SCIENCE
ACADEMY

RBF: Learning Models (2/2)

- Use iterative approach (similar to EM algorithm):
  - Fix γ, solve for $\underline{\boldsymbol{w}}$ (one-step learning)
  - Fix $\underline{\boldsymbol{w}}$, solve for γ (gradient descent)
- **Step 1**
  - Assume γ is known and fixed
  - Learn $\underline{\boldsymbol{w}}$
- Impose perfect interpolation:

$$E_{in} = \frac{1}{n} \sum (h(\underline{\boldsymbol{x}}_i) - y_i)^2 = 0$$

- **Problem**:

$$h(\underline{\boldsymbol{x}}_j) = \sum_i w_i \exp(-\gamma \|\underline{\boldsymbol{x}}_i - \underline{\boldsymbol{x}}_j\|^2) = \sum_i w_i \phi_{i,j} = \underline{\boldsymbol{\phi}}_j^T \underline{\boldsymbol{w}} = y_i$$

  - N equations (one per point) and N unknowns $\underline{\boldsymbol{w}}$
  - $\underline{\underline{\boldsymbol{\Phi}}}$ is known, function of data set and γ

SCIENCE ACADEMY

Learning RBF Models

- The problem in matrix form is:

$$\underline{\underline{\boldsymbol{\Phi}}} \cdot \underline{\boldsymbol{w}} = \underline{\boldsymbol{y}}$$

  - If $\underline{\underline{\boldsymbol{\Phi}}}$ is invertible, then $\underline{\boldsymbol{w}} = \underline{\underline{\boldsymbol{\Phi}}}^{-1}\underline{\boldsymbol{y}}$
    - Desired values on training points
    - Exponential interpolates other points
  - If $\underline{\underline{\boldsymbol{\Phi}}}$ is not invertible, optimize in least square sense:

$$\text{argmin}_{\underline{\boldsymbol{w}}} E_{in} = \sum_i (h(\underline{\boldsymbol{x}}_j) - y_i)^2$$

    - Compute pseudo-inverse (assuming $\underline{\underline{\boldsymbol{\Phi}}}^T \underline{\underline{\boldsymbol{\Phi}}}$ is invertible)
    - Assign weights:

$$\underline{\boldsymbol{w}} = (\underline{\underline{\boldsymbol{\Phi}}}^T \underline{\underline{\boldsymbol{\Phi}}})^{-1} \underline{\underline{\boldsymbol{\Phi}}}^T \underline{\boldsymbol{y}}$$

- **Step 2**
  - Assume $\underline{\boldsymbol{w}}$ is known and fixed
  - Learn γ

RBF Network vs Neural Networks

- The regression model for Neural Networks and RBF model is **similar**:
  - **RBF**:

$$h(\underline{\boldsymbol{x}}) = \sum_i w_i e^{-\gamma \|\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}_i\|^2} = \underline{\boldsymbol{w}}^T \underline{\underline{\phi}}$$

  - **Neural networks**:

$$h(\underline{\boldsymbol{x}}) = \Theta(\underline{\boldsymbol{w}}^{(L)T} \underline{\boldsymbol{x}}^{(L)}) = \Theta(\underline{\boldsymbol{w}}^{(L)T} \underline{\boldsymbol{\Theta}}(\underline{\underline{\boldsymbol{W}}}^{(L-1)} \ldots))$$

- **Difference**:
  - RBF has a single layer
  - Neural networks have multiple layers
- **Similarities**:
  - Combine features with weights using dot product
  - Extract features from inputs
    - RBF features: $e^{-\gamma \|\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}_i\|^2}$, always > 0
    - NN hidden layers: features can be > 0 or < 0

SCIENCE ACADEMY

RBF Network vs SVM

- The **model form** is the same:
  - RBF:

$$h(\underline{\boldsymbol{x}}) = \text{sign}(\sum_i w_i e^{-\gamma \|\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}_i\|^2}$$

  - SVM:

$$h(\underline{\boldsymbol{x}}) = \text{sign}(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}} + b)$$

- The interpretation is completely different (interpolation vs large margin)
  - RBF: all vectors (or centers of few clusters) contribute to the model
  - SVM: only support vectors contribute to the model

