Role of Computer science: Efficient algorithms to solve the optimization problem
- Representing and evaluating the model for inference

What We Talk About When We Talk About “Learning”
- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:
  People who bought “Blink” also bought “Outliers” (www.amazon.com)
- Build a model that is a good and useful approximation to the data.

Data Mining
- Retail: Market basket analysis, Customer relationship management (CRM)
- Finance: Credit scoring, fraud detection
- Manufacturing: Control, robotics, troubleshooting
- Medicine: Medical diagnosis
- Telecommunications: Spam filters, intrusion detection
- Bioinformatics: Motifs, alignment
- Web mining: Search engines

Classification
- Example: Credit scoring
- Differentiating between low-risk and high-risk customers from their income and savings

Regression
- Example: Price of a used car
- x: car attributes
- y: price
- \( y = g(x \mid \theta) \)
- \( g \mid \) model, \( \theta \) parameters

Face Recognition
- Training examples of a person
- Test images

Regression Applications
- Navigating a car: Angle of the steering
- Kinematics of a robot arm
- Response surface design
Learning “what normally happens”

Neural Networks

Output:

Robot in a maze

No output

Journals on Statistics/Data Mining/Signal Processing/Natural Language Processing/Bioinformatics...

Learning a Class from Examples

Class C of “family car”

Positive (+) and negative (–) examples

Input representation: 

\( x_1: \text{price}, x_2 : \text{engine power} \)

Error of \( h \) on \( H \)

\( \text{Error of } h \text{ on } H = \sum (h(x) \neq r(x)) \)

Class \( C \)

\( (x_1 \leq \text{price} \leq x_2) \) AND \( (x_1 \leq \text{engine power} \leq x_2) \)

Training set \( X \)

\( X = \{x_i \} \)

\[ r(x) = \begin{cases} 
1 & \text{if } x \text{ is positive} \\
0 & \text{if } x \text{ is negative} 
\end{cases} \]

Hypothesis class \( H \)

\( \{ h \in H \text{ is positive if } h \text{ says } x \text{ is positive} \} \)

Probably Approximately Correct (PAC) Learning

\( N \) points can be labeled in \( 2^N \) ways as +/- 

\( H \) shatters \( N \) if there exists \( h \in H \) consistent for any of these: 

\( \text{VC}(H) = N \)

VC Dimension

\( \text{How many training examples } N \text{ should we have, such that with } \epsilon \text{ and } \delta? \)

\( (1 - \delta) \text{ is error at most } \epsilon \) 

\( \text{Blumer et al., 1989} \)

Resources: Journals

- Journal of Machine Learning Research
- Machine Learning
- Neural Computation
- Neural Networks
- IEEE Trans on Neural Networks and Learning Systems
- IEEE Trans on Pattern Analysis and Machine Intelligence
- Journals on Statistics/Data Mining/Signal Processing/Natural Language Processing/Bioinformatics...

Resources: Datasets

- Statlib: http://lib.stat.cmu.edu

Resources: Conferences

- International Conference on Machine Learning (ICML)
- European Conference on Machine Learning (ECML)
- Neural Information Processing Systems (NIPS)
- Uncertainty in Artificial Intelligence (UAI)
- Computational Learning Theory (COLT)
- International Conference on Artificial Neural Networks (ICANN)
- International Conference on AI & Statistics (AISTATS)
- International Conference on Pattern Recognition (ICPR)

Learning a policy: A sequence of outputs

- No supervised output but delayed reward
- Credit assignment problem
- Game playing
- Robot in a maze
- Multiple agents, partial observability, ...

Unsupervised Learning

- Learning “what normally happens”
- No output
- Clustering: Grouping similar instances
- Example applications
  - Customer segmentation in CRM
  - Image compression: Color quantization
  - Bioinformatics: Learning motifs

Reinforcement Learning

- Learning a policy: A sequence of outputs
- No supervised output but delayed reward
- Credit assignment problem
- Game playing
- Robot in a maze
- Multiple agents, partial observability, ...

Resources: Conferences

- International Conference on Machine Learning (ICML)
- European Conference on Machine Learning (ECML)
- Neural Information Processing Systems (NIPS)
- Uncertainty in Artificial Intelligence (UAI)
- Computational Learning Theory (COLT)
- International Conference on Artificial Neural Networks (ICANN)
- International Conference on AI & Statistics (AISTATS)
- International Conference on Pattern Recognition (ICPR)

Learning Theory (COLT)

- Journals on Statistics/Data Mining/Signal Processing/Natural Language Processing/Bioinformatics/...
Noise and Model Complexity

Use the simpler one because
- Simpler to use (lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)
- Generalizes better (lower generalization error)

Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
  1. Complexity of $H \in \mathcal{H}_i$
  2. Training set size, $N_i$
  3. Generalization error, $E$, on new data
- As $N_i \uparrow$, $E \downarrow$
- As $c(\mathcal{H}_i)$, first $E \downarrow$ and then $E \uparrow$

Cross-Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
  - Test (publication) set (25%)
- Resampling when there is few data

Dimensions of a Supervised Learner

1. Model: $g(x, \theta)$
2. Loss function: $E(\theta | X) = \sum_r I(r', \arg\max_k g(x, \theta) )$
3. Optimization procedure: $\theta^* = \arg\min_{\theta} E(\theta | X)$

Bayes’ Rule

For any hypothesis $C_i$, state $x_i$,
$$
\text{prior} \quad \Rightarrow \quad \text{likelihood} \quad \Rightarrow \quad \text{posterior}
$$
$$
\begin{align*}
P(C_i | x_i) & \propto P(x_i | C_i) P(C_i) \\
& \propto P(x_i | C_i) \sum_{j \in K} P(C_j)
\end{align*}
$$
$$
\begin{align*}
P(C_i | x_i) & = \frac{P(x_i | C_i) P(C_i)}{\sum_{i \in K} P(x_i | C_i) P(C_i)} \\
P(C = 0 | x_i) + P(C = 1 | x_i) = 1 \\
P(C = 0 | x_i) = P(\bar{C} | x_i) \\
P(C = 1 | x_i) = P(C | x_i)
\end{align*}
$$

