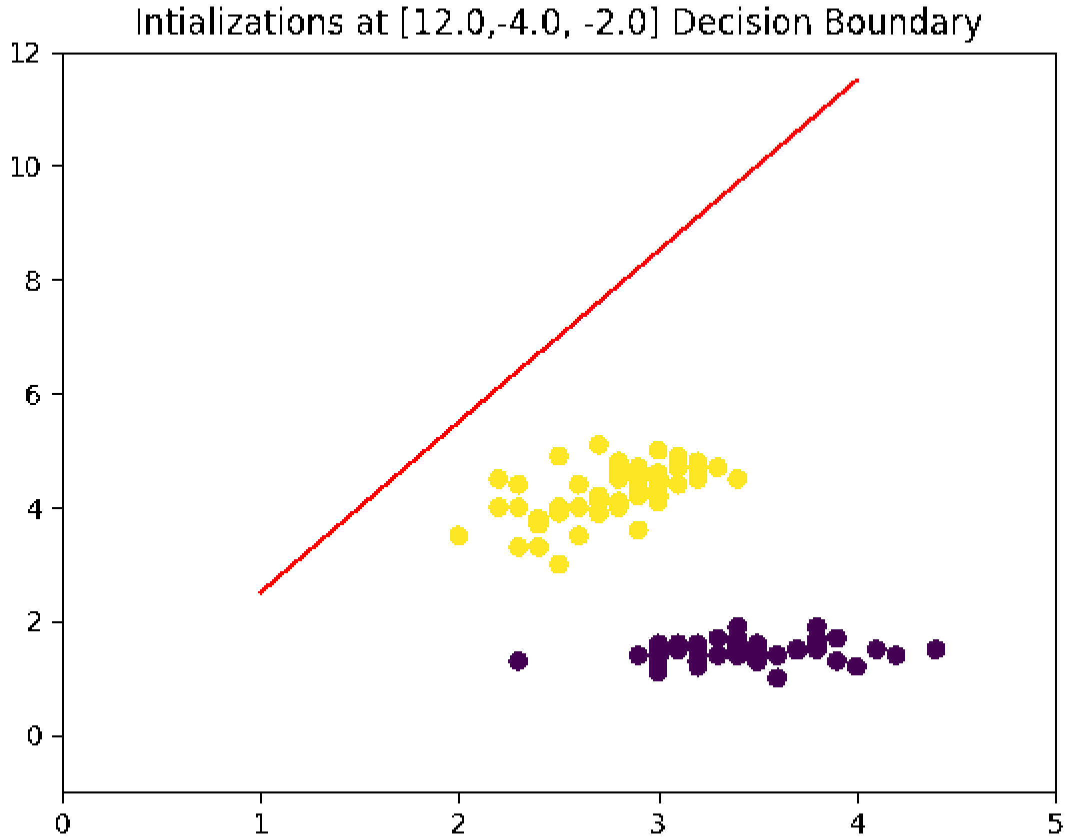
Gradient Descent - Initialization





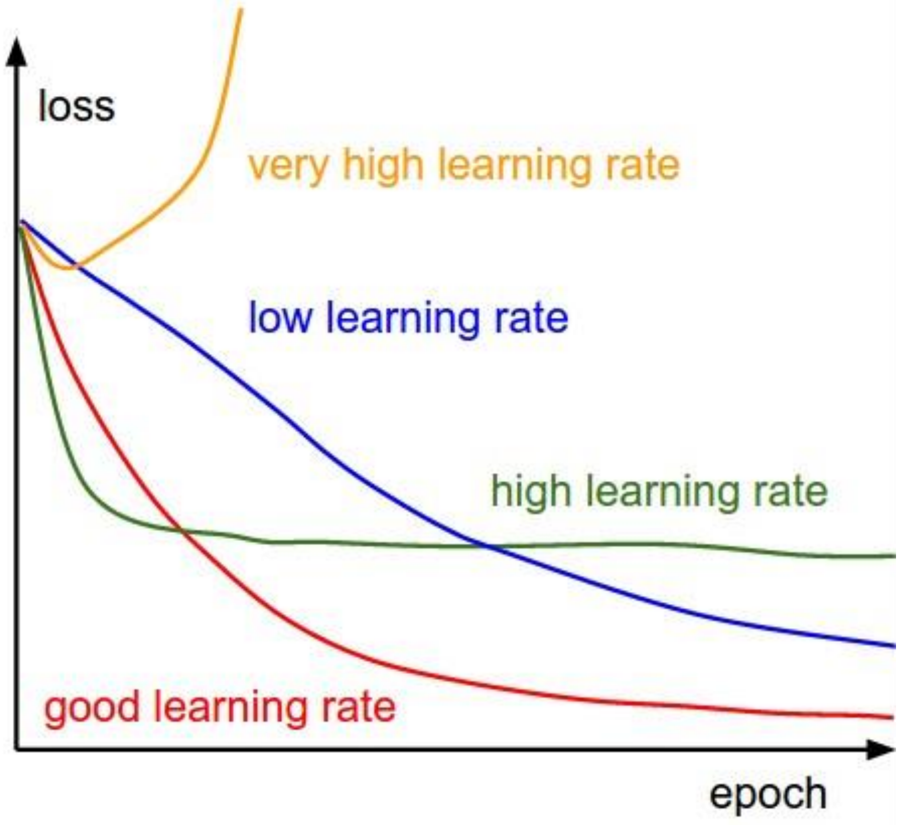
Gradient Descent - Initialization





Scale varied for better visibility

Gradient Descent - Learning rate

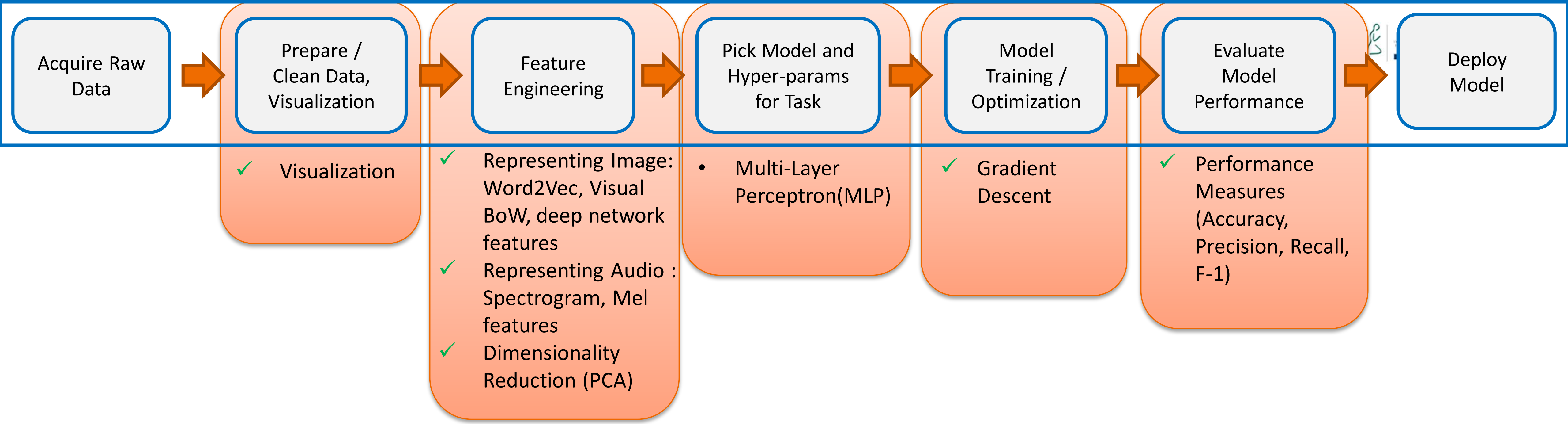


http://cs231n.github.io/neural-networks-3/#baby

Mini Batch Gradient Descent

Pseudo Code

```
model = initialization(...)
train_data = load_training_data()
for i in 1...n:   // epochs
    $Tb_1$, $Tb_2$, $Tb_3$, ……………,$Tb_m$ = split_training_batches(train_data)
    for $Tb_j$ in $Tb_1$, $Tb_2$, $Tb_3$, ……………,$Tb_m$ :     // mini batches
            error = 0
            for X, y in $Tb_j$ :   // samples in mini batch $Tb_j$
                    predictions = predict(X, model)
                    error += calculate_error(y, predictions)
            gradient  = differentiate(model_params, error)
            model = update_model(model, gradient)
```



Acquire Raw Data

Prepare / Clean Data, Visualization
- ✓ Visualization

Feature Engineering
- ✓ Representing Image: Word2Vec, Visual BoW, deep network features
- ✓ Representing Audio : Spectrogram, Mel features
- ✓ Dimensionality Reduction (PCA)

Pick Model and Hyper-params for Task
- Multi-Layer Perceptron(MLP)

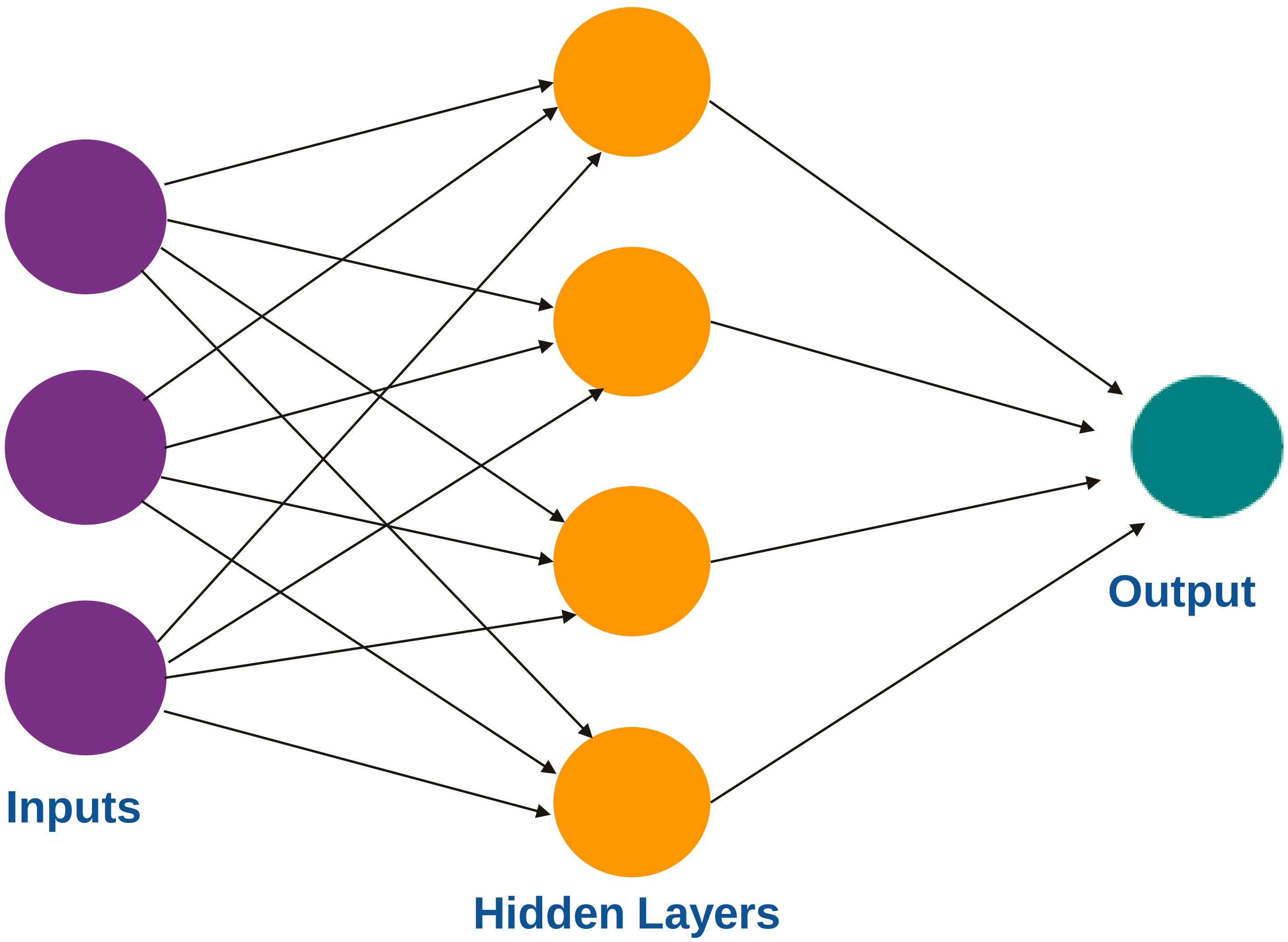
Model Training / Optimization
- ✓ Gradient Descent

Evaluate Model Performance
- ✓ Performance Measures (Accuracy, Precision, Recall, F-1)

