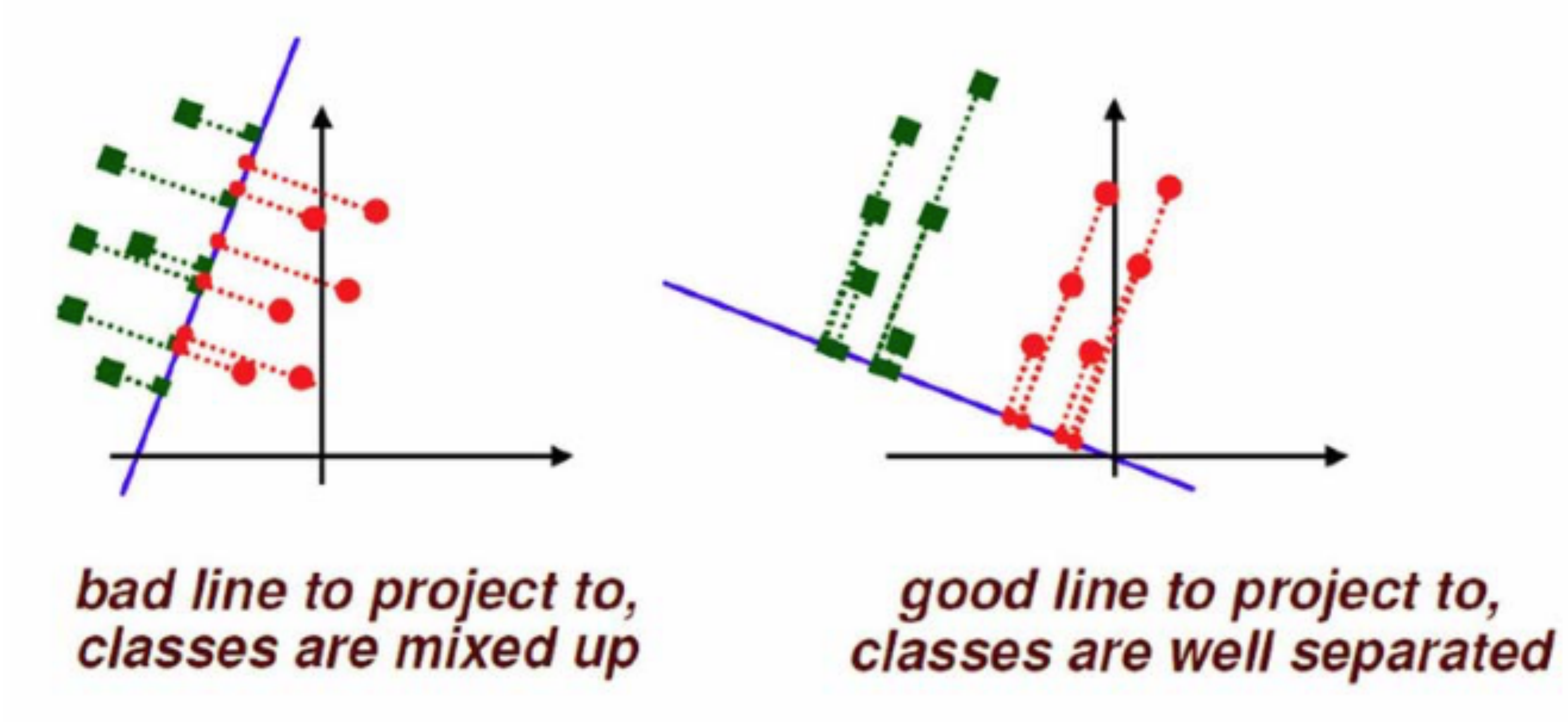
Basic Ideas

Projection



- Projecting points onto a projection matrix
- PCA (Principal Component Analysis)
  - Choose the number of dimensions we want, k < D and project the original data onto the principal components.
- LDA (Linear Discriminant Analysis)
  - Find projection to a line such that samples from different classes are well separated

Information gain / entropy

1. entropy
   - Given probability of event v_1, ..., v_n as $P(v_1)$, ..., $P(v_n)$ we can compute entropy $H(P(v_1),...,P(v_n)) = \sum_{i=1}^{n}(-P(v_i)log_2P(v_i))$
   - Entropy measure the randomness of the data
2. Information gain (IG)
   - $IG(A) = H(\frac{p}{p+n}, \frac{n}{p+n}) - reminder(A)$
   - Choose the attribute with the largest IG!

Feature types

1. Categorical
    - Examples: Car Model, School
2. Finite Discrete Valued
    - Ordering still matters, but there's only so many values out there
3. Continuous Valued
    - Examples: Blood Pressure, Height

Standardization

- Standardized data has **Zero mean** and **Unit deviation**.
- When do we standardizing data?
    - **Before we start using data** we typically want to standardize non-categorical features (this is sometimes referred to as normalizing them).
- How to standardizing data?
    1. Treat each feature independently
    2. Center it (subtract the mean from all samples)
    3. Make them all have the same span (divide by standard deviation)
- Why do we need to standardizing data?
    - If we used the data as-is, then one feature may have more influence than the other.