Bayes’ Rule: $K \geq 2$ Classes

$$
P(C_i | x_i) \propto P(x_i | C_i) P(C_i) \\
\sum_{i \in K} P(x_i | C_i) P(C_i) \\
P(C_i \geq 0 \text{ and } \sum_{i \in K} P(C_i) = 1)
$$

choose $C$, if $P(C_i | x_i) = \max_i P(C_i | x_i)$

Losses and Risks

- Actions: $a_i$
- Loss of $a_i$ when the state is $C_i$: $\lambda_i$
- Expected risk (Duda and Hart, 1973)
$$
\rho(a_i | x_i) = \sum_{i \in K} P(C_i | x_i) \\
\rho(a_i | x_i) = \rho(a_i | x_i)
$$

For minimum risk, choose the most probable class

Losses and Risks: $0/1$ Loss

$$
\lambda_i = \begin{cases} 
0 & \text{if } i = k \\
1 & \text{if } i \neq k
\end{cases}
$$

$$
R(a_i | x_i) = \sum_{i \in K} \lambda_i \rho(C_i | x_i) \\
R(a_i | x_i) = \sum_{i \in K} \rho(C_i | x_i)
$$

For minimum risk, choose the most probable class
Losses and Risks: Reject

- If $X \rightarrow Y$, a 3-item set, to be frequent (have enough support), $(X,Y)$, $(X,Z)$, and $(Y,Z)$ should be frequent.
- If $(X,Y)$ is not frequent, none of its supersets can be frequent.
- Once we find the frequent k-item sets, we convert them to rules: $X \rightarrow Y$, $Y \rightarrow Z$, ...
  and $X \rightarrow Y$, $Z$, ...

Apriori algorithm (Agrawal et al., 1996)

For $(X,Y,Z)$, a 3-item set, to be frequent (have enough support), $(X,Y)$, $(X,Z)$, and $(Y,Z)$ should be frequent.

Utility Theory

- Prob of state $k$ given evidence $x$: $P(S_k | x)$
- Utility of $\alpha_i$ when state is $k$: $U_{ik}$
- Expected utility: $\max_{k \in K} [U_{ik} | x]$

Association measures

- Support $(X \rightarrow Y)$:
  $\frac{\text{# [customers who bought $X$ and $Y$]}}{\text{# [customers]}}$
- Confidence $(X \rightarrow Y)$:
  $\frac{\text{# [customers who bought $X$ and $Y$]}}{\text{# [customers who bought $X$]}}$
- Lift $(X \rightarrow Y)$:
  $\frac{\text{# [customers who bought $X$ and $Y$]}}{\text{# [customers who bought $X$]}}$

Example:

Transaction | Items in basket
---|---
1 | milk, bananas, chocolate, milk, chocolate
2 | milk, bananas, milk
3 | milk, bananas, orange, milk
4 | chocolate
5 | milk, chocolate, chocolate
6 | milk, chocolate

SOLUTION:
- Support = 2/6, Confidence = 2/6
- Support = 3/6, Confidence = 3/6
- Support = 3/6, Confidence = 3/6

Parametric Estimation

- $X = \{ x^i \}$, where $x^i \sim p(x)$
- Parametric estimation: Assume a form for $p(x | \theta)$ and estimate $\theta$, its sufficient statistics, using $X$
  e.g., $N(\mu, \sigma^2)$ where $\theta = \{ \mu, \sigma^2 \}$

Maximum Likelihood Estimation

- Likelihood of $\theta$ given the sample $X$:
  $l(\theta | X) = p(X | \theta) = \prod_i p(x^i | \theta)$
- Log likelihood:
  $L(\theta; X) = \log l(\theta | X) = \sum_i \log p(x^i | \theta)$
- Maximum likelihood estimator (MLE)
  $\hat{\theta'} = \arg \max_{\theta'} L(\theta; X)$

Examples: Bernoulli/Multinomial

- Bernoulli: Two states, failure/success, $x \in \{0, 1\}$
  $p(x) = p_c^{x} (1 - p_c)^{1-x}$
  $L(p_c | X) = \log \prod_x p_c^{x} (1 - p_c)^{1-x}$
  MLE $p_c = \frac{\sum x}{N}$
- Multinomial: $K > 2$ states, $x_i \in \{0, 1\}$
  $p(x_1, x_2, \ldots, x_K) = \prod_i x_i^{p_i}$
  $L(p_1, p_2, \ldots, p_K | X) = \log \prod_i x_i^{p_i}$
  MLE $p_i = \frac{\sum x_i}{N}$

Gaussian (Normal) Distribution

- $p(x) \sim N(\mu, \sigma^2)$
  $p(x) = \frac{1}{\sqrt{2\pi} \sigma} \exp \left[ -\frac{(x - \mu)^2}{2 \sigma^2} \right]$
- MLE for $\mu$ and $\sigma^2$.
Bias and Variance

- Unknown parameter $\theta$
- Estimator $\hat{\theta}_i = \hat{f}(x|\theta)$ on sample $X_i$
- Bias: $b_i(\hat{\theta}) = E[\hat{f}(x|\theta) - \theta]$
- Variance: $V[\hat{f}(x|\theta) - \theta]$ = $E[(\hat{f}(x|\theta) - \theta)^2]$
- Mean square error: $E[|\hat{f}(x|\theta) - \theta|^2]$
- $b_i(\hat{\theta}) = E[\hat{f}(x|\theta) - \theta]$ = $E[\hat{f}(x|\theta) - \theta]$ = $E[(\hat{f}(x|\theta) - \theta)^2]$
- $b_i(\hat{\theta}) = E[\hat{f}(x|\theta) - \theta]$ = $E[\hat{f}(x|\theta) - \theta]$ = $E[(\hat{f}(x|\theta) - \theta)^2]$

Bayes' Estimator

- Treat $\theta$ as a random var with prior $p(\theta)$
- Bayes' rule: $p(\theta|X) = p_0(\theta) p(X|\theta)$
- Full: $p(x|\theta) = \int p(x|\theta)p(\theta) d\theta$
- Maximum a Posterior (MAP): $\theta_{MAP} = \text{argmax}_\theta p_0(\theta|X)$
- Maximum likelihood (ML): $\theta_{ML} = \text{argmax}_\theta p(x|\theta)$
- Bayes' $\hat{\theta}_{Bayes} = E[\theta|X] = \int \theta p(\theta|X) d\theta$

Bias and Variance: Example

- $x^i = \mathcal{N}(\mu, \sigma^2)$ and $\theta = \mathcal{N}(\mu, \sigma^2)$
- $\theta_{MAP} = m$
- $\theta_{MAP} = \text{argmax}_\theta p_0(\theta|X)$
- $\theta_{MAP} = \text{argmax}_\theta p(x|\theta)$
- $\beta(\theta) = E[\theta|X] = \int \theta p(\theta|X) d\theta$