Deploy Model

MLP

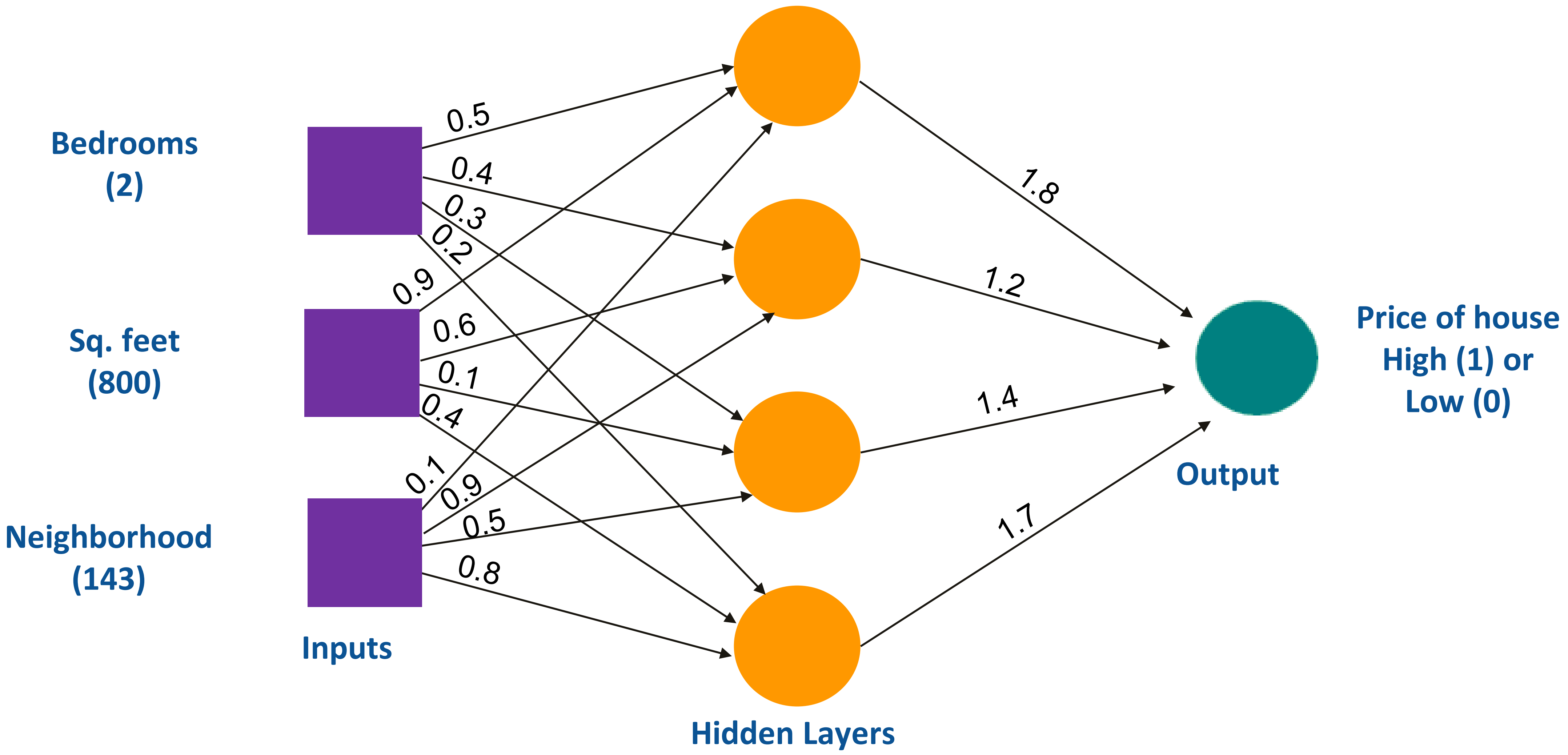
MLP



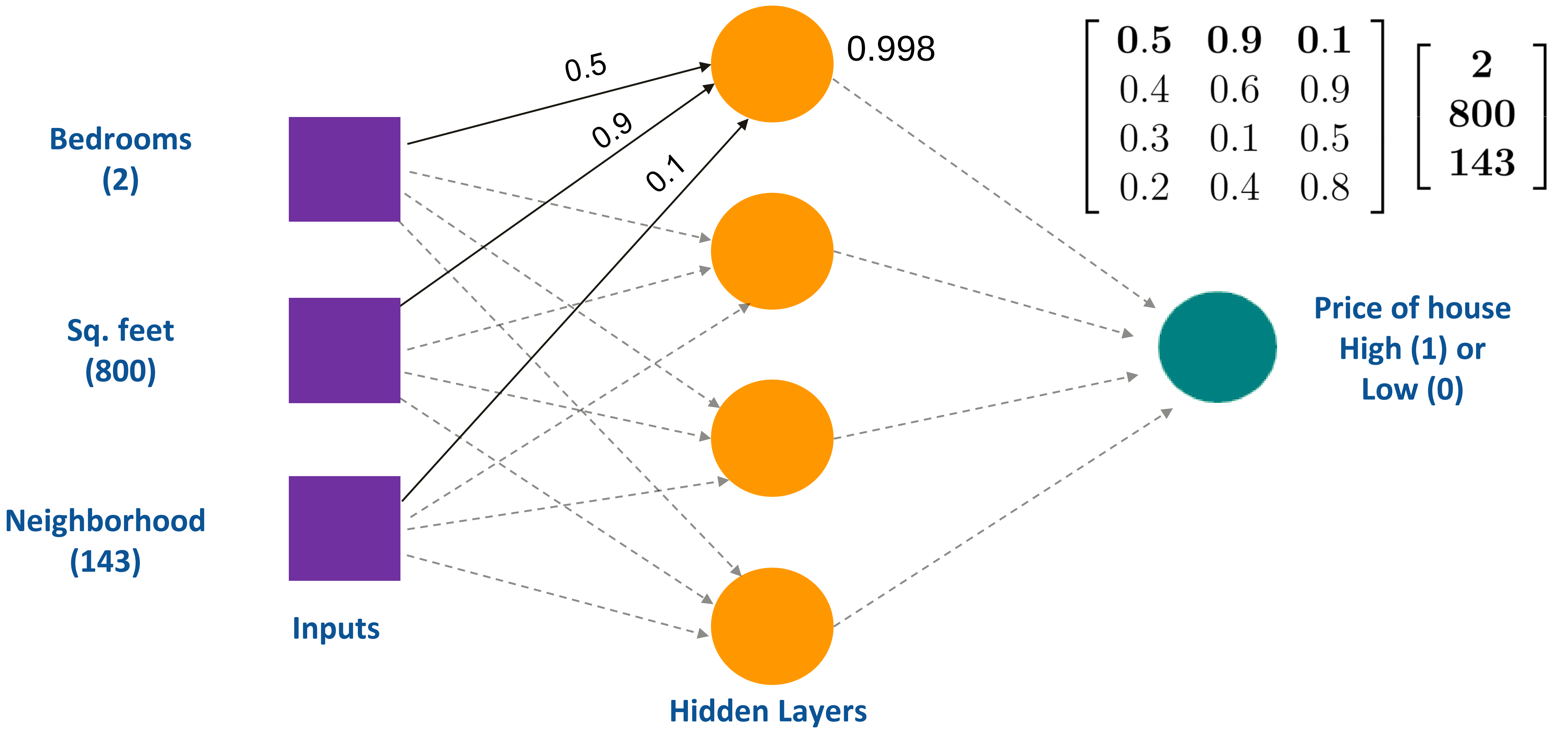
Eg. House price from attributes

Bedrooms	Sq. Feet	Neighborhood (no. of houses in the locality)	Price high or low? High (1), Low (0)
3	2000	90	1
2	800	143	0
2	850	167	0
1	550	267	0
4	2000	396	1

Initialize weights



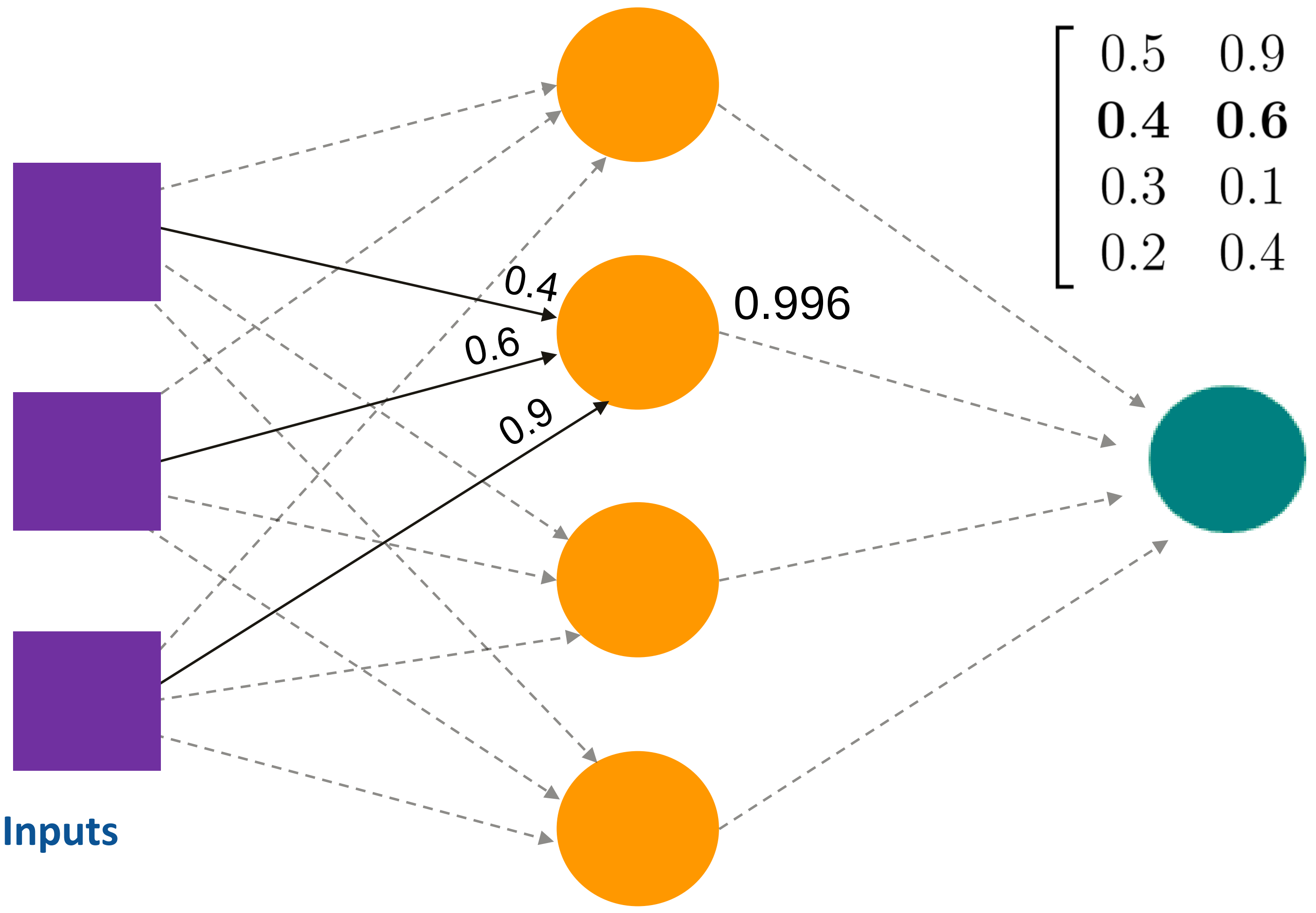
Weights at the first neuron



$$\begin{bmatrix} \mathbf{0.5} & \mathbf{0.9} & \mathbf{0.1} \\ 0.4 & 0.6 & 0.9 \\ 0.3 & 0.1 & 0.5 \\ 0.2 & 0.4 & 0.8 \end{bmatrix} \begin{bmatrix} \mathbf{2} \\ \mathbf{800} \\ \mathbf{143} \end{bmatrix}$$

Weights at the second neuron

$$\begin{bmatrix} 0.5 & 0.9 & 0.1 \\ \mathbf{0.4} & \mathbf{0.6} & \mathbf{0.9} \\ 0.3 & 0.1 & 0.5 \\ 0.2 & 0.4 & 0.8 \end{bmatrix} \begin{bmatrix} \mathbf{2} \\ \mathbf{800} \\ \mathbf{143} \end{bmatrix}$$



Bedrooms (2)

Sq. feet (800)