LSE (least squared estimate)

MLE (maximum likelihood estimate)

Overfitting / Underfitting

- Identifying (detect)
    - [under-fitting] Don't do well on either the training or the testing set
    - [over-fitting] Do well on the training set but poorly on the testing set
- Solving
    - under-fitting
        - Make a more complex model (May involve need more features)
        - Trying a different algorithm
    - over-fitting

        - Use a less complex model (May involve using less features)
        - Try a different algorithm
        - Get more data
        - Use a third set to choose between hypothesis (called a validation set)
        - Add a penalization (or regularization) term to the equations to penalize model complexity

Bayes' Rule

- $$P(y=i|f=x)=\frac{P(y=i)P(f=x|y=i)}{P(f=x)}$$
- In Bayes' Rule we call
    - $P(y=i|f=x)$ the posterior (what we want)
    - $P(y=i)$ the prior (probability that $y=i$)
    - $P(f=x|y=i)$ the **likelihood** (likelihood of generating x given y)
    - $P(f=x)$ the evidence

Evaluation

1. RMSE *root mean squared error*
    - $$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i-\widehat{y}_i)^2}$$
2. Accuracy
    - $$\frac{CorrectNum}{TotalNum}$$
3. Precision, Recall, F-Measure
    - Error Type
        - Example: FP = Negative Examples, Predicted Positive
        - True positive = Hit
        - True negative = Correct rejection
        - False positive = False Alarm (Type 1 error)
        - False negative = Miss (Type 2 error)
    - $$Percision=\frac{TP}{TP+FP}$$

- $Recall = \frac{TP}{TP+FN}$
- $f-measure = \frac{2 \times Percision \times Recall}{Percision + Recall}$

4. PR Graph, ROC Graph
    - PR Graph: Precision VS Recall
    - ROC Graph:TPR vs FPR
        - $TPR = \frac{TP}{TP+FN}$
        - $FPR = \frac{FP}{FP+TN}$
5. Area Under Curve (AUC)
6. Thresholding
    - Anything below that threshold is class 0, anything above it is class 1

Decision Trees

- Building via ID3: Select highest IG

Kernels

- A function that takes two samples and returns a similarity is called kernel function, $K(X_i, X_j)$

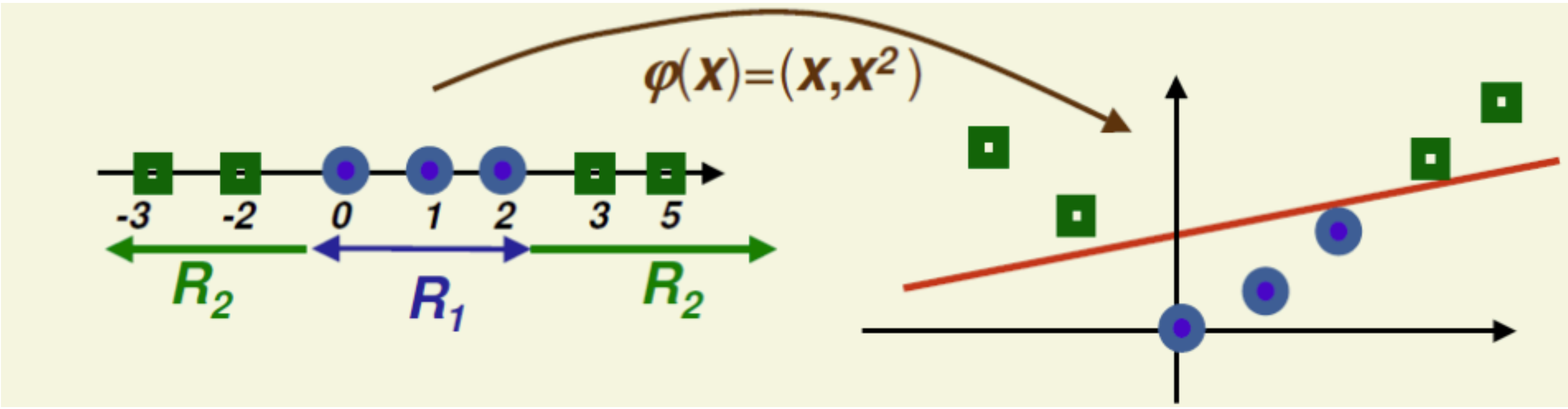
Mapping Functions

- Linear / Cosine: $K(X_i, X_j) = X_i X_j^T$
- Polynomial kernel: $K(X_i, X_j) = (X_i X_j^T + 1)^p$
- Gaussian Radial Basis kernel (RBF): $K(X_i, X_j) = e^{-\frac{\|Xi - Xj\|^2}{2\sigma^2}}$
- Histogram intersection: $K(X_i, X_j) = \sum_{k=1}^{N} min(X_{i,k}, X_{j,k})$