Bias/ Variance Dilemma

- $M$ samples $X_i = [x_i^1, x_i^2, \ldots, x_i^n]$, $i = 1, \ldots, M$
- are used to fit $g(x_i)$, $i = 1, \ldots, M$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\text{Var}(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
- $\beta(g) = \frac{1}{M} \sum_{i=1}^M [g(x_i) - \theta]^2$
CHAPTER 5: MULTIVARIATE METHODS

Multivariate Data
- Multiple measurements (sensors)
- d inputs/features/attributes: d-variate
- N instances/observations/examples

Multivariate Parameters
- Mean: \( \mu = [\mu_1, \ldots, \mu_d]^T \)
- Covariance: \( \Sigma = \text{Cov}(X, X) \)
- Correlation: \( \text{Corr}(X, X) = \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \)

Parameter Estimation
- Sample mean: \( \bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i \)
- Covariance matrix: \( \Sigma = \frac{1}{N} \sum_{i=1}^{N} (X_i - \bar{X})(X_i - \bar{X})^T \)
- Correlation matrix: \( \chi = \frac{1}{N} \Sigma \)

Estimation of Missing Values
- What to do if certain instances have missing attributes?
- Ignore those instances: not a good idea if the sample is small
- Use “missing” as an attribute: may give information
- Imputation: Fill in the missing value
  - Mean Imputation: Use the most likely value (e.g., mean)
  - Imputation by regression: Predict based on other attributes

Bayesian Model Selection
- Prior on models, \( p(\text{model}) \)
  - \( p(\text{model} | \text{data}) = \frac{p(\text{data} | \text{model}) p(\text{model})}{p(\text{data})} \)
- Regularization, when prior favors simpler models
- Bayes, MAP of the posterior
- Average over a number of models with high posterior (voting, ensembles; Chapter 17)

Model Selection
- Cross-validation: Measure generalization accuracy by testing on data unused during training
- Regularization: Penalize complex models
  - \( \text{E} + \lambda \text{error on data} + \lambda \text{model complexity} \)
  - Akaike’s information criterion (AIC), Bayesian information criterion (BIC)
- Minimum description length (MDL), Kalmogorov complexity, shortest description of data
- Structural risk minimization (SRM)
Independent Inputs: Naive Bayes

- If $x_i$ are independent, off-diagonals of $\Sigma$ are 0.
- Mahalanobis distance reduces to weighted by $1/\sigma_i$ Euclidean distance:
  $$p(x) = \prod_i \frac{1}{\sqrt{2\pi\sigma_i}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}\right)$$

- If variances are also equal, reduces to Euclidean distance.

**Different $S_i$**
- Quadratic discriminant
  $$g_i(x) = -\frac{1}{2} \log |\Sigma_i| - \frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) + \log P(C_i)$$
  where
  $w_i = \frac{1}{2} \Sigma_i^{-1}$
  $w_{m, \Sigma_i} = \frac{1}{2} m_i \Sigma_i^{-1} + \log P(C_i)$

**Diagonal $S$**
- When $x_i$ are independent, $\Sigma$ is diagonal
  $$p(x | C) = \prod_i p(x_i | C_i)$$
  (Naive Bayes’ assumption)
  $$g_i(x) = -\frac{1}{2} \sum_j (x_{ij} - m_{ij})^2 + \log P(C_i)$$

Classify based on weighted Euclidean distance (in $s_i$ units) to the nearest mean.

**Model Selection**

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Covariance matrix</th>
<th>No of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared, Hyperspheric</td>
<td>$S \sim \mathcal{N}$</td>
<td>$1$</td>
</tr>
<tr>
<td>Shared, Diagonal</td>
<td>$S \sim \mathcal{N}$, $S = \sigma^2 I$</td>
<td>$d$</td>
</tr>
<tr>
<td>Shared, Hyperspherical</td>
<td></td>
<td>$d(d+1)/2$</td>
</tr>
<tr>
<td>Different, Hyperspherical</td>
<td></td>
<td>$d(d+1)/2$</td>
</tr>
</tbody>
</table>

- As we increase complexity (less restricted $S$), bias decreases and variance increases.
- Assume simple models (allow some bias) to control variance (regularization).

Parametric Classification

- If $p(x | C_i) \sim N(\mu_i, \Sigma)$
  $$p(x | C_i) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu_i)^T \Sigma^{-1} (x - \mu_i)\right)$$

- Discriminant functions
  $$g_i(x) = \log p(x | C_i) + \log P(C_i)$$
  $$= -\frac{d}{2} \log \sigma_i - \frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) + \log P(C_i)$$

**Common Covariance Matrix $S$**
- Shared common sample covariance $S$
  $$S = \sum_i \hat{p}(C_i) S_i$$

- Discriminant reduces to
  $$g_i(x) = -\frac{1}{2} (x - m_i)^T S^{-1} (x - m_i) + \log P(C_i)$$
  which is a linear discriminant
  $$g_i(x) = w_i^T x + w_{m, \Sigma}$$
  where
  $$w_i = S^{-1} m_i$$
  $$w_{m, \Sigma} = \frac{1}{2} m_i S^{-1} m_i + \log P(C_i)$$

**Diagonal $S$, equal variances**
- Nearest mean classifier: Classify based on Euclidean distance to the nearest mean
  $$g_i(x) = \frac{1}{2} \sum_j (x_{ij} - m_{ij})^2 + \log P(C_i)$$
  $$= -\frac{1}{2} \sum_j (x_{ij} - m_{ij})^2$$

- Each mean can be considered a prototype or template and this is template matching.