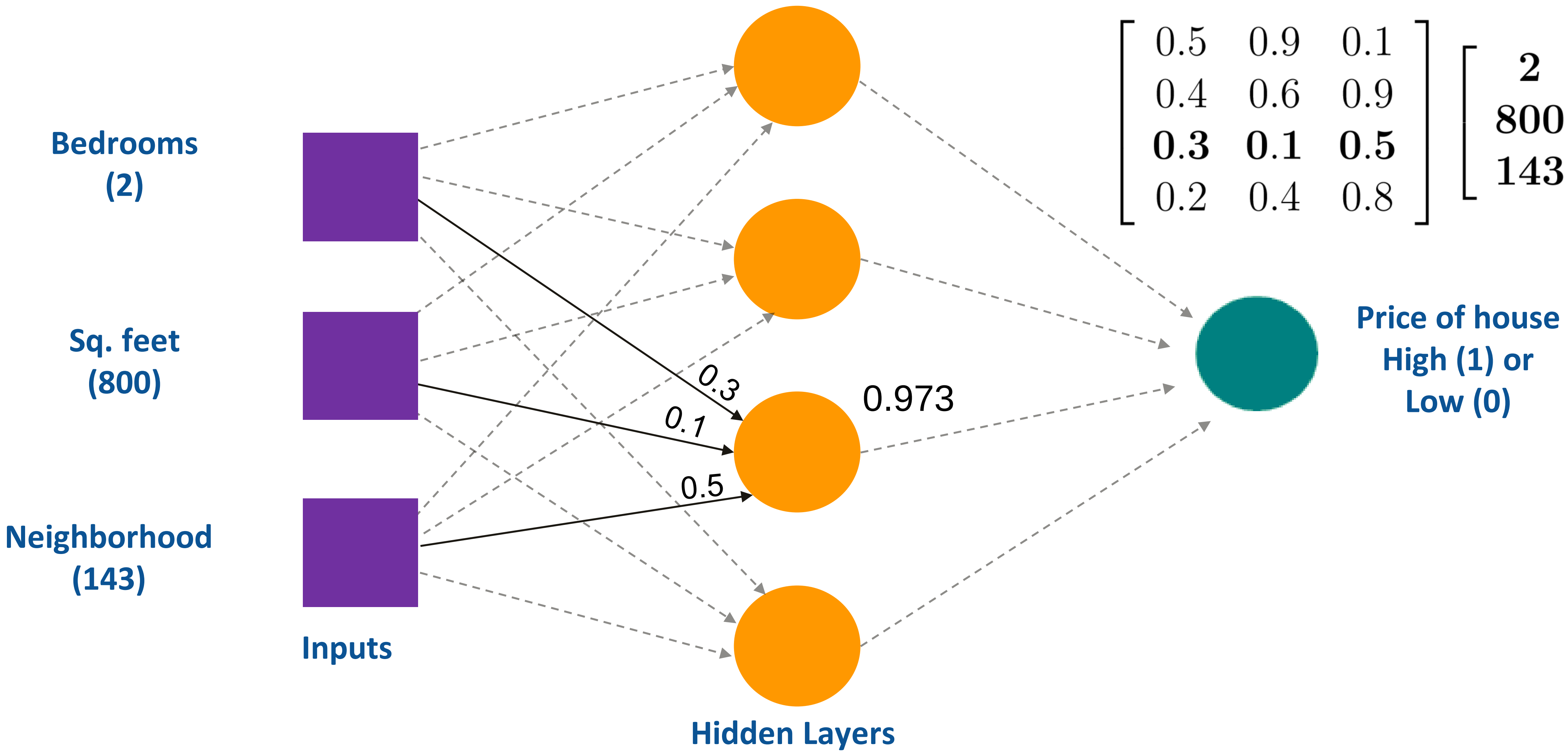
Neighborhood (143)

Inputs

Hidden Layers

Price of house High (1) or Low (0)

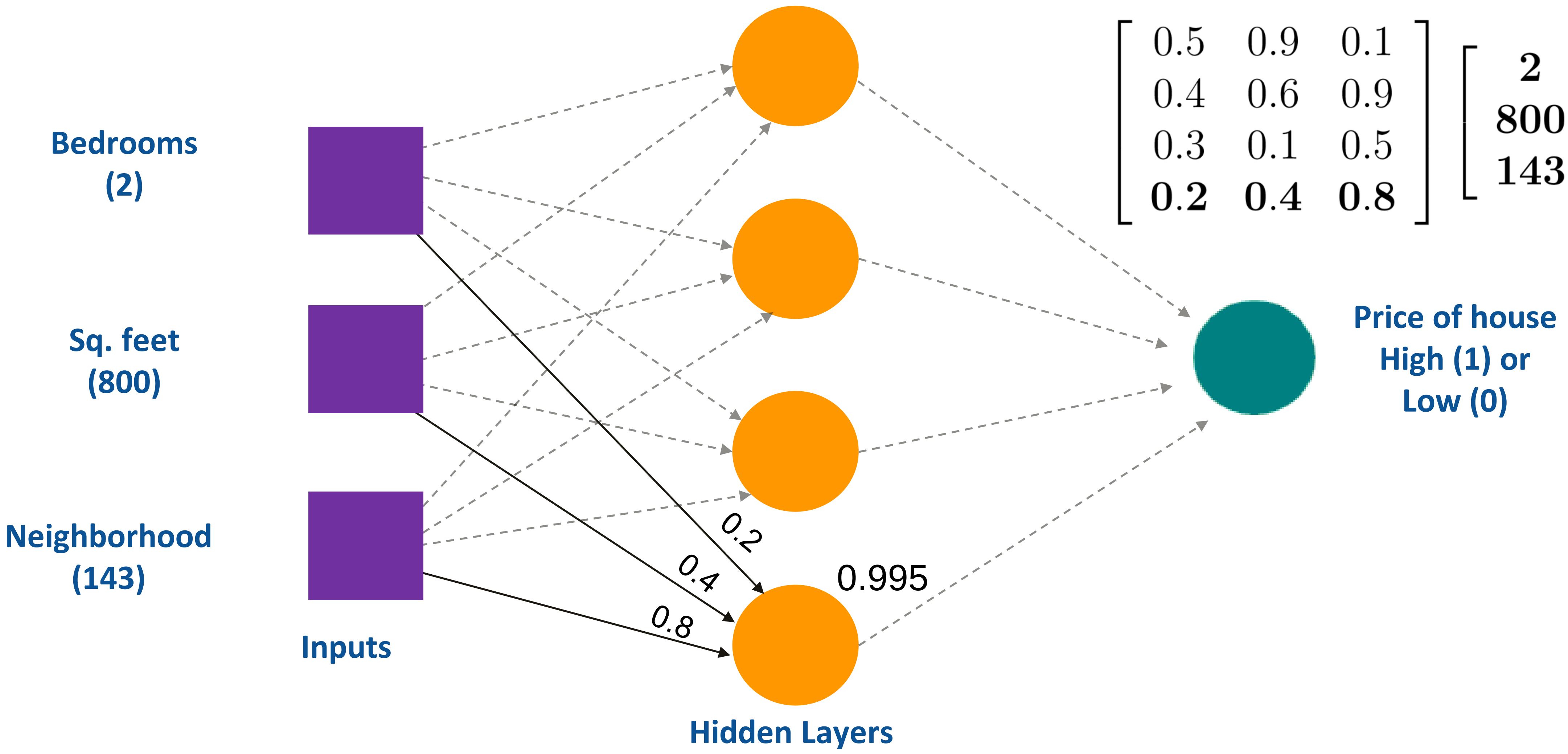
Weights at the third neuron



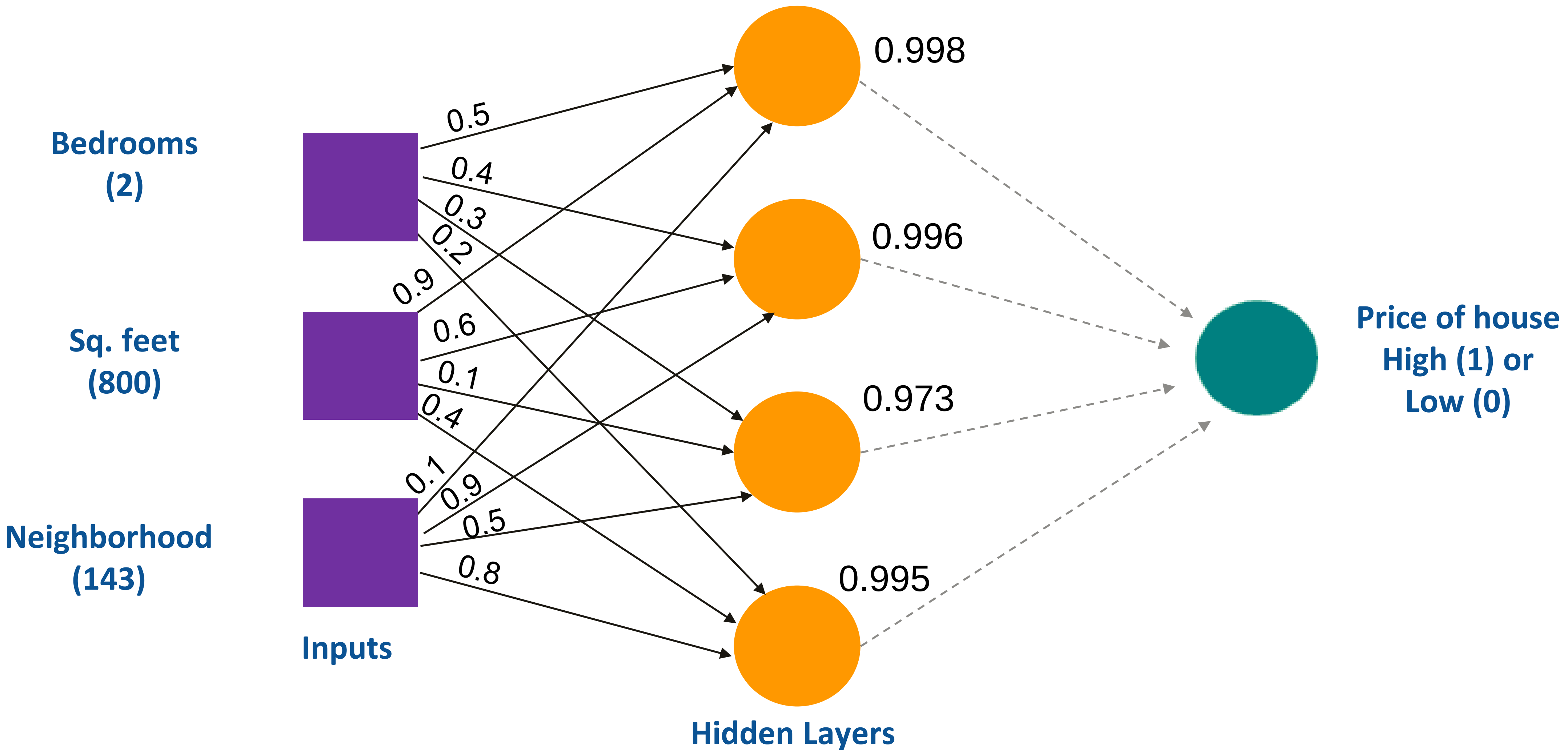
$$\begin{bmatrix}0.5 & 0.9 & 0.1\\ 0.4 & 0.6 & 0.9\\ \mathbf{0.3} & \mathbf{0.1} & \mathbf{0.5}\\ 0.2 & 0.4 & 0.8\end{bmatrix}\begin{bmatrix}\mathbf{2}\\ \mathbf{800}\\ \mathbf{143}\end{bmatrix}$$