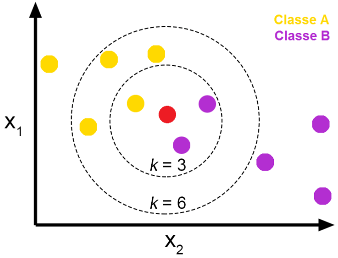
SCIENCE ACADEMY

K-Nearest Neighbor (KNN) Model

- The **model form** is like:

$$h_{\underline{\boldsymbol{w}}}(\underline{\boldsymbol{x}}) = \frac{1}{n} \sum_{\underline{\boldsymbol{x}}_i \text{ closest to } \underline{\boldsymbol{x}}} w_i$$

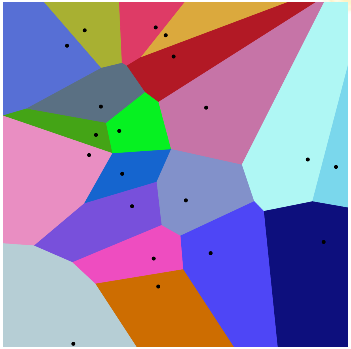
- **Idea**:
  - Closeness implies a distance (e.g., euclidean metric) or similarity (e.g., a kernel)
  - Consider the k closest points to the evaluation point $\underline{\boldsymbol{x}}$
  - Take an average of their response



SCIENCE ACADEMY

KNN: Intuition of Number Degrees of Freedom

- Nearest neighbor model ($k = 1$)
  - Use response of closest point to $\underline{\boldsymbol{x}}$
  - Similar to Voronoi tessellations: each point has a region where it is the closest and assigns its output to that region
- k is **the only parameter** for KNN
  - For $k = 1 \implies N$ neighborhoods, one around each training point
  - For $k = N \implies$ single neighborhood
  - Effective number of parameters: $\frac{N}{k}$, imagining N/k non-overlapping neighborhoods



SCIENCE ACADEMY

KNN: Assumptions on the Data

- KNN makes **no assumption on data**
  - Opposite of linear model with strong data assumption
- KNN assumes **locality in parameter space**
  - Model is constant in example's neighborhood
  - E.g., $k = 1$: Voronoi tessellation (low-bias/high-variance)
  - E.g., $k = N$: Average value (high-bias/low-variance)

KNN: Training and Test Error

- For $k = 1$
  - No error on training set (low bias / high variance) assuming non-noisy target
  - E_{out} larger than E_{in}
- Increasing k
  - Training error E_{in} increases
  - Test error E_{out} decreases, then increases
  - Typical model complexity behavior in bias-variance diagrams

KNN vs RBF Models

- **Similarities**
  - K-Nearest Neighbor is a *discrete* version of the RBF model
- **Differences:**
  - Consider only the *k closest examples* to the point $\underline{\boldsymbol{x}}$ (not all examples in the training set)
  - Use a *constant kernel* (responses are not weighted by distance)

- ***Clustering***
- Anomaly Detection

K-Means Clustering: Problem Formulation

- N *unlabeled* points $\{\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2, \ldots, \underline{\boldsymbol{x}}_N\}$
- Partition points into K clusters $S_1, \ldots, S_K$
  - Each cluster defined by center $\underline{\boldsymbol{\mu}}_k$
  - Each point $\underline{\boldsymbol{x}}_i$ assigned to cluster $c(\underline{\boldsymbol{x}}_i)$
  - Unknowns are $c(\underline{\boldsymbol{x}}_1), \ldots, c(\underline{\boldsymbol{x}}_N), \underline{\boldsymbol{\mu}}_1, \ldots, \underline{\boldsymbol{\mu}}_K$
- Minimize distance between each $\underline{\boldsymbol{x}}_i$ and assigned center $\underline{\boldsymbol{\mu}}_k$ where $k = c(\underline{\boldsymbol{x}}_i)$

$$J(c_1, \ldots, c_N, \mu_1, \ldots, \mu_K) = \sum_{k=1}^{K} \sum_{\underline{\boldsymbol{x}}_n \in S_k} \|\underline{\boldsymbol{x}}_n - \underline{\boldsymbol{\mu}}_k\|^2 \text{(scanning the clusters)}$$

$$= \sum_{i=1}^{N} \|\underline{\boldsymbol{x}}_i - \underline{\boldsymbol{\mu}}_{c(\underline{\boldsymbol{x}}_i)}\|^2 \qquad \text{(scanning the points)}$$

- **K-means clustering is NP-hard** (combinatorial) and thus intractable
  - In fact there are K^N possible assignments