Discrete Features

- Binary features: $p_i = p(x_i = 1 | C)$
  if $x_i$ are independent (Naive Bayes’)
  $$p(x | C_i) = \prod_i \left(\frac{p_i}{1-p_i}\right)^{x_i}$$
  the discriminant is linear
  $$g_i(x) = \sum_j x_j \log (p_j) + \log P(C_i)$$
  $$= \sum_j x_j \log (p_j) + \log P(C_i)$$

- Estimated parameters
  $$\hat{p}_i = \frac{\sum x_{ij}}{\sum x_j}$$

- Multinomial ($1$ of $n$) features: $x_j \sim \{1, \ldots, n\}$
  $p_{ij} = p(x_j = 1 | C_i) = p(x_j = n_j | C_i)$
  if $x_i$ are independent
  $$p(x | C_i) = \prod_j x_j^{n_j}$$
  $$g_i(x) = \sum_j x_j \log p_j + \log P(C_i)$$
  $$\hat{p}_i = \frac{\sum x_{ij}}{\sum x_j}$$
Multivariate Regression

\[ r' = g(x') | w_0, w_1, \ldots, w_d \] + \varepsilon

Multivariate linear model:
\[ w_0 + w_1 x_1 + \ldots + w_d x_d \]

\[ E(w_0, w_1, \ldots, w_d | X) = \frac{1}{N} \sum_{i=1}^{N} [r_i - w_0 - w_1 x_{i1} - \ldots - w_d x_{id}] \]

Multivariate polynomial model:
- Define new higher-order variables
  \[ x_1^2, x_1 x_2, \ldots, x_1 x_d, x_2^2, \ldots, x_d^2 \]
- and use the linear model in the new \( x \) space (basis functions, kernel trick: Chapter 1.3)

Why Reduce Dimensionality?
- Reduces time complexity: Less computation
- Reduces space complexity: Fewer parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- More interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc.) if plotted in 2 or 3 dimensions

Feature Selection vs Extraction

Feature selection: Choosing \( k < d \) important features, ignoring the remaining \( d - k \)
- Subset selection algorithms
  - Forward selection: Add the best feature at each step
    - Set of features \( F \) initially Ø.
    - At each iteration, find the best new feature
      - \( \emptyset \cup \{ f \} \)
    - Add \( f \) to \( F \) if \( E(F \cup \{ f \}) < E(\emptyset) \)
  - Backward search: Start with all features and remove one at a time, if possible.
  - Floating search (Add \( f \), remove \( f \))

Principal Components Analysis

Find a low-dimensional space such that when \( x \) is projected there, information loss is minimized.
- The projection of \( x \) on the direction of \( w \) is \( z = \mathbf{w}^T \mathbf{x} \)
- Find \( w \) such that \( \text{Var}(z) \) is maximized
  \[ \text{Var}(z) = \text{Var}(\mathbf{w}^T \mathbf{x}) = \mathbb{E}[(\mathbf{w}^T \mathbf{x} - \mu_w)(\mathbf{w}^T \mathbf{x} - \mu_w)] \]
  \[ = \mathbb{E}[(\mathbf{w}^T \mathbf{x} - \mu_w)(\mathbf{w}^T \mathbf{x} - \mu_w)] \]
  \[ = \mathbb{E}[(\mathbf{w}^T \mathbf{x}) - \mu_w][(\mathbf{w}^T \mathbf{x}) - \mu_w] \]
  \[ = (\mathbf{w}^T \mathbf{x} - \mu_w)^2 \]
  \[ \text{Var}(z) = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
- Second principal component: Max \( \text{Var}(z_2) \), s.t., \( \| \mathbf{w}_2 \| = 1 \) and orthogonal to \( \mathbf{w}_1 \)
  \[ \max \text{Var}(z_2) = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ \sum_{i=2}^{d} \mathbf{w}_i \cdot \mathbf{x}_i - \alpha \mathbf{w}_1 \cdot \mathbf{x}_1 - \beta \mathbf{w}_1 \cdot \mathbf{x}_1 = 0 \]
- Maximizing \( \text{Var}(z) \) subject to \( \| \mathbf{w} \| = 1 \)
  \[ \max \sum_{i=1}^{d} \mathbf{w}_i \cdot \mathbf{x}_i - \alpha \mathbf{w}_1 \cdot \mathbf{x}_1 - \beta \mathbf{w}_1 \cdot \mathbf{x}_1 \]
  \[ \sum_{i=1}^{d} \mathbf{w}_i \cdot \mathbf{x}_i - \alpha \mathbf{w}_1 \cdot \mathbf{x}_1 - \beta \mathbf{w}_1 \cdot \mathbf{x}_1 = 0 \]
  \[ \sum_{i=1}^{d} \mathbf{w}_i \cdot \mathbf{x}_i = \beta \mathbf{w}_1 \cdot \mathbf{x}_1 \]

How to choose \( k \)?
- Proportion of Variance (PoV) explained
  \[ \frac{\lambda_1 + \lambda_2 + \ldots + \lambda_k}{\lambda_1 + \lambda_2 + \ldots + \lambda_d} \]
  when \( \lambda_i \) are sorted in descending order
- Typically, stop at PoV > 0.9
- Scree plots of PoV vs \( k \), stop at “elbow”

Principal Components Analysis

Find a low-dimensional space such that when \( x \) is projected there, information loss is minimized.
- The projection of \( x \) on the direction of \( w \) is \( z = \mathbf{w}^T (x - \mu_x) \)
- Find \( w \) such that \( \text{Var}(z) \) is maximized
  \[ \text{Var}(z) = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]

Principal Components Analysis

Find a low-dimensional space such that when \( x \) is projected there, information loss is minimized.
- The projection of \( x \) on the direction of \( w \) is \( z = \mathbf{w}^T (x - \mu_x) \)
- Find \( w \) such that \( \text{Var}(z) \) is maximized
  \[ \text{Var}(z) = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]
  \[ = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)] \]

Subset Selection

There are 2^d subsets of \( d \) features
- Forward search: Add the best feature at each step
  - Set of features \( F \) initially Ø.
  - At each iteration, find the best new feature
    - \( \emptyset \cup \{ f \} \)
  - Add \( f \) to \( F \) if \( E(F \cup \{ f \}) < E(F) \)
- Hill-climbing \( O(d) \) algorithm
  - Backward search: Start with all features and remove one at a time, if possible.
  - Floating search (Add \( f \), remove \( f \))

Iris data: Single feature

What PCA does

\[ z = \mathbf{W}(x - \mu) \]

where the columns of \( W \) are the eigenvectors of \( \Sigma \)

and \( \mu \) is sample mean

Centers the data at the origin and rotates the axes

Feature Embedding

When \( X \) is the Nxd data matrix,

- \( XX' \) is the dxd matrix (covariance of features, if mean-centered)
- \( \Sigma \) is the Nxd matrix (pairwise similarities of instances)
  - PCA uses the eigenvectors of \( XX' \) which are d-dim and can be used for projection
  - Feature embedding uses the eigenvectors of \( XX' \) which are N-dim and can give directly the coordinates after projection
  - Sometimes, we can define pairwise similarities (or distances) between instances, then we can use feature embedding without needing to represent instances as vectors.
Factor Analysis

- Find a small number of factors $z_i$, which when combined generate $x$:
  $$x_i - E_i = v_1 z_1 + v_2 z_2 + \ldots + v_k z_k + \xi_i$$
  where $z_k$, $i=1,\ldots,k$ are the latent factors with
  $$E(z_k) = 0, \operatorname{Var}(z_k) = 1, \operatorname{Cov}(z_i, z_j) = 0, i \neq j,$$
  $\xi_i$ are the noise sources.