Weights at the fourth neuron

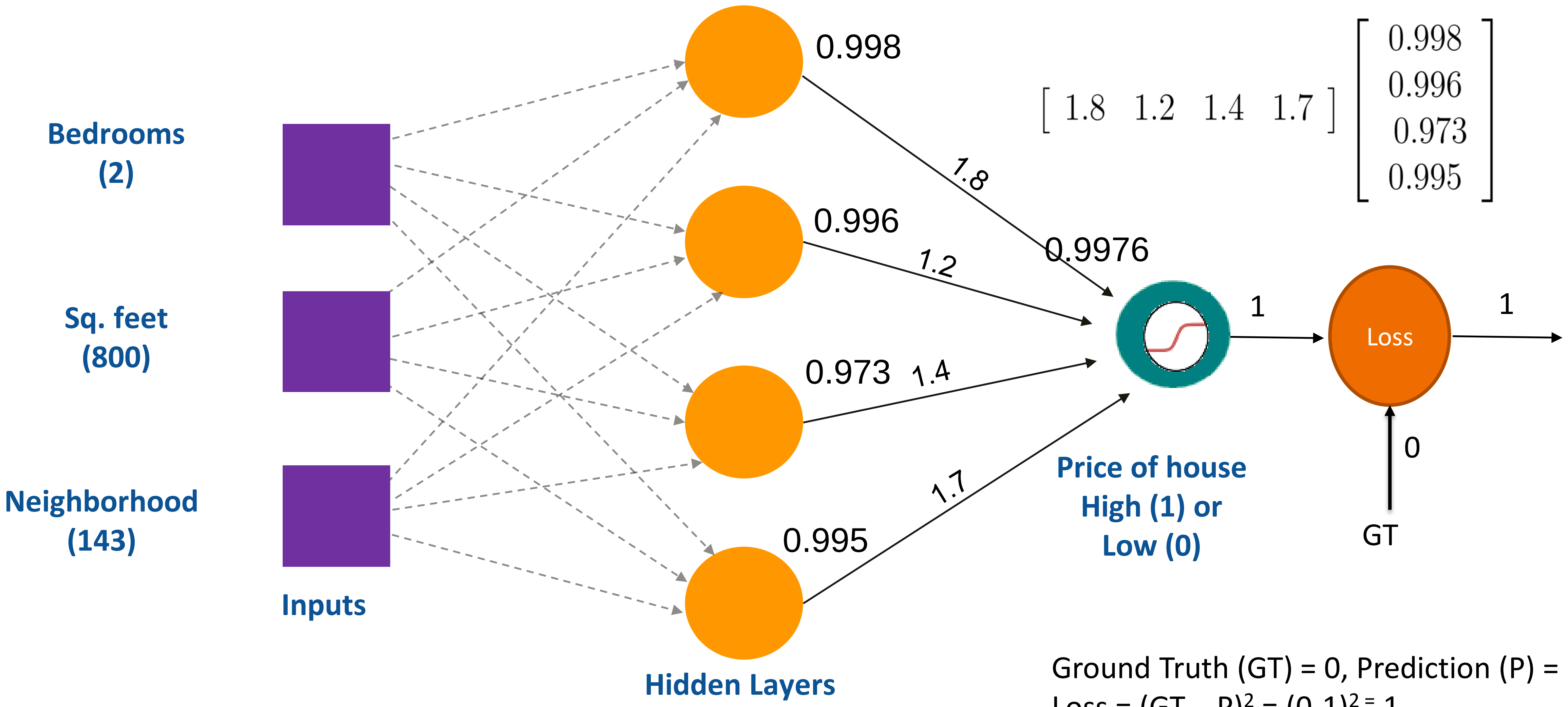
$$\begin{bmatrix} 0.5 & 0.9 & 0.1 \\ 0.4 & 0.6 & 0.9 \\ 0.3 & 0.1 & 0.5 \\ \mathbf{0.2} & \mathbf{0.4} & \mathbf{0.8} \end{bmatrix} \begin{bmatrix} \mathbf{2} \\ \mathbf{800} \\ \mathbf{143} \end{bmatrix}$$



Inputs to the next layer



Computations in next layer

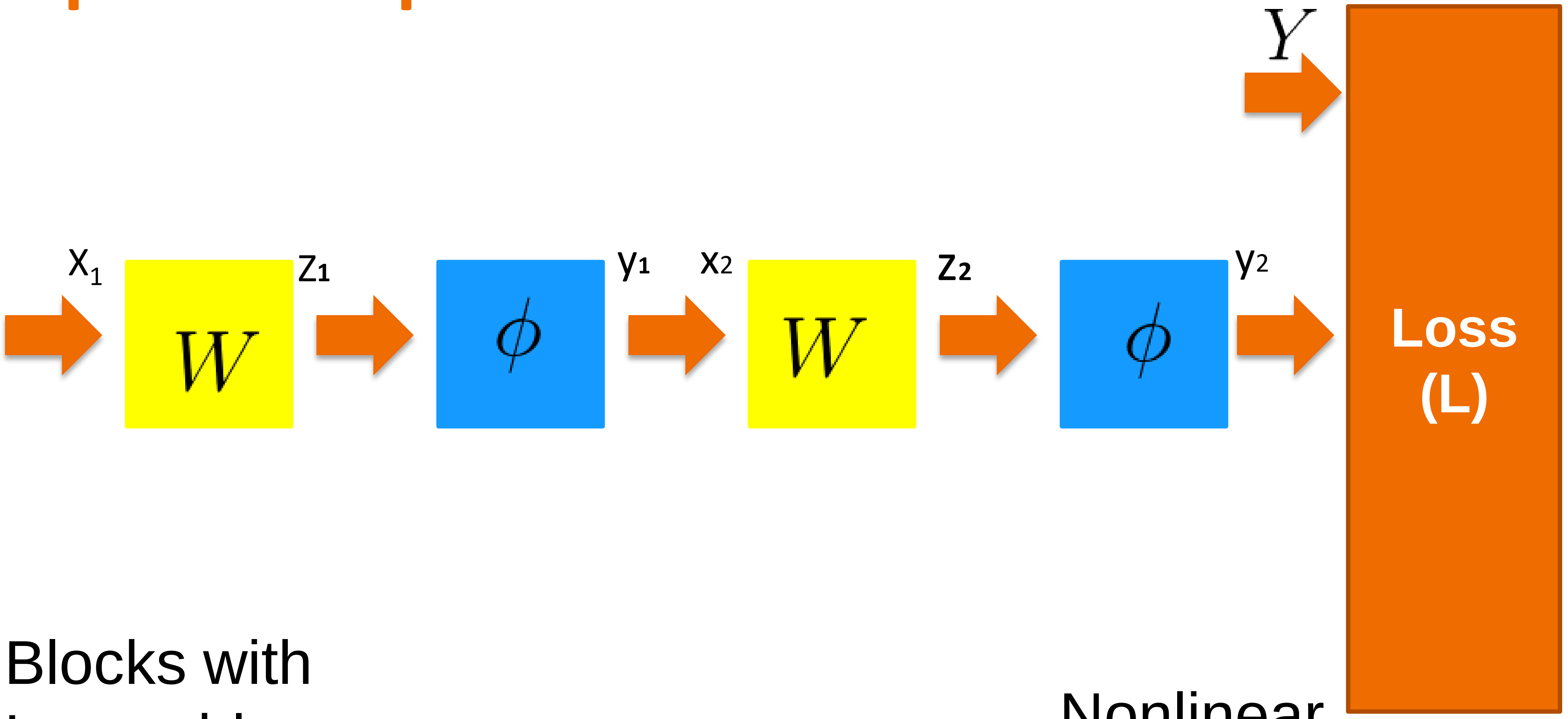


$$\begin{bmatrix} 1.8 & 1.2 & 1.4 & 1.7 \end{bmatrix} \begin{bmatrix} 0.998 \\ 0.996 \\ 0.973 \\ 0.995 \end{bmatrix}$$

Ground Truth (GT) = 0, Prediction (P) = 1

Loss = $(GT – P)^2 = (0-1)^{2 =}$ 1

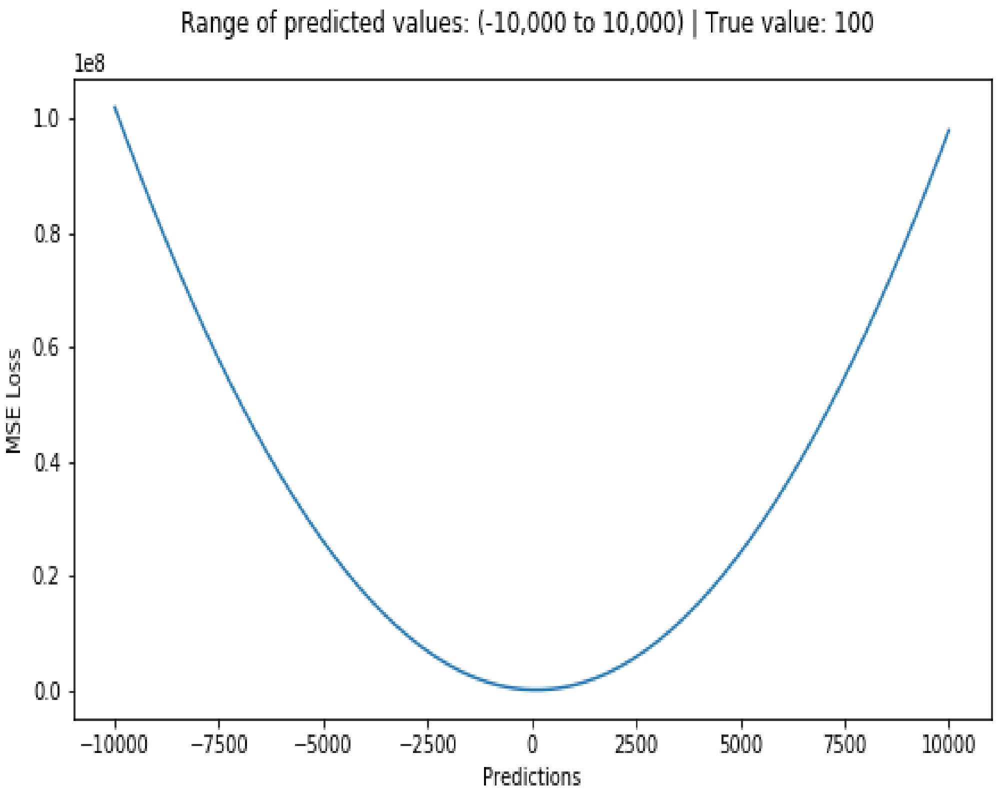
A simpler view point

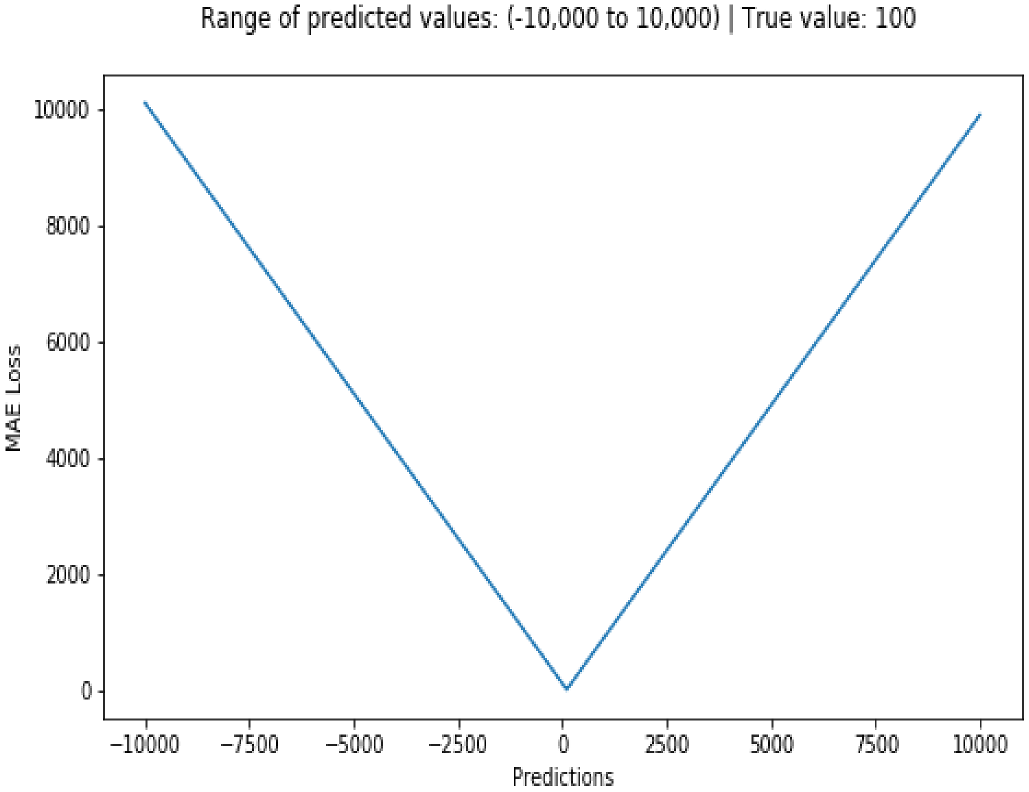


Blocks with Learnable parameters Matrix Multiplication

Nonlinear functions (often non learnable)