UNIVERSITY OF MARYLAND 18 56
SCIENCE ACADEMY

K-Means Clustering: Lloyd's Algorithm

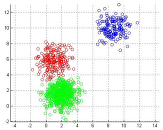
- **Start with a random assignment** of N points to K clusters
  - Better than picking random centroids
- **Each iteration** has 2 steps
  - **Step 1**: Move centroid
    - Move each cluster's centroid to the mean point
    - Iterate over K clusters
    - $\underline{\boldsymbol{\mu}}_k \leftarrow \frac{1}{|S_k|} \sum_{\underline{\boldsymbol{x}}_n \in S_k} \underline{\boldsymbol{x}}_n$
  - **Step 2**: Cluster assignment
    - Assign each $\underline{\boldsymbol{x}}_n$ to the closest cluster center
    - Iterate over N points
    - $S_k \leftarrow \{\underline{\boldsymbol{x}}_n : \|\underline{\boldsymbol{x}}_n - \underline{\boldsymbol{\mu}}_k\| \leq \|\underline{\boldsymbol{x}}_n - \underline{\boldsymbol{\mu}}_l\| \; \forall l \neq k\}$

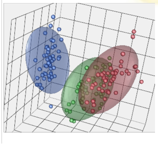
K-Means Clustering: Convergence

- **K-means algorithm converges** since:
  - Finite number of possible partitions (and values of objective functions)
  - Objective function $J(\cdot)$ always decreases
- **Objective function always decreases**
  - Cost function $J(\underline{\boldsymbol{\mu}}_1, ..., \underline{\boldsymbol{\mu}}_K, c_1, ..., c_N)$ depends on:
    - Centroids $c_1, .., c_N$
    - Point assignments $\underline{\boldsymbol{\mu}}_1, ..., \underline{\boldsymbol{\mu}}_K$
  - K-means minimizes J by:
    - Adjusting centroids (fixed assignments)
    - Adjusting assignments (fixed centroids)
  - Similar to coordinate descent
- Generally converges to a **local minimum**
  - Run K-means multiple times with different random initializations
  - Choose best result

K-Means Clustering: Non-Separable Clusters

- For simplicity, you tend to imagine a clear separation between clusters
  - Clusters, especially in high dimensions, are not obviously separable
- Using K-means on data not obviously separated
  - E.g., market segmentation
  - E.g., t-shirt sizing
    - Collect height and width of customers
    - Run K-means
    - Find optimal way to split population into 3 sizes (S, M, L)

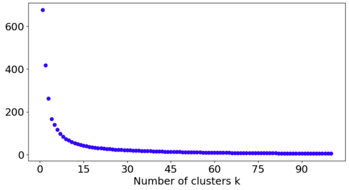




SCIENCE ACADEMY

Choosing the Number of Clusters

- Often unclear how many clusters K exist
  - Visual analysis can be inconclusive in 2D or 3D
  - More difficult in high dimensions
- **Elbow Method**
  - Vary clusters K
  - Compute optimal cost function $J(\cdot)$
  - Choose K at "elbow" point if visible
  - Elbow absent if curve resembles hyperbole $\approx 1/K$



- **End-to-end approach**
  - Choose K to optimize later stages
  - E.g., more t-shirt sizes (more clusters) $\Longrightarrow$
    - Satisfy customers
    - Complicates manufacturing
    - Increases inventory management

SCIENCE ACADEMY

Interpretation of Clusters

- Cluster meaning **often interpreted manually** (difficult to automate)
  - Examine cluster centroids
    - Centroid values show “typical” point in each cluster
    - High, low, or zero feature values highlight key characteristics
  - Analyze distribution of features per cluster
    - Plot histograms or boxplots for each feature
    - Identify features that vary sharply across clusters
  - Visualize clusters in 2D or 3D
    - E.g., PCA, t-SNE, UMAP
    - Understand separation and internal structure
  - Compare clusters to external labels if available
    - See if clusters align with known real-world groups
  - Train a classifier like decision tree
    - Important features for predicting cluster reveal their meaning
- **Example: Customer Segmentation**

Cluster	Age	Annual Income	Spending Score	Label
Cluster 1	25 yrs	30K	90	Young Big Spenders
Cluster 2	50 yrs	80K	40	Comfortable Mid-Lifers
Cluster 3	35 yrs	120K	20	High Income, Low Spending Customers