- Find $z_i$ that maximizes the class scatter:
  $$\sum_{i,j} (x_i - \mu_i)(x_j - \mu_j)$$

- Between-class scatter:
  $$\sum_{i=1}^K \sum_{j=1}^K (x_i - \mu_i)(x_j - \mu_j)$$

- Within-class scatter:
  $$\sum_{i=1}^K \sum_{j=1}^K (x_i - \mu_i)(x_j - \mu_i)$$

PCA vs FA

- PCA: From $x$ to $z$:
  $$z = W(x - \mu)$$

- FA: From $z$ to $x$:
  $$x = \mu + Vz + \xi$$

Factor Analysis

In FA, factors $z_i$ are stretched, rotated and translated to generate $x$.

Singular Value Decomposition and Matrix Factorization

- Singular value decomposition: $X = U A V^T$
  $U$ is $N \times N$ and contains the eigenvectors of $XX^T$
  $V$ is $k \times d$ and contains singular values on its first $k$ diagonal.

- $X = w_1 a_1 v_1^T + \ldots + w_k a_k v_k^T$ where $k$ is the rank of $X$.

Matrix Factorization

- Matrix factorization: $X = FG$
  $F$ is $N \times k$ and $G$ is $k \times d$.

- Latent semantic indexing.

Multidimensional Scaling

- Given pairwise distances between $N$ points, $d_{ij}$, find a low-dim map such that distances are preserved:
  $$d_w(i,j) = \left(\sum_{r=1}^k (w_{ir} - w_{jr})^2\right)^{1/2}$$

- Find $w$ that minimizes Sammon stress:
  $$E(w) = \frac{1}{n^2} \sum_{i \neq j} \frac{(d_i - d_j)}{d_i} \left(\frac{d_{ij}}{d_i + d_j}\right)^2$$

- Monotone $z$ that minimizes the sum of squares:
  $$z = g(x \mid \theta)$$

Linear Discriminant Analysis

- Find a low-dimensional space such that when $x$ is projected, classes are well-separated.

- Find $w$ that maximizes
  $$J(w) = \frac{m_r - m_i^2}{s_r^2}$$

- $m_r = \sum_{i \neq j} w_i x_i \left(\frac{x_j - m_j}{s_j}\right)$

PCA vs LDA

- $X = \{x^T, y^T\}$; two sets of variables $x$ and $y$.

- We want to find two projections $w$ and $v$ so when $x$ is projected along $w$ and $y$ is projected along $v$, the correlation is maximized:
  $$r_{xy} = \frac{\text{Corr}(w^T x, v^T y)}{\text{Var}(w^T x) \text{Var}(v^T y)} = \frac{w^T \text{Cov}(x, y) v}{\sqrt{w^T \text{Var}(x) v} \sqrt{v^T \text{Var}(y) v}}$$

CCA

- $x$ and $y$ may be different views of the same data; e.g., image and word tags, and CCA does a joint mapping.

Canonical Correlation Analysis

- CCA: $x$ and $y$ may be two different views or modalities; e.g., image and word tags, and CCA does a joint mapping.

Isomap

- Geodesic distance is the distance along the manifold that the data lies in, as opposed to the Euclidean distance in the input space.

Fisher's Linear Discriminant

- Find $w$ that maximizes
  $$J(w) = \frac{w^T S_b w}{w^T S_w w}$$

- $S_b = \sum_{i=1}^K S_i$, $S_i = \sum_{x \in C_i}(x^T - m^T) (x^T - m)^T$

- Within-class scatter:
  $$S_w = \sum_{i=1}^K (x^T - m^T)(x^T - m)^T$$

- Between-class scatter:
  $$S_b = \sum_{i=1}^K (x^T - m^T)(x^T - m)^T$$

- Find $W$ that maximized $J(W) = \frac{W^T S_b W}{W^T S_w W}$

The largest eigenvectors of $S_b^{-1} S_w$ maximum rank of $K-1$
Instances \( r \) and \( s \) are connected in the graph if
\[ ||x_r - x_s|| < e \text{ or if } x_s \text{ is one of the } k \text{ neighbors of } x_r \]
The edge length is \(| ||x_r - x_s|| | \).
- For two nodes \( r \) and \( s \) not connected, the distance is equal to the shortest path between them.
- Once the \( N \times N \) distance matrix is thus formed, use MDS to find a lower-dimensional mapping.

**Isomap**

- \( N \times N \) distance matrix is formed.
- Use MDS to find a lower-dimensional mapping.

**LLE on Optdigits**

- For two instances \( r \) and \( s \), their similarity is \( B_{rs} \).
- \( B_{rs} \) can be defined in terms of similarity in an original space: 0 if \( x_r \) and \( x_s \) are too far, otherwise.
- \( R = \exp \left[ -\frac{||x_r - x_s||^2}{2\sigma^2} \right] \)
- Defines a graph Laplacian, and feature embedding returns \( z_r \)

- \( \min \sum_{ij} (z_i - z_j)^2 B_{ij} \)
- \( B_{rs} \) is their similarity, we want to find \( z_r \) and \( z_s \) that minimize \( \sum_{ij} ||z_i - z_j||^2 B_{ij} \)

**Laplacian Eigenmaps**

- Semiparametric density estimation
- Encoding/Decoding

**Locally Linear Embedding**

- Given \( x \) find its neighbors \( x_{ij} \)
- Find \( W \) that minimize \( \mathbb{E}(W | x) = \sum_{ij} -\sum W_{ij} x_{ij} \)
- Use nearest (most similar) reference:

\[ f(x) = \sum_{ij} W_{ij} x_{ij} \]

- Reconstruction error:
\[ E(m,r) = \sum_{ij} W_{ij} ||x_i - m_i||^2 \]
\[ E(m,r) = \sum_{ij} W_{ij} ||x_i - m_i||^2 \]

**Mixture Densities**

- Parametric: Assume a single model for \( \rho(x | \mathcal{C}) \)
- Semiparametric: \( \rho(x | \mathcal{C}) \) is a mixture of densities
- Multiple possible explanations/prototypes:
- Different handwriting styles, accents in speech
- Nonparametric: No model; data speaks for itself

- Gaussian mixture where \( \rho(x | G_i) = \mathcal{N}(\mu, \Sigma) \)
- Parameters \( \Phi = \{ (G, \mu, \Sigma) \}_{i=1}^k \)
- Unlabeled sample \( X = [x_i] \) (unsupervised learning)
In classification, the input comes from a mixture of classes.