Loss Functions





Mean Squared Loss

$$L(y, y^{'}) = (y - y^{'})^2$$

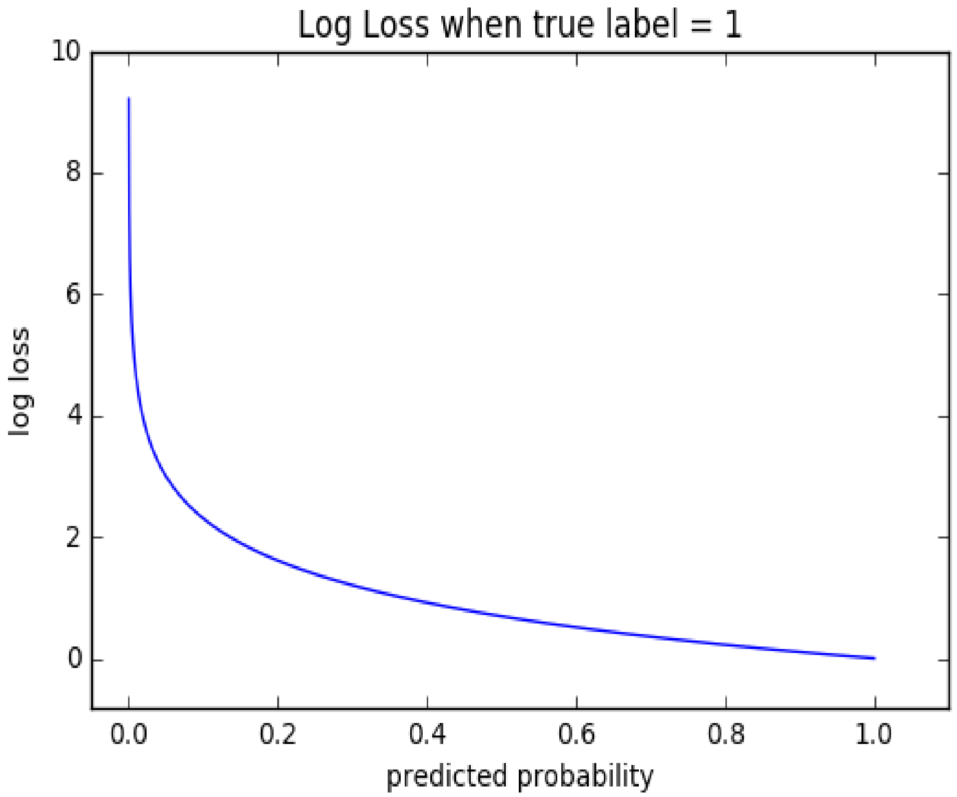
y - Actual value

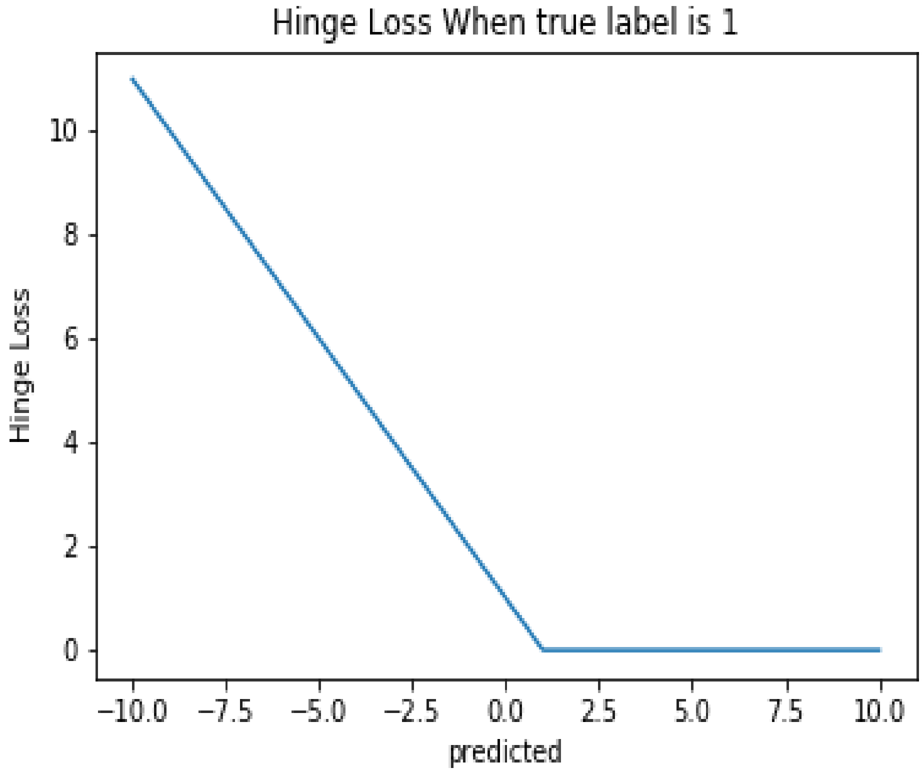
$y^{'}$ - Predicted value

Mean Absolute Loss

$$L(y, y^{'}) = |y - y^{'}|$$

Loss Functions





Cross Entropy Loss

$$L(y, y') = -(y\log(y') + (1 - y)\log(1 - y'))$$

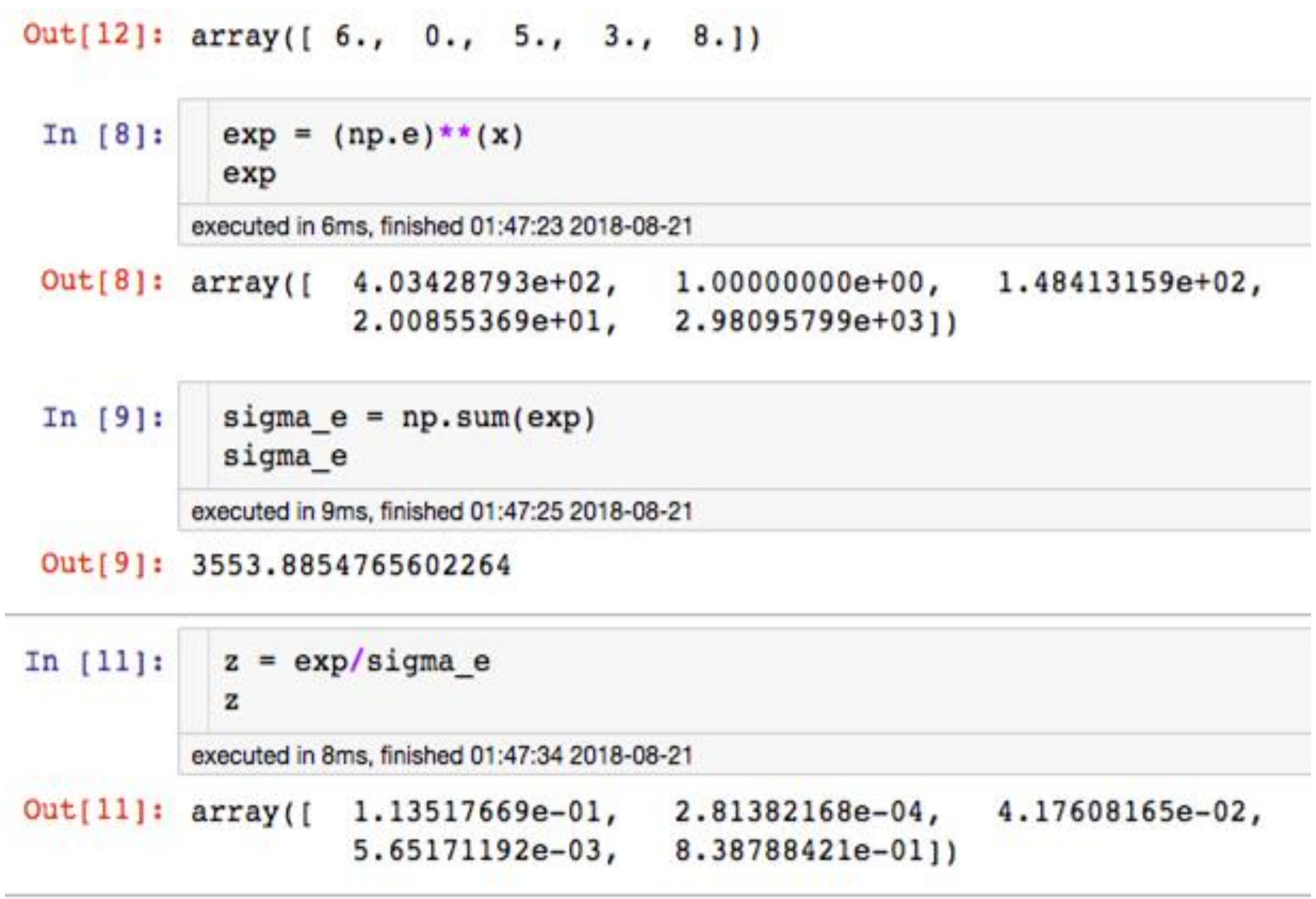
Hinge Loss

$$L(y, y') = max(0, 1 - y * y')$$

y - Actual value

y' - Predicted value

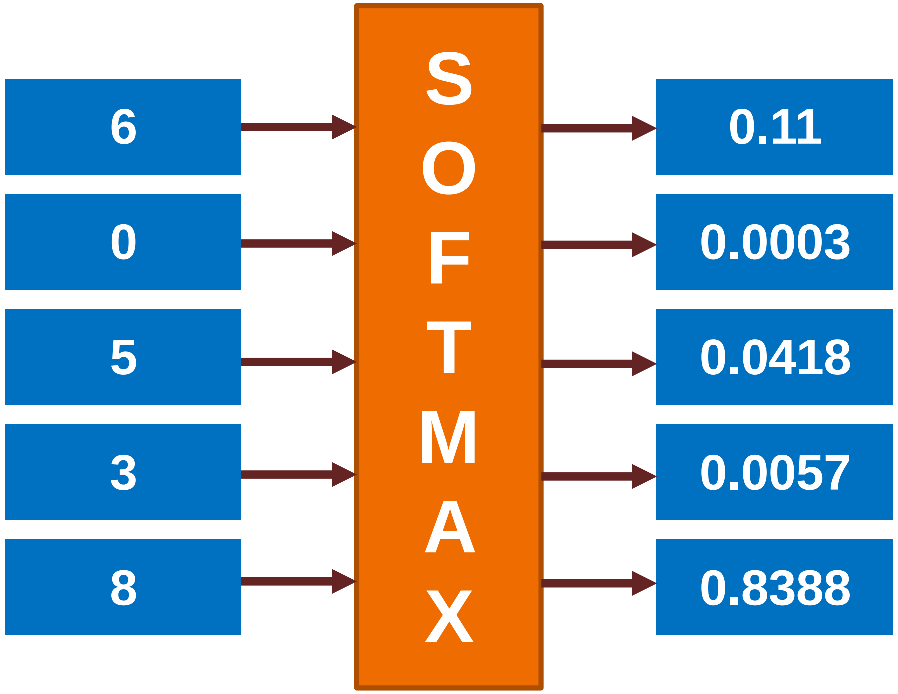
Softmax

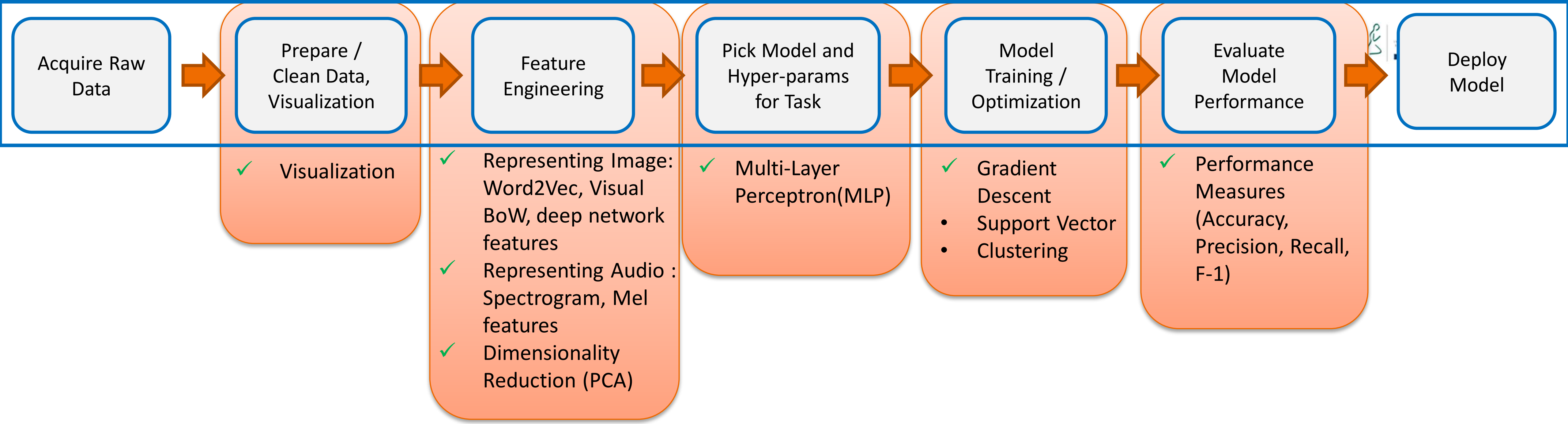


```
Out[12]: array([ 6.,  0.,  5.,  3.,  8.])

In [8]:  exp = (np.e)**(x)
         exp
         executed in 6ms, finished 01:47:23 2018-08-21

Out[8]:  array([  4.03428793e+02,   1.00000000e+00,   1.48413159e+02,
                  2.00855369e+01,   2.98095799e+03])

In [9]:  sigma_e = np.sum(exp)
         sigma_e
         executed in 9ms, finished 01:47:25 2018-08-21

Out[9]:  3553.8854765602264

In [11]: z = exp/sigma_e
         z
         executed in 8ms, finished 01:47:34 2018-08-21

Out[11]: array([  1.13517669e-01,   2.81382168e-04,   4.17608165e-02,
                  5.65171192e-03,   8.38788421e-01])
```