- Clustering
- ***Anomaly Detection***

Anomaly Detection: Problem Formulation

- **Problem**:
  - $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N\}$ examples with features $\underline{\boldsymbol{x}} \in \mathbb{R}^P$ for good instances
  - Detect bad/anomalous instances
- **Algorithm**:
  - Unknown characteristics of *"bad instances"*
  - Learn common traits of *"good instances"* using unsupervised learning
    - Find distribution for $\underline{\boldsymbol{x}}_i$: $\Pr(\underline{\boldsymbol{x}}$ is good$)$
  - Pick features
    - Find "sensitive" features, e.g., ratio between CPU load and network traffic
  - Estimate distribution $\Pr(\underline{\boldsymbol{x}}$ is good$)$
  - Choose threshold ε
  - For new instance $\underline{\boldsymbol{x}}_{new}$, if $\Pr(\underline{\boldsymbol{x}}_{new}$ is good$) \leq \varepsilon$ flag as anomaly

SCIENCE ACADEMY

Anomaly Detection: Example of Aircraft Engines

- **Problem**
  - Test aircraft engines to identify anomalies in a new engine
- **Solution**:
  - Features $\underline{\boldsymbol{x}}_i$ can be:
    - Heat generated
    - Vibration intensity
    - ...
  - Collect data for all engines
  - Model Pr($\underline{\boldsymbol{x}}$ is good)
  - Decide if a new engine is acceptable Pr($\underline{\boldsymbol{x}}_{new}$ is good) $\leq \varepsilon$ or needs more testing

SCIENCE ACADEMY

Anomaly Detection: Example of Hacked Account

- **Problem**
  - Find if an account for a given user *i* was hacked
- **Solution**:
  - Model features that represent "user *i* activity"
  - Features $\underline{\boldsymbol{x}}_i$ can be:
    - How many times s/he logs a day
    - How many times s/he fails to enter the password
    - How fast s/he types
    - How many pages s/he visits
    - How many times s/he posts comments
    - ...
  - Model Pr($\underline{\boldsymbol{x}}$ is good)
  - Identify unusual users by checking Pr($\underline{\boldsymbol{x}}_{new}$ is good) $\leq \varepsilon$

SCIENCE ACADEMY

Anomaly Detection: Example of Data Center

- **Problem**
  - Monitor servers in a data center to find malfunctioning or hanged servers
- **Solution**:
  - Features $\underline{\boldsymbol{x}}_i$ can be:
    - Memory in use
    - CPU load
    - Network traffic
    - Number of reads/writes per sec
    - CPU load / network activity
    - ...
  - Model $\Pr(\underline{\boldsymbol{x}}$ is good$)$
  - Identify unusual systems by checking $\Pr(\underline{\boldsymbol{x}}_{new}$ is good$) \leq \varepsilon$

SCIENCE ACADEMY

Using a Gaussian Model for Anomaly Detection

- Aka "density estimation"
- Given N examples $\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N \in \mathbb{R}^p$
- Ensure that the **features have a Gaussian distribution**
    - If not, apply some transformations, e.g., $\log(x_i + k)$
- **Estimate the parameters** of the Gaussian model $f_X(\underline{\boldsymbol{x}})$
- Given a new example $\underline{\boldsymbol{x}}_{new}$ **compute** $\Pr(\underline{\boldsymbol{x}}_{new}$ is good$) \leq \varepsilon$ to flag an anomaly

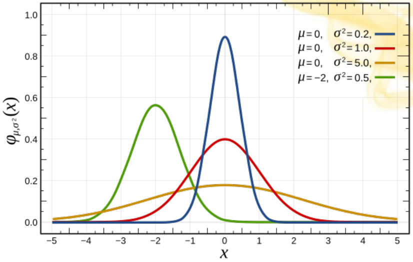
Estimate Univariate Gaussian Model

- You have N (scalar) examples $\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N \in \mathbb{R}$ representing *"good instances"*
- Assume the data is generated by a Gaussian distribution

$$X \sim \mathcal{N}(\mu, \sigma)$$

which has a PDF:

$$f_X(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$



- Estimate mean and sigma with maximum likelihood:

$$\mu = \frac{1}{N} \sum_i x_i$$

$$\sigma^2 = \frac{1}{N-1} \sum_i (x_i - \mu)^2$$

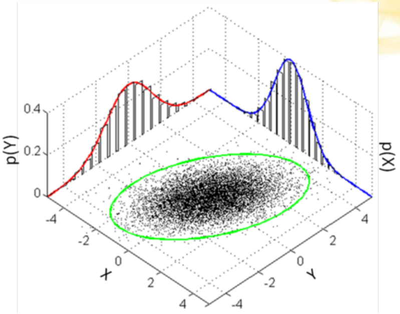
SCIENCE ACADEMY

Estimate Multivariate Independent Gaussian Model

- You have N examples $\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_N \in \mathbb{R}^p$ for *"good instances"*
- Assume independence of the features, the PDF of a multi-variate Gaussian X is:
$$f_X(\underline{\boldsymbol{x}}; \underline{\boldsymbol{\mu}}, \underline{\boldsymbol{\sigma}}) = \prod_{i=1}^{p} f_{X_i}(x_i; \mu_i, \sigma_i)$$
- Infer the parameters μ_i and σ_i using discrete formulas to get the complete model