If each class is also a mixture, e.g., of Gaussians, we have a mixture of mixtures:

\[
p(x) = \sum_{i} \sum_{j} p(x | G_i, G_j)
\]

\[
p(x) = \sum_{i} p(x | C_i) p(C_i)
\]

In clustering, the input comes from a mixture of clusters.

If each cluster is also a mixture, e.g., of Gaussians, we have a mixture of mixtures:

\[
p(x) = \sum_{i} \sum_{j} p(x | Z_i, Z_j)
\]

\[
p(x) = \sum_{i} p(x | Z_i) p(Z_i)
\]

In classification, the input comes from a mixture of classes.

If each class is also a mixture, e.g., of Gaussians, we have a mixture of mixtures:

\[
p(x) = \sum_{i} \sum_{j} p(x | G_i, G_j)
\]

\[
p(x) = \sum_{i} p(x | C_i) p(C_i)
\]
Nonparametric Estimation

- Parametric (single global model), semiparametric (small number of local models)
- Nonparametric Similar inputs have similar outputs
- Functions (pdf, discriminant, regression) change smoothly
- Keep the training data; let the data speak for itself
- Given $x$, find a small number of closest training instances and interpolate from these
- Aka lazy/memory-based/case-based/instance-based learning

Kernel Estimator

- Kernel function, e.g., Gaussian kernel:
  $$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$$
- Kernel estimator (Parzen windows)
  $$\hat{p}(x) = \frac{1}{Nh} \sum_{t=1}^{N} \frac{1}{h} K\left(\frac{x-x'}{h}\right)$$
- Multivariate Gaussian kernel
  - Spherical
  $$K(u) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2} u^T u\right)$$
  - Ellipsoid
  $$K(u) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2} u^T S u\right)$$

Density Estimation

- Given the training set $X = \{x_i\}$, drawn iid from $p(x)$
- Divide data into bins of size $h$
- Histogram
  $$\hat{p}(x) = \frac{\# \{x \in X \mid x \in [x-h, x+h]\}}{Nh}$$
- Naïve estimators
  $$\hat{p}(x) = \frac{1}{Nh} \sum_{i=1}^{Nh} w(u) = \begin{cases} \frac{1}{Nh} & \text{if } |i| = 1 \\ 0 & \text{otherwise} \end{cases}$$

k-Nearest Neighbor Estimator

- Instead of fixing bin width $h$ and counting the number of instances, fix the instances (neighbors) $k$ and check bin width
  $$\hat{p}(x) = \frac{k}{Nh}$$
  $$d_k(x), \text{ distance to } k^{th} \text{ closest instance to } x$$

Nonparametric Classification

- Estimate $p(x \mid C_j)$ and use Bayes’ rule
- Kernel estimator
  $$\hat{p}(x) = \frac{1}{Nh} \sum_{i=1}^{Nh} K\left(\frac{x-x'}{h}\right) = \frac{1}{Nh} \sum_{i=1}^{Nh} \frac{1}{h} K\left(\frac{x-x'}{h}\right)$$
- $k$-NN estimator
  $$\hat{p}(x) = \frac{k}{Nh}$$

Condensed Nearest Neighbor

- Incremental algorithm: Add instance if needed

Distance-based Classification

- Find a distance function $D(x, x')$ such that if $x$ and $x'$ belong to the same class, distance is small and if they belong to different classes, distance is large
- Assume a parametric model and learn its parameters using data, e.g.,
  $$D(x, x') = \frac{1}{2} (x-x')^T M (x-x')$$

Learning a Distance Function

- The three-way relationship between distances, dimensionality reduction, and feature extraction.
  - $M = LL^T$ is $d \times d$ and $L$ is $d \times k$
  - $d(x, x') = \frac{1}{2} (x-x')^T M (x-x') = \frac{1}{2} (Lx-Lx')^T (Lx'-Lx') = \frac{1}{2} (x-x')^T (x-x')$
- Similarity-based representation using similarity scores
- Large-margin nearest neighbor (chapter 13)
Outlier Detection

- Find outlier/novelty points
- Not a two-class problem because outliers are very few, of many types, and seldom labeled
- Instead, one-class classification problem: Find instances that have low probability
- In nonparametric case: Find instances far away from other instances

Local Outlier Factor

Nonparametric Regression

- Aka smoothing models
- Regressogram
  \[ \hat{g}(x) = \frac{\sum w_i(x)h_i}{\sum w_i} \]
  where
  \[ w_i(x) = \begin{cases} 1 & \text{if } x_i \text{ is in the same bin with } x \\ 0 & \text{otherwise} \end{cases} \]

Running Mean/Kernel Smoother

- Running mean smoother
  \[ \bar{y}(x) = \frac{\sum_{i=1}^{n} (x_i \cdot y_i)^r}{\sum_{i=1}^{n} (x_i)^r} \]
  where \( K(r) \) is Gaussian
- Running line smoother
- Kernel smoother
  \[ \hat{g}(x) = \frac{\sum_{i=1}^{n} w_i(x_i)h_i}{\sum_{i=1}^{n} w_i} \]

How to Choose k or h?

- When k or h is small, single instances matter; bias is small, variance is large (undersmoothing): High complexity
- As k or h increases, we average over more instances and variance decreases but bias increases (oversmoothing): Low complexity
- Cross-validation is used to finetune k or h.