- Normalizes the output.
- K is total number of classes

$$z_n = \frac{e^{x_n}}{\sum_{i=1}^{K} e^{x_i}}$$





Acquire Raw Data

Prepare / Clean Data, Visualization

- ✓ Visualization

Feature Engineering

- ✓ Representing Image: Word2Vec, Visual BoW, deep network features
- ✓ Representing Audio : Spectrogram, Mel features
- ✓ Dimensionality Reduction (PCA)

Pick Model and Hyper-params for Task

- ✓ Multi-Layer Perceptron(MLP)

Model Training / Optimization

- ✓ Gradient Descent
- • Support Vector
- • Clustering

Evaluate Model Performance

- ✓ Performance Measures (Accuracy, Precision, Recall, F-1)

Deploy Model

SVMs and Kernels

Kernel as Similarity Function

Linear Classifier

- Decision boundary: Hyperplane

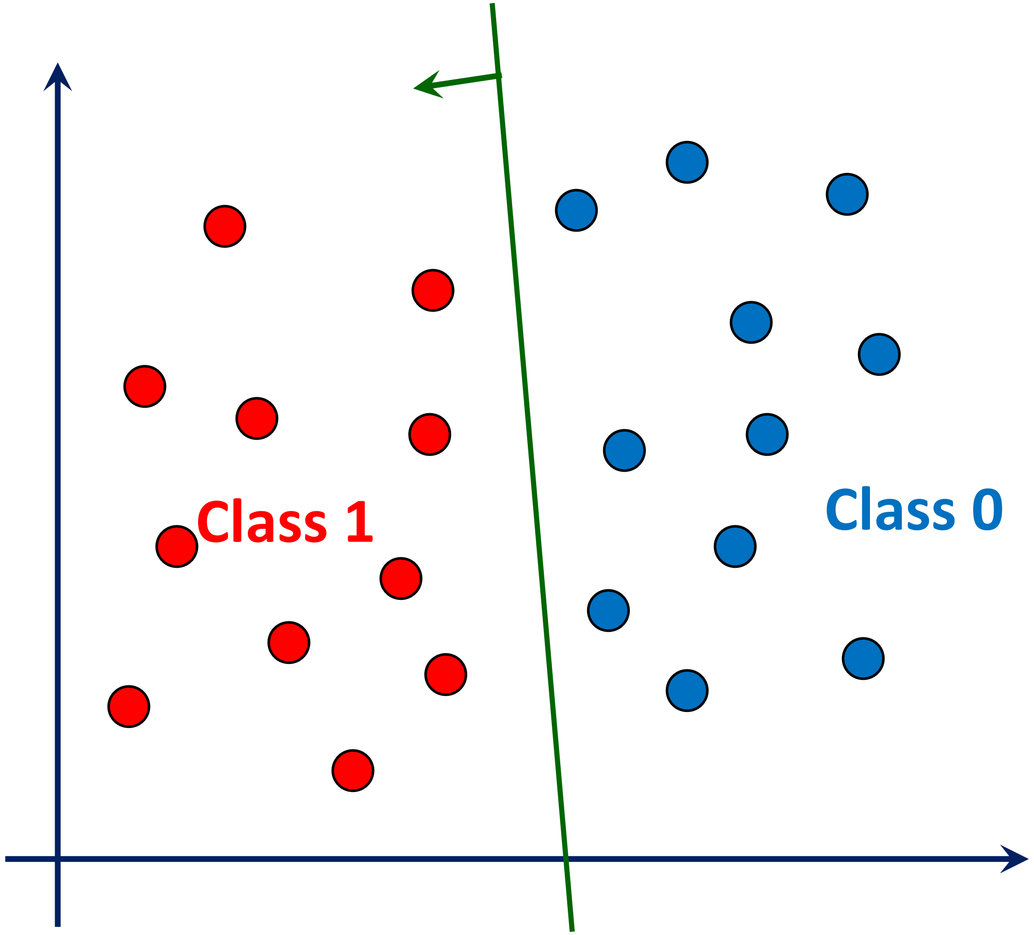
$$w^T x = 0$$

- Class 1 lies on the positive side

$$w^T x > 0$$

- Class 0 lies on the negative side

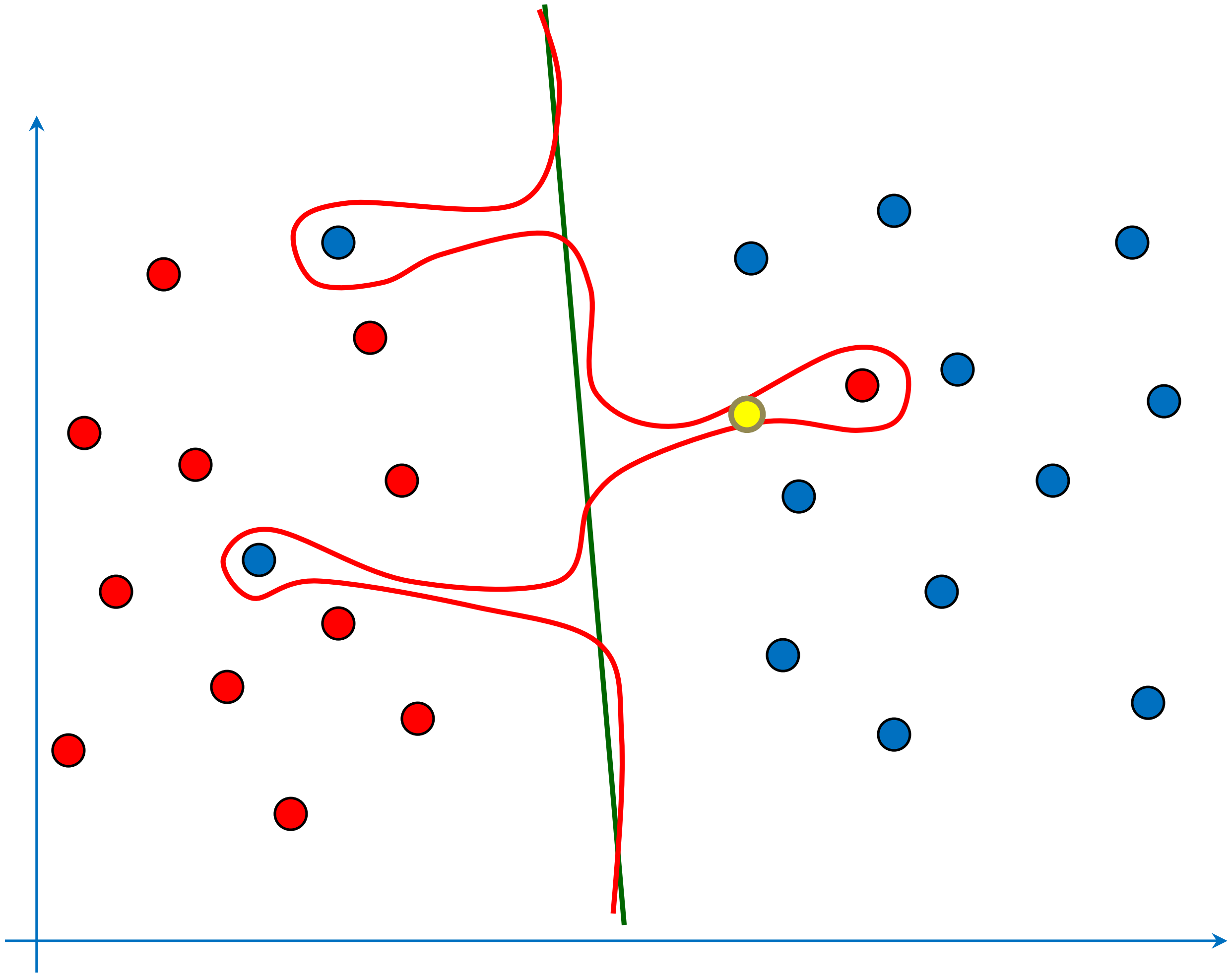
$$w^T x < 0$$



Why Linear? Generalization vs. Complexity

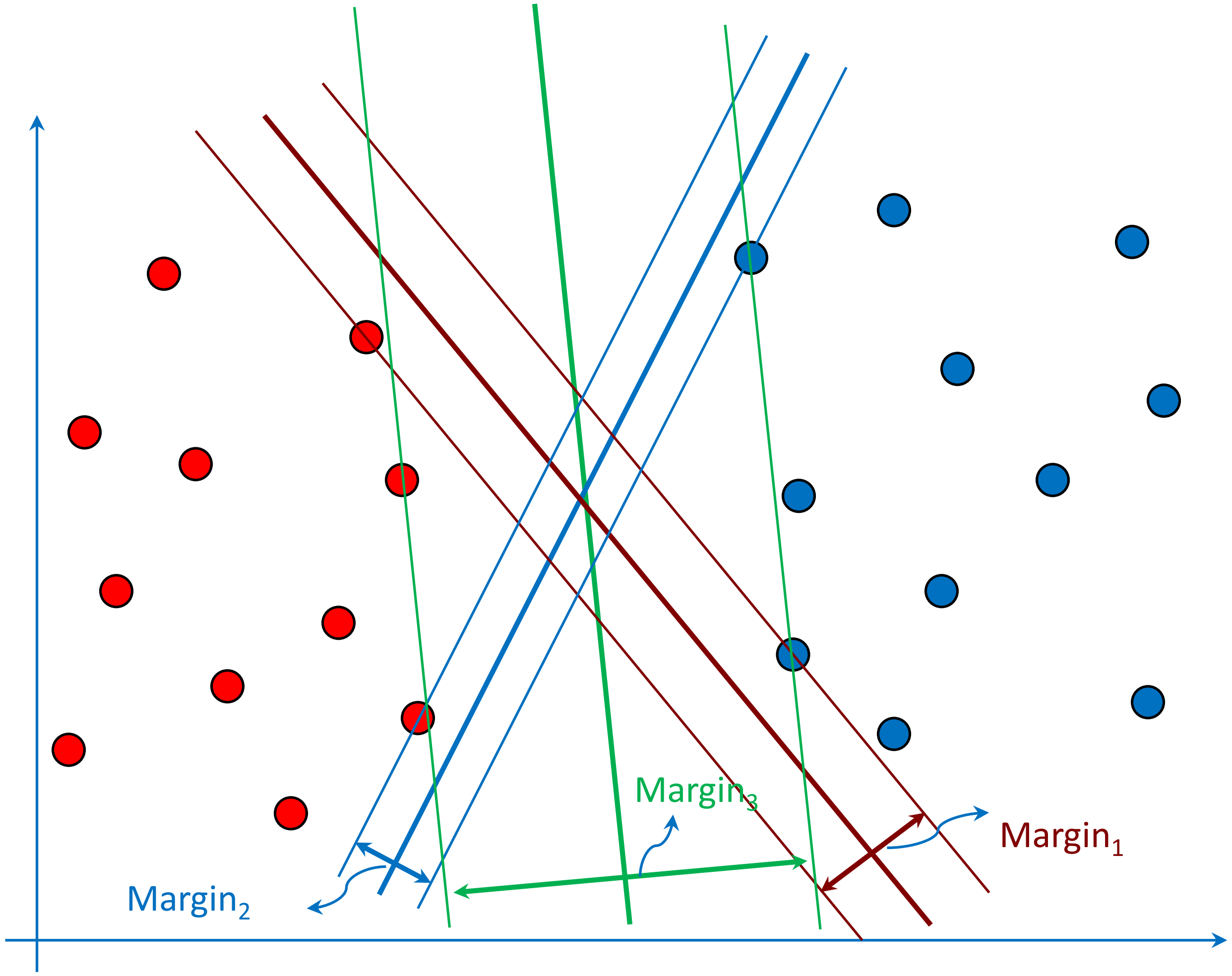
- Is it good to use a complex curve to reduce training error?

Are both solutions equally good?