- Hellinger kernel: $K(X_i,X_j)=\sum_{k=1}^{N}\sqrt{X_{i,k}X_{j,k}}$

Kernel Trick

- Sometimes we may want to go to a higher feature space. Because we have a linear classifier and the data is not directly linearly separable.
- One solution would be to map our current space to another separable space. Then project data to **higher** dimension using mapping function $\phi(x)$.
- Using the polynomial kernel of degree two on observations with a single feature is equivalent to compute the cosine similarity on observations in 3D space.



K-Nearest Neighbors

- Idea: Assign label to x according to the label of the training example nearest $x' \in trainingset$

Support Vector Machines

- Maximize distance to closest example (of each type)

Intuition

- We want hyperplane as far as possible from any sample
- New samples close to old samples will then be classified correctly
- Our goal is to maximize the margin
  - The margin is twice the absolute value of distance b of the closest example to the hyperplane

Logistic Regression

- Logistic Regression is not regression. It’s **classification**!
- sigmoid or logistic function $g_\theta(x) = \frac{1}{1+e^{-x\theta}}$.
  - $P(y=1|x,\theta) = g_\theta(x)$
  - $P(y=0|x,\theta) = 1 - g_\theta(x)$

Model form and how to use

logistic function 用一遍

Maximum Log Likelihood Estimate approach (MLE)

- $l(Y|X,\theta) = \sum_{t=1}^{N} ln(g_\theta(X_t))(1-Y_t)ln(1-g_\theta(X_t))$
- Ideally we’d like to take the derivative of this with respect to θ, set it equal to zero, and solve for θ to find the maxima
  - The closed form approach
  - But this isn’t easy
- So what’s our other approach
  - Do partial derivatives on the parameters and use gradient descent! (actually in this case gradient ascent, since we’re trying to maximize)

Gradient Ascent derivation

- $\frac{\partial}{\partial \theta_j} l(y|x,\theta) = (y - g_\theta(x))x_j$
- We want this to go towards zero (local maxima)
- So, update θ_j as
  - $\theta_j := \theta_j \eta \frac{\partial}{\partial \theta_j} l(y|x,\theta)$
  - $\theta_j = \theta_j \eta (y - g_\theta(x))x_j$

Artificial Neural Networks

Computing Forward Propagation

```
hidden = 1 ./ ( 1 + exp(-1 .* data * beta) );
output = 1 ./ ( 1 + exp(-1 .* hidden * theta) );
```

Performing Backwards Propagation

```
delta_out = correctValue - output;
theta = theta + (eta/N) .* (hidden' * delta_out);
delta_hid = (delta_out * theta') .* hidden .* (1 - hidden);
beta = beta + (eta/N) .* (data' * delta_hid);
```

Derivation of back propagation rules

```

```

Deep Learning

- Same idea as regular ANNs but with additional hidden layers.

Multi-Layer Intuition

- Output layer – Here predicting a supervised target
- Hidden layers – These learn more abstract representations as you head up
- Input layer – Raw sensory inputs (roughly)

Issues with multi-layers

- First layer learns 1st order features (e.g. edges, etc..)
- 2^{nd} layer learns high order features (combinations of first layer features, combination of edges, etc..)
- Then final layer features are fed into supervised layer(s)