Tree Uses Nodes and Leaves

- Internal decision nodes
  - Univariate: Uses a single attribute, \( x_i \)
  - Numeric \( x_i \): Binary split \( x_i > w \)
  - Discrete \( x_i \): n-way split for n possible values
  - Multivariate: Uses all attributes, \( x \)
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric, r average, or local fit
- Learning is greedy: find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)
Classification Trees (ID3, CART, C4.5)

- For node \( m \), \( N_m \) instances reach \( m \), \( N_m \) belong to \( C_i \).
  \[ P(C_i | m) = \frac{N_{m_i}}{N_m} \]
- Node \( m \) is pure if \( P_m = 0 \) or 1.
- Measure of impurity is entropy
  \[ I_m = - \sum_{i=0}^{K} p_i \log p_i \]

Best Split

- If node \( m \) is pure, generate a leaf and stop, otherwise split and continue recursively.
- Impurity after split: \( N_m \) of \( N_m \) take branch \( j \) \( N_m \) belong to \( C_i \).
  \[ P(C_i | m, j) = \frac{N_{m_j}}{N_m} \]
  \[ I_m = - \sum_{i=0}^{K} \sum_{j=0}^{M} P_{m_j} \log P_{m_j} \]
- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)

Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees that overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

Pruning Rules

\[ \text{Rule } \rightarrow \text{Leaf} \]
For each Rule in RuleSet
- If \( \text{Rule} \rightarrow \text{Leaf} \), return RuleSet
- If \( \text{Rule} \rightarrow \text{Rule} \)
  - Update RuleSet
- If RuleSet is empty, return RuleSet

Likelihood- vs. Discriminant-based Classification

- Likelihood-based: Assume a model for \( p(x | C_i) \), use Bayes’ rule to calculate \( P(C_i | x) \)
  \[ g(x) = \log P(C_i | x) \]
- Discriminant-based: Assume a model for \( g_i(x | \Phi) \); no density estimation
- Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries

Linear Discriminant

- Linear discriminant
  \[ g(x | w_i) = w_i^T x + w_0 = \sum_{j=0}^{d} w_j x_j + w_0 \]
- Advantages:
  - Simple: \( O(d) \) space/computation
  - Knowledge extraction: Weighted sum of attributes; positive/negative weights, magnitudes (credit scoring)
  - Optimal when \( p(x | C_i) \) are Gaussian with shared cov matrix; useful when classes are (almost) linearly separable
**Generalized Linear Model**

- Quadratic discriminant:
  \[ g(x; w, w_0) = w^T x + w_0 \]
- Higher-order (product) terms:
  \[ z_i = x_1, z_i = x_2, z_i = x_3, z_i = x_4, z_i = x_5 \]

Map from \( x \) to \( z \) using nonlinear basis functions and use a linear discriminant in a space.

\[ g(x) = \sum_{i=1}^n w_i g_i(x) \]

**Pairwise Separation**

\[ g_i(x) = \begin{cases} 1 & \text{if } x \in C_i \\ 0 & \text{if } x \notin C_i \end{cases} \]

Choose \( C_i \) if
\[ y = \max \left[ \sum_{i=1}^n y_i, \sum_{i=1}^n y_j \right] \]

**From Discriminants to Posteriors**

When \( P(x | C_i) \sim N(\mu_i, \Sigma) \)
\[ g_i(x; w) = w^T x + w_0 \]
\[ w_i = \Sigma^{-1} \mu_i + \log P(C_i) \]
\[ y = P(C_i | x) \text{ and } P(C_j | x) = 1 - y \]

Choose \( C_i \) if
\[ y / (1 - y) > 1 \quad \text{and } C_i \text{ otherwise} \]

\[ \log \left[ \frac{P(C_i | x)}{P(C_j | x)} \right] = \log \frac{w_i^T x + w_0}{w_j^T x + w_0} \]

The inverse of logit
\[ \log \left[ \frac{P(C_i | x)}{P(C_j | x)} \right] = \frac{1}{1 + \exp[-w_i^T x - w_0]} \]

**Gradient Descent**

- \( E(w | X) \) is error with parameters \( w \) on sample \( X \)
  \[ w^* = \arg \min_{w} E(w | X) \]
- Gradient
  \[ \nabla E = \frac{\partial E}{\partial w_w} \]
- Gradient descent:
  Starts from random \( w \) and updates \( w \) iteratively in the negative direction of gradient

**Training: Gradient Descent**

- \( E(w, w_0 | X) = \sum r \log y + (1 - r) \log (1 - y) \)
- If \( y = \text{sigmoid} \) \( \frac{\partial E}{\partial w} = y(1 - y) \)
- \( \Delta w = -\eta \frac{\partial E}{\partial w} \)

**K>2 Classes**

- \( X = [x, p^T] \quad r^T x = \text{Boumoul}(p) \)
- \( y = \hat{P}(C_i | x) = \frac{\exp[w_i^T x + w_0]}{\sum_{i=1}^K \exp[w_i^T x + w_0]} \)
- \( E = -\log \frac{\exp[w_i^T x + w_0]}{\sum_{i=1}^K \exp[w_i^T x + w_0]} \)
- \( \Delta w_j = \eta \sum_{r \neq y} (r - y)^2 x_j \)

**Logistic Discrimination**

- Two classes: Assume log likelihood ratio is linear
  \[ \frac{\log P(C_1 | x)}{\log P(C_2 | x)} = w^T x + w_0 \]
  \[ y = \hat{P}(C_i | x) = \frac{\exp[w_i^T x + w_0]}{\sum_{i=1}^2 \exp[w_i^T x + w_0]} \]
  \[ E = -\log \frac{\exp[w_i^T x + w_0]}{\sum_{i=1}^2 \exp[w_i^T x + w_0]} \]
  \[ \Delta w_j = \eta \sum_{r \neq y} (r - y)^2 x_j \]
Learning to Rank

- Ranking: A different problem than classification or regression
- Let us say x* and x' are two instances, e.g., two movies.
- We prefer u to v implies that g(x*) > g(x')
- where g(x) is a score function, here linear: g(x) = w*x
- Find a direction w such that we get the desired ranks when instances are projected along w.

Example

Generalizing the Linear Model

- Quadratic:
  \[
  \log \frac{p(x|1)}{p(x|0)} = \mathbf{w}^T \mathbf{x} + \mathbf{w}_0
  \]
- Sum of basis functions:
  \[
  \log \frac{p(x|C)}{p(x|\bar{C})} = \sum_i w_i \phi_i(x) + w_0
  \]
- where \(\phi_i(x)\) are basis functions. Examples:
- Hidden units in neural networks (Chapters 11 and 12)
- Kernels in SVM (Chapter 13)

Discrimination by Regression

- Classes are not mutually exclusive and exhaustive
- \(r' = y' + c\) where \(-N(x|0,a)\)
- \(y' = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + \mathbf{w}_0)\)

Neural Networks

- Networks of processing units (neurons) with connections (synapses) between them
- Large number of neurons: \(10^{10}\)
- Large connectivity: \(10^5\)
- Parallel processing
- Distributed computation/memory
- Robust to noise, failures