Margin: The No-mans Band

- Margin: Width of a band around decision boundary without any training samples
- Margin varies with the position and orientation of the separating hyperplane

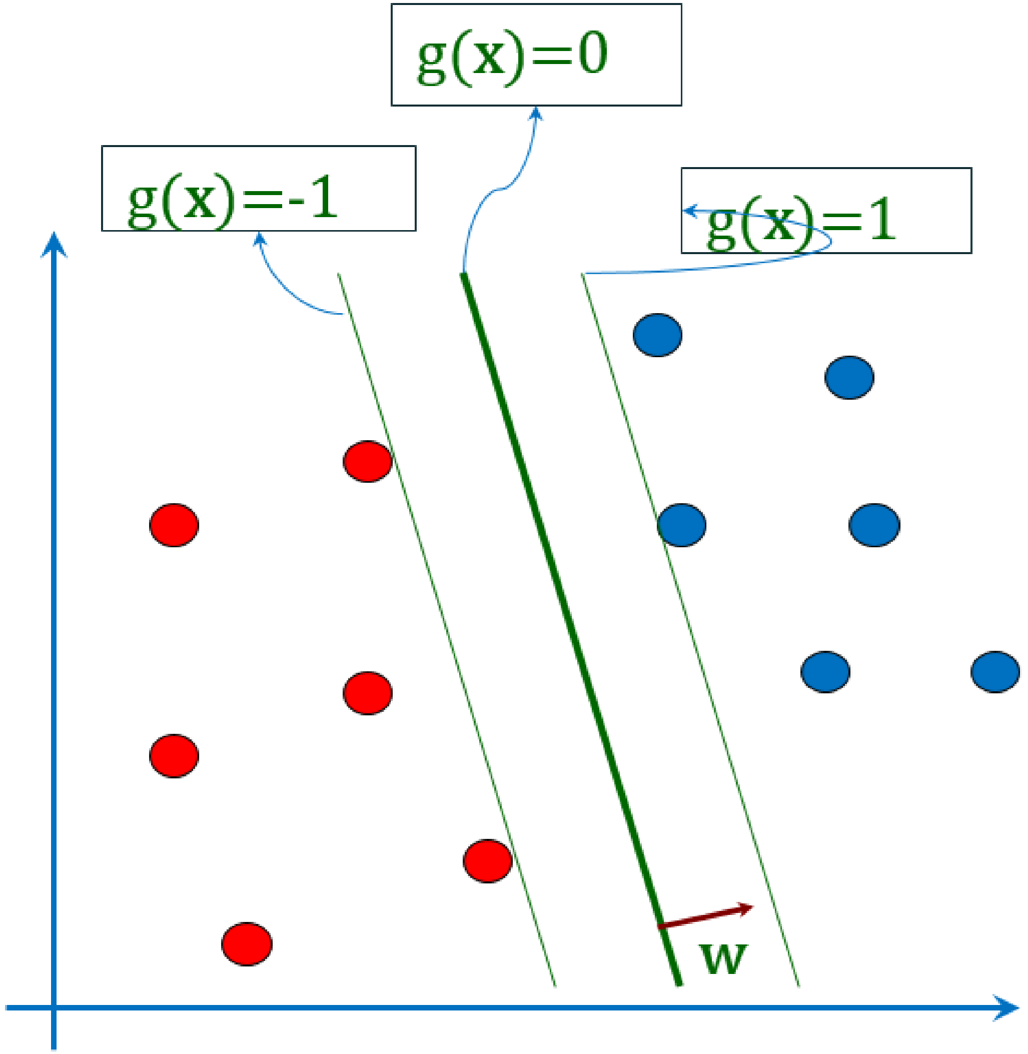


Is a Larger Margin better? Why?

SVM: Formulation

- Let $g(X) = W^TX + b$

- We want to maximize margin:
  - $W^TX_i + b \leq -1$ for $y_i = -1$
  - $W^TX_i + b \geq 1$ for $y_i = 1$
  - Or $y_i(W^TX_i + b) \geq 1$ for i.



SVM Optimization: Convex Optimization

$$min\ \frac{1}{2}W^TW$$

$$y_i(W^Tx_i + b) - 1 \geq 0 \forall i$$

$$y_i \epsilon \{1, -1\}$$

$$J_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\alpha_i \alpha j y_i y_j \boxed{x_i^T x_j} \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

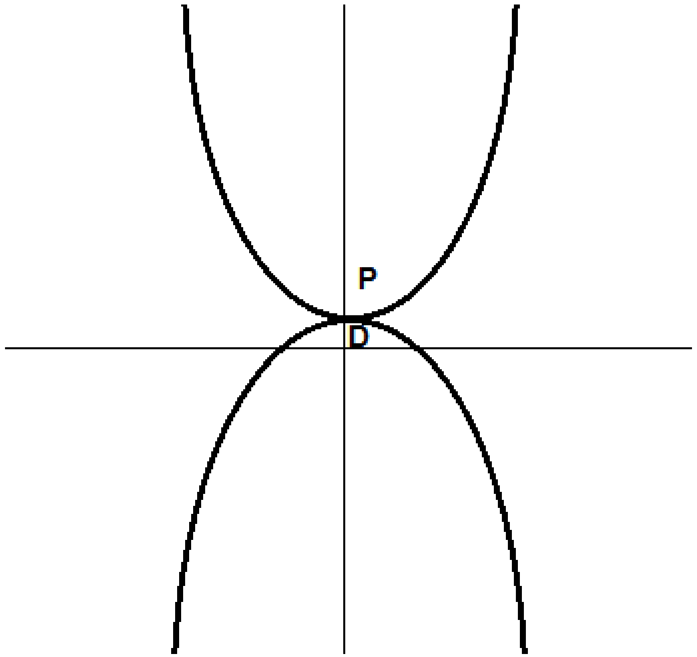
Convex Opt.
Only dot products

$$W = \sum_{i=1}^{N} \boxed{\alpha_i y_i x_i}$$

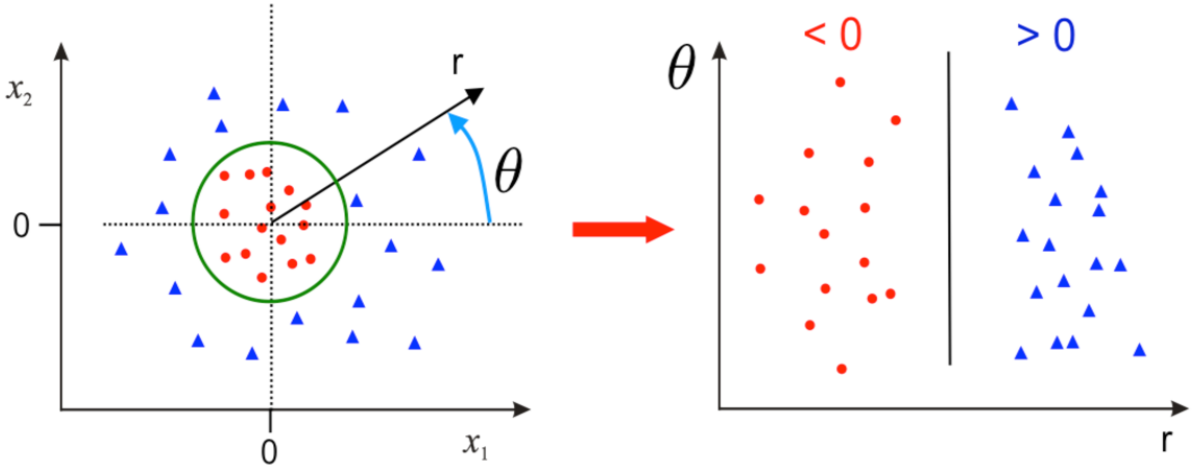
Support Vectors

$$W^T X = \sum^{N} \alpha_i y_i \boxed{x_i^T x}$$

Only dot products



Nonlinearity with Feature Maps

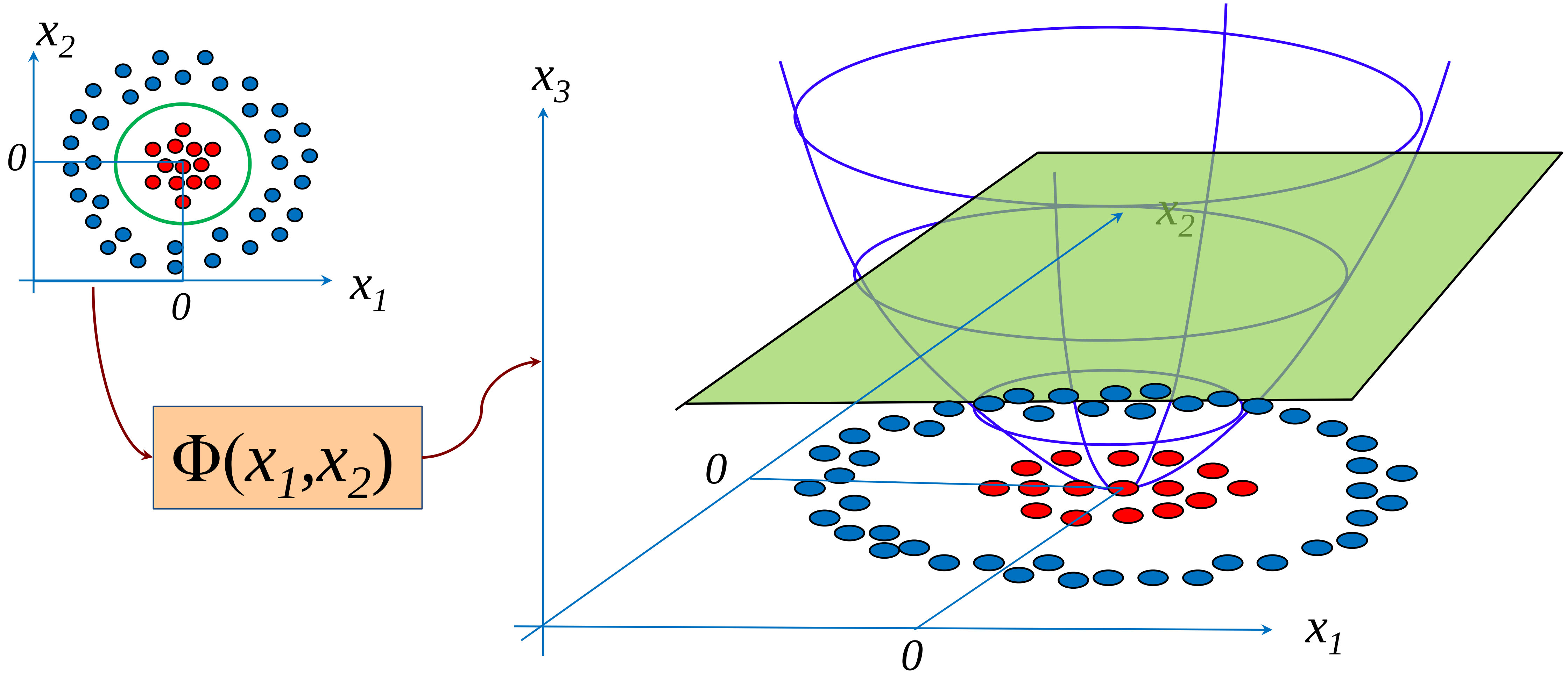


$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} r \\ \theta \end{pmatrix}$$

- With a “smart” feature map, a linearly non-separable problem can be converted to a separable problem!!