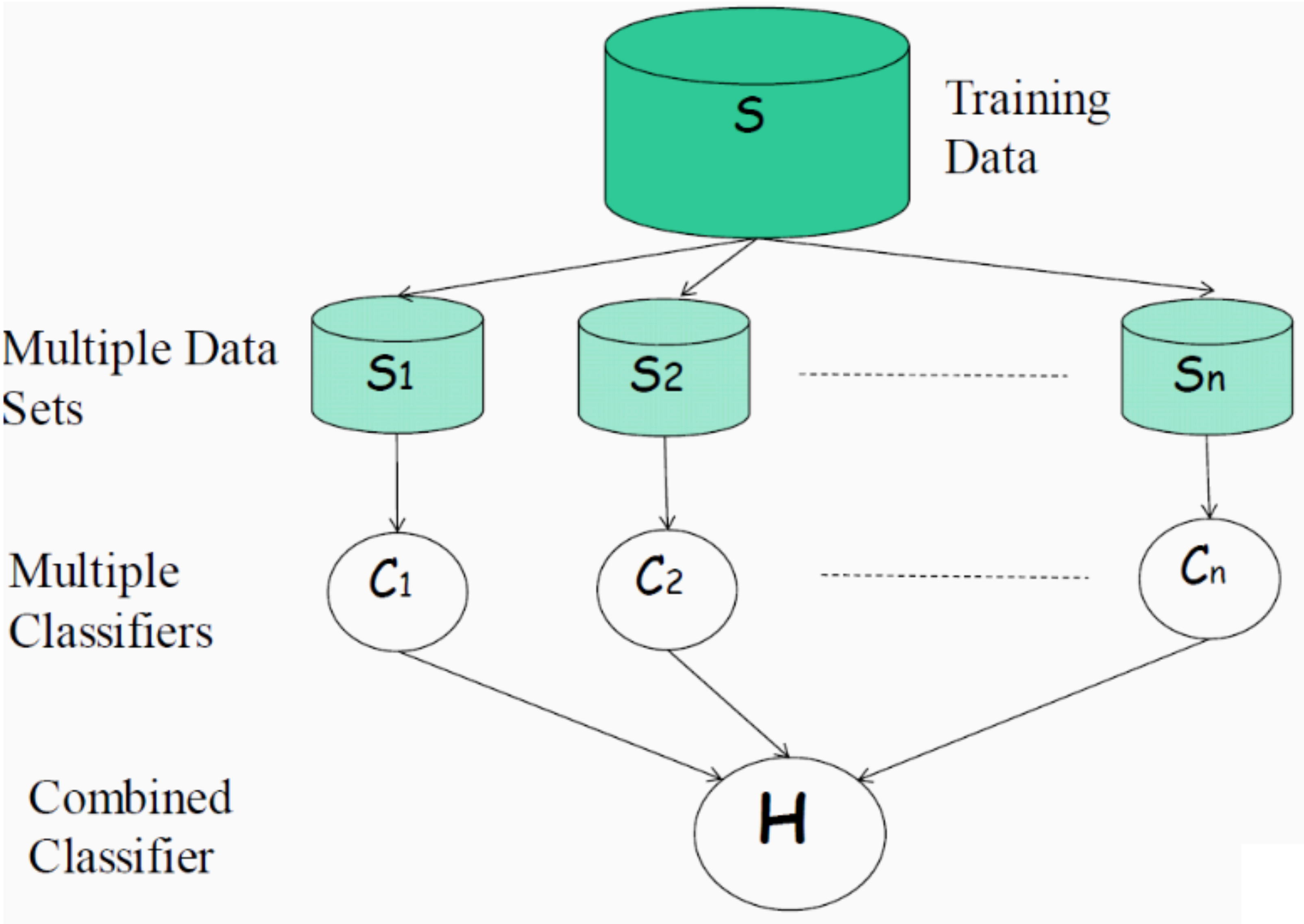
Auto-Encoders

- Attempts to find a hidden layer that can reproduce the input
- Basic process to get a hidden layer from one auto-encoder is:
  1. Take the input, add some noise to it, and add a bias node
  2. Choose the hidden layer size to be less than the input size
  3. The output layer should be the same size as the input (minus the bias node)
  4. Train this auto-encoder using the uncorrupted data as the desired output values.
  5. After training, remove the output layer (and its weights). Now you have your hidden layer to act as the input to the next layer!
- Stacked auto-encoders
  - Do supervised training on last layer
  - Then do supervised training on whole network to fine tune the weights

---

Ensemble Learning

Basic idea: Build different “experts” and let them collaborate to come up with a final decision.



Intuition

- Advantages:
  - Improve predictive performance
  - Different types of classifiers can be directly included
  - Easy to implement
  - Not too much parameter tuning (other than that of the individual classifiers themselves)
- Disadvantages
  - Not compact
  - Combine classifier not intuitive to interpret

Voting

- Classification: Given unseen sample x
  - Each classifier c_j returns the probability that x belongs to class i =1, ...C as $P_{ji}(x)$
    - Or if they can't return probabilities, they will return $P_{ji}(x) \in \{0,1\}$

- Decide how to combine these "votes" to get a value (probability) for each class y_i, and make final decision.

- How to combine the opinions of classifiers

Mean:	$\widehat{y_k} = \frac{1}{T}\sum_{j=1}^{T} P_{jk}$
Weighted Mean:	$\widehat{y_k} = \sum_{j=1}^{T} \alpha_j P_{jk}$ **where** $\sum_j \alpha_j = 1$
Median :	$\widehat{y_k} = \mathrm{median_j}(P_{jk})$
Minimum:	$\widehat{y_k} = \min_{\mathrm{j}}(P_{\mathrm{j}k})$
Maximum:	$\widehat{y_k} = \max_{\mathrm{j}}\left(P_{jk}\right)$
Product:	$\widehat{y_k} = \prod_j P_{jk}$

Bagging

Boosting (Adaboost algorithm)

Random Forests

Intuition

Building