Estimate a Multi-Variate Gaussian Model

- **Problem**:
  - Often features vary together (e.g., network use and CPU load), causing mis-classifications with independent assumptions
  - Components of $\underline{\boldsymbol{x}}_{new}$ are within range but nonsensical together
    - E.g., low network use with high CPU load
- **Solution 1**:
  - Engineer features to highlight anomalies
  - Address variable correlation not modeled in independent Gaussian models
- **Solution 2**:
  - Estimate the entire multivariate model instead of assuming independence



Estimate a Multi-Variate Gaussian Model

- The PDF of a multi-variate Gaussian is:

$$f_X(\underline{\boldsymbol{x}}; \underline{\boldsymbol{\mu}}, \underline{\underline{\boldsymbol{\Sigma}}}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\underline{\underline{\boldsymbol{\Sigma}}}|^{\frac{1}{2}}} \exp(-\frac{1}{2}(\underline{\boldsymbol{x}} - \underline{\boldsymbol{\mu}})^T \underline{\underline{\boldsymbol{\Sigma}}}^{-1}(\underline{\boldsymbol{x}} - \underline{\boldsymbol{\mu}}))$$

- Estimate:

$$\underline{\boldsymbol{\mu}} \in \mathbb{R}^d = \frac{1}{N} \sum_{k=1}^{N} \underline{\boldsymbol{x}}_k$$

$$\underline{\underline{\boldsymbol{\Sigma}}} \in \mathbb{R}^{d \times d} = \{s_{ij}\} = \frac{1}{N-1} \sum_{k=1}^{N} (\underline{\boldsymbol{x}}_k - \underline{\boldsymbol{\mu}})(\underline{\boldsymbol{x}}_k - \underline{\boldsymbol{\mu}})^T$$

- Model requires more examples to train due to more parameters
- Independence between variables decomposes multivariate Gaussian into product of Gaussian distributions

Evaluate Anomaly Detection Systems

- To evaluate models one needs to:
  - Compare different models
  - Tune hyperparameters (e.g., ε) of models
  - Estimate out-of-sample error
- As always we should use a single real number for comparison
  - Use any classification metric, e.g.,
    - True/false positive/negative rate
    - Precision or recall
  - F-score
- *Labeled* data is still needed to rate models

$$y = 0 \text{ good}$$
$$y = 1 \text{ anomalous}$$

SCIENCE ACADEMY

Evaluate Anomaly Detection Systems

- Often, anomalous examples $y = 1$ are much fewer than good examples $y = 0$
  - E.g., 10,000 good vs 20 bad examples
  - Address class imbalance for accurate model performance
- **Algorithm**:
  - Pick 60% of $y = 0$ data to train (only on good examples)
  - Split remaining $y = 0$ and $y = 1$ into validation and test sets
    - Ensure both sets represent the overall dataset
    - Train, validation, and test sets should not overlap but have the same characteristics
    - Helps evaluate model performance accurately
  - Use validation set to compare models, estimate hyperparameters
    - E.g., ε is the threshold for anomaly detection
  - Use test set to evaluate final model
    - Train on normal data, test on both normal and anomalous data
- **Aircraft engine example**

Dataset	$y = 0$ (Good Engines)	$y = 1$ (Bad Engines)
Total Example	10,000	20
Train Set	6,000	0
Validation Set	2,000	10
Test Set	2,000	10

Anomaly Detection vs Supervised Learning

- Even in unsupervised learning, you need labeled data for model evaluation
- Difference with supervised learning:
  - Anomaly detection/unsupervised learning: Train only on good examples
  - Supervised learning: Train on both good and bad examples
- Use:
  - Anomaly detection/unsupervised learning:
    - Learn from good examples due to few anomalous examples
    - Strong prior on the model
    - Future anomalous examples unknown (no prior)
  - Supervised learning: Less skewed classes in training set
  - Not a clear-cut decision

SCIENCE ACADEMY