Non-linear Mapping

$$x_3 = x_1^2 + x_2^2$$



Φ is a non-linear mapping into a possibly high-dimensional space

Kernel Strategy

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

What we need is only $w^T x = \sum_{i=1}^{N} \alpha_i y_i x_i^T x$

We can do the same in a new feature space:

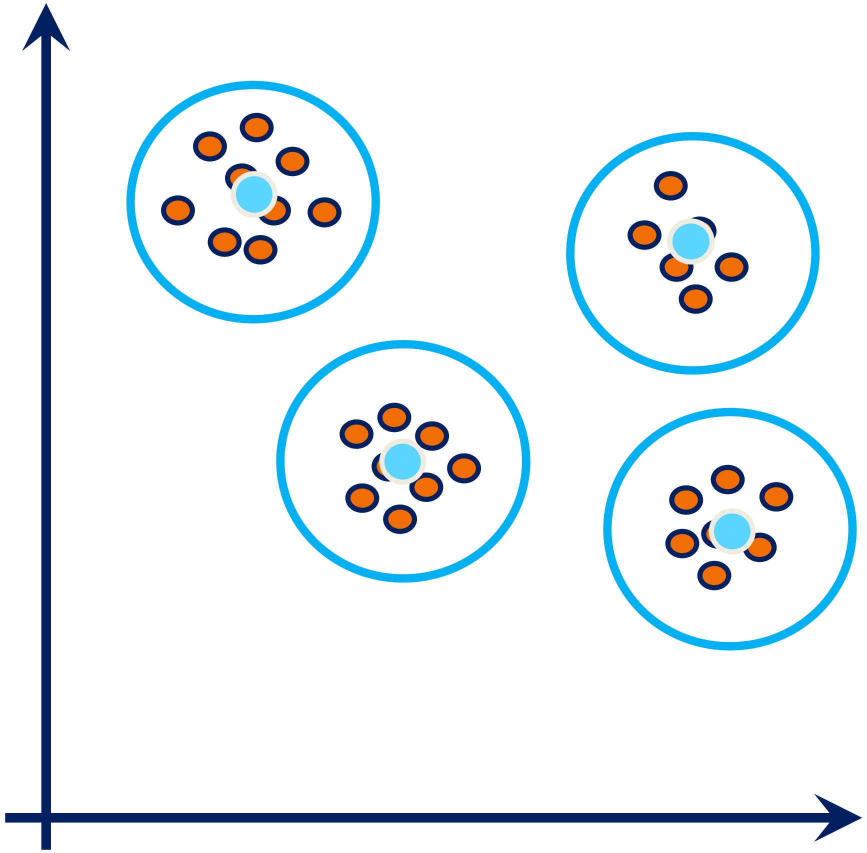
$$w^T x = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)^T \phi(x) \qquad w^T x = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x)$$

Clustering

Identifying Similar Patterns

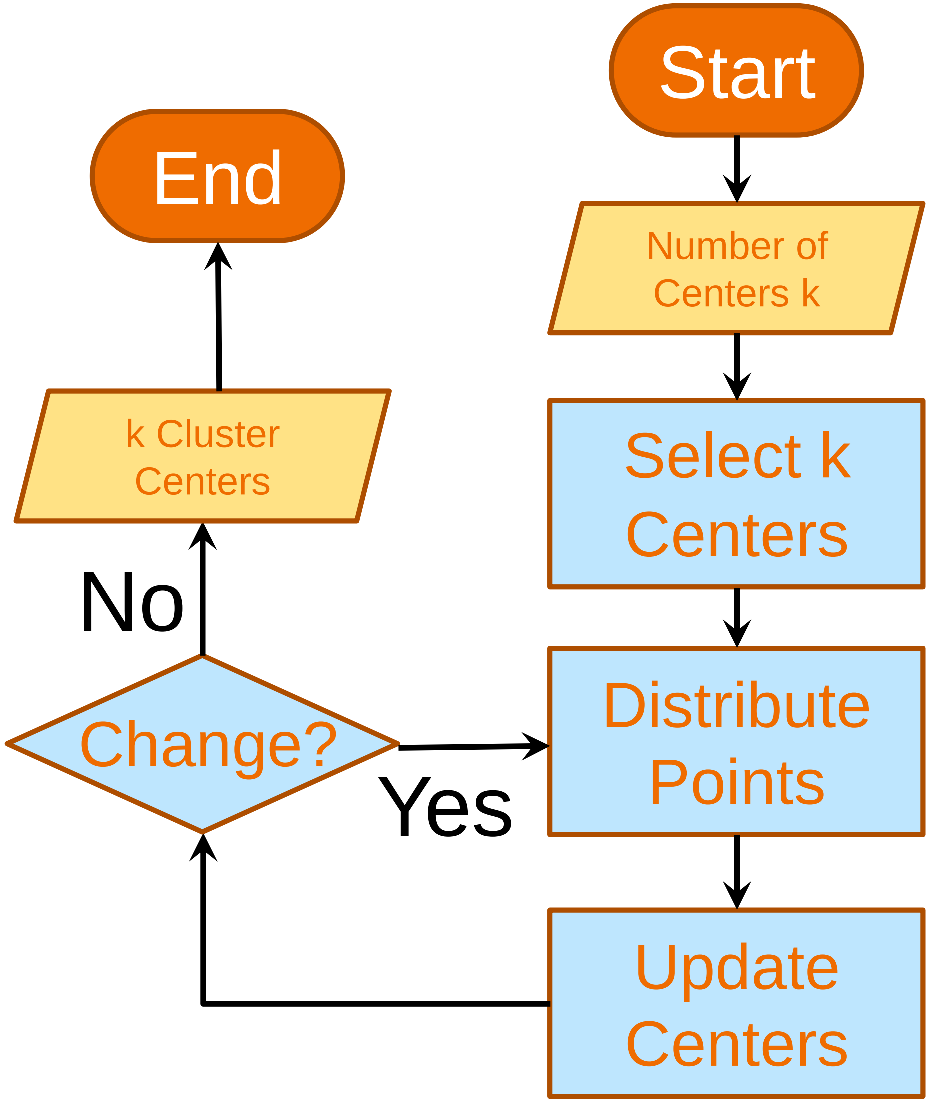
K-Means

- You are given N points
- How do we find k clusters?
  - What if we know the cluster centers?
- How do we find the cluster centers?
  - What if we know the k clusters?



K-Means

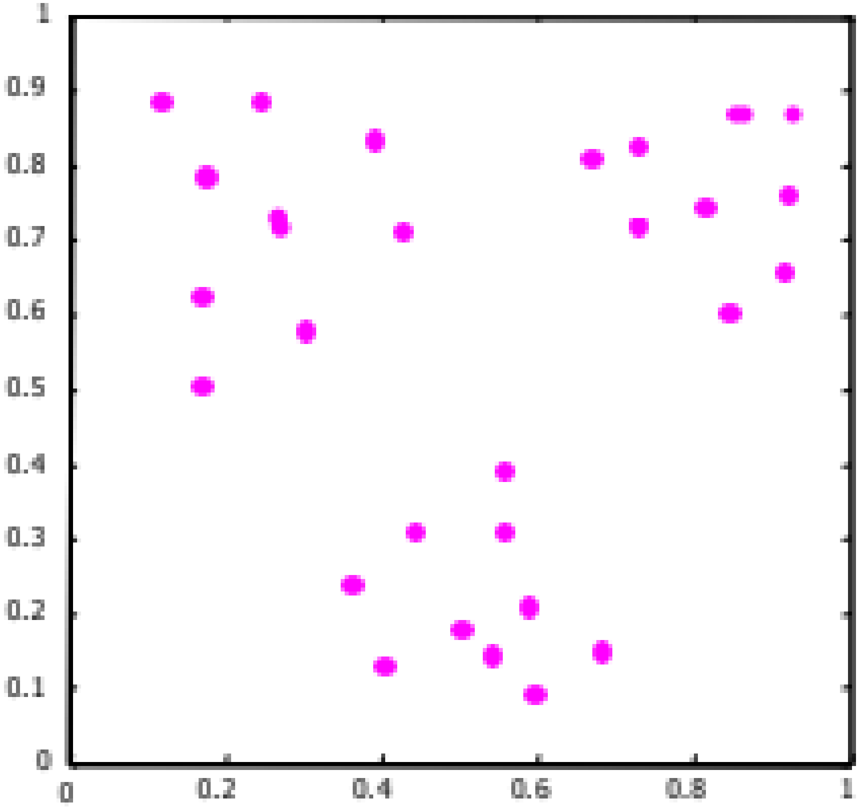
1. Input: k (number of clusters)
2. Randomly select k centers
3. Distribute Points
4. Update Centers
5. Repeat 3,4 till convergence
6. Output: Cluster centers



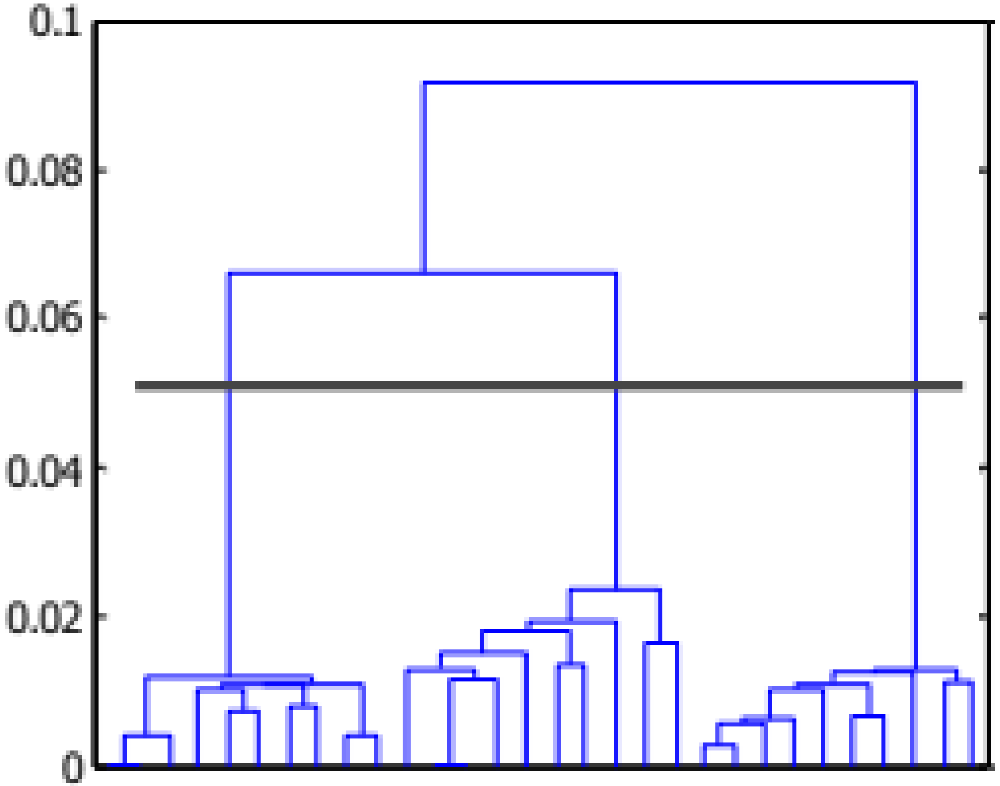
Single-Link Algorithm

- Form a hierarchy for the data points (dendrogram), which can be used to partition the data
- The “closest” data points are joined to form a cluster at each step

Single-Link Algorithm

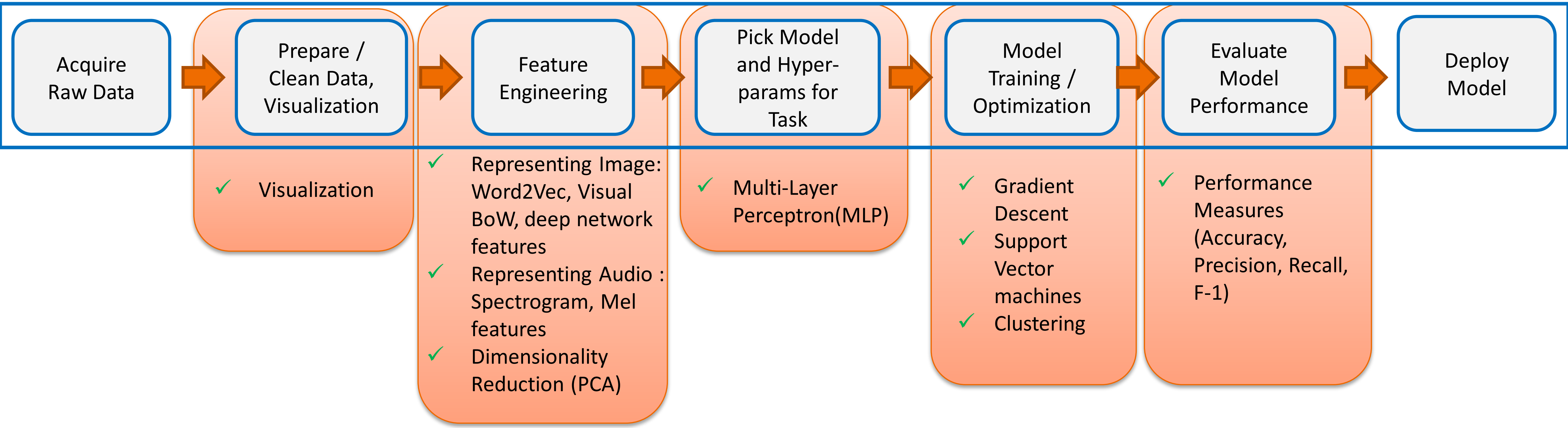


Data



Dendrogram

Summary



Acquire Raw Data

Prepare / Clean Data, Visualization

- ✓ Visualization

Feature Engineering

- ✓ Representing Image: Word2Vec, Visual BoW, deep network features
- ✓ Representing Audio : Spectrogram, Mel features
- ✓ Dimensionality Reduction (PCA)

Pick Model and Hyper-params for Task

- ✓ Multi-Layer Perceptron(MLP)

Model Training / Optimization

- ✓ Gradient Descent
- ✓ Support Vector machines
- ✓ Clustering

Evaluate Model Performance

- ✓ Performance Measures (Accuracy, Precision, Recall, F-1)

Deploy Model

Thanks!!

Questions?