Understanding the Brain

- Levels of analysis (Marr, 1982)
- 1. Computational theory
- 2. Representation and algorithm
- 3. Hardware implementation
- Reverse engineering: From hardware to theory
- Parallel processing: SIMD vs MIMD
- Neural net: SIMD with modifiable local memory
- Learning: Update by training/experience

Perceptron

- \(y = \sum_{i=1}^{K} w_i x_i + w_0 = \mathbf{w}^T \mathbf{x}\)
- \(\mathbf{w} = [w_1, w_2, \ldots, w_K]\)
- \(\mathbf{x} = [x_1, x_2, \ldots, x_K]\) (Rosenblatt, 1962)

What a Perceptron Does

- Regressor: \(y = w_0 x + w_0\)
- Classification: \(y = 1(xw + w_0 > 0)\)

K Outputs

- \(y = \mathbf{w}^T \mathbf{x}\)
- \(y_i = \sum w_i x_i + w_0 = \mathbf{w}^T \mathbf{x} \implies \mathbf{y} = \mathbf{W} \mathbf{x}\)

Training

- Online (instances seen one by one) vs batch (whole sample) learning.
- No need to store the whole sample.
- Problem may change in time.
- Wear and degradation in system components.
- Stochastic gradient-descent: Update after a single pattern.
- Generic update rule (LMS rule):
- \(\Delta \mathbf{w}_i = \eta (d_i - y_i') y_i\)
Training a Perceptron: Regression

- Regression (Linear output):
  \[ E(W, v \mid x) = \frac{1}{2} \sum (y - y_l)^2 \]
  \[ \Delta w_i = \eta (y_i - y_l) x_i \]

Multilayer Perceptrons

- Single sigmoid output
  \[ y = \text{sigmoid}(w^T x) \]
  \[ E(W, v \mid x) = \frac{1}{2} \sum \log y_i \log (1 - y_i) \]
  \[ \Delta w_i = \eta (y_i - y_l) x_i \]
- K > 2 softmax outputs
  \[ y_i = \frac{\exp w_i^T x}{\sum \exp w_j^T x} \]
  \[ E(W, v \mid x) = -\sum \log y_i \]
  \[ \Delta w_i = \eta (y_i - y_l) x_i \]

Regression with Multiple Outputs

- \[ E(W, v \mid x) = \frac{1}{2} \sum (y - y_l)^2 \]
- \[ y_l = \sum w_i x_i + v_0 \]
- \[ \Delta w_i = \eta (y_i - y_l) x_i \]
- \[ \Delta v_0 = \eta \sum (y - y_l) x_i \]

Two-Class Discrimination

- One sigmoid output \( y' = \text{sigmoid}(\sum x_i' w_i + v_0) \)

Classification

- Single sigmoid output
  \[ y = \text{sigmoid}(w^T x) \]
  \[ E(W, v \mid x) = \frac{1}{2} \sum \log y_i \log (1 - y_i) \]
  \[ \Delta w_i = \eta (y_i - y_l) x_i \]
- \( K > 2 \) softmax outputs
  \[ y_i = \frac{\exp w_i^T x}{\sum \exp w_j^T x} \]
  \[ E(W, v \mid x) = -\sum \log y_i \]
  \[ \Delta w_i = \eta (y_i - y_l) x_i \]

K > 2 Classes

- \[ y_i = \sum w_i' x_i + v_0 \]
- \[ E(W, v \mid x) = -\sum \log e_i \]
  \[ \Delta w_i' = \eta (e_i - e_l) x_i \]

Multiple Hidden Layers

- MLP with one hidden layer is a universal approximator (Hornik et al., 1989), but using multiple layers may lead to simpler networks
- \( z_i = \text{sigmoid}(w_i^T x + v_i) \)
  \[ E(W, v \mid x) = -\sum \log e_i \]
  \[ \Delta w_i = \eta (e_i - e_l) x_i \]

Improving Convergence

- Momentum
  \[ \Delta w_i = -\eta \frac{\partial E}{\partial w_i} + \alpha \Delta w_i^{-1} \]
- Adaptive learning rate
  \[ \eta = \begin{cases} \alpha & \text{if } E^{t+1} < E^t \\ \eta_0 & \text{otherwise} \end{cases} \]
Convolutional networks (Deep learning)

Can be trained layer by layer using an autoencoder

Destructive

Augmented error:

Sequence reproduction: Time-series prediction

Sequence association

Consider weights $w_i$ as random vars, prior $p(w)$

Weight decay, ridge regression, regularization

$p(w | X) = \frac{p(X | w) p(w)}{p(X)}$ $\hat{w}_{MAP} = \arg \max \log p(w | X)$

$\log p(w | X) = \log p(X | w) + \log p(w)$

$p(w) = \prod p(x_i)$ where $p(x_i) = c \exp \left(\frac{-w^2}{2\lambda}\right)$

- Weight decay, ridge regression, regularization
- Cost=cost+misfit + A complexity
- More about Bayesian methods in chapter 1

- Overfitting/Overtraining

$E = \sum_{i}^{H} \lambda w_i$ $\Delta w_i = -\frac{\partial E}{\partial w_i} = -\lambda w_i$

$$E = \sum_{i}^{H} \lambda w_i + \frac{1}{2} \sum_{i}^{H} w_i^2$$

- Tuning the Network Size

Destructive

Weight decay:

Growing networks

Bayesian Learning

Dimensionality Reduction

Applications:
- Sequence recognition: Speech recognition
- Sequence reproduction: Time-series prediction
- Sequence association

Network architectures
- Time-delay networks (Waibel et al., 1989)
- Recurrent networks (Rumelhart et al., 1986)

Deep Networks

- Layers of feature extraction units
- Can have local receptive fields as in convolution networks, or can be fully connected
- Can be trained layer by layer using an autoencoder in an unsupervised manner
- No need to craft the right features or the right basis functions or the right dimensionality reduction method; learns multiple layers of abstraction all by itself given a lot of data and a lot of computation
- Applications in vision, language processing, ...

Applications:
- Convolutional networks (Le Cun et al., 1989)
- Recurrent networks (Rumelhart et al., 1986)
- Time-delay networks (Waibel et al., 1989)
- Recurrent networks (Rumelhart et al., 1986)
Introduction

- Divide the input space into local regions and learn simple (constant/linear) models in each patch

- Unsupervised: Competitive, online clustering
- Supervised: Radial-basis functions, mixture of experts

Adaptive Resonance Theory

- Incremental; add is new cluster if not covered, defined by vigilance
  \[ \rho = \frac{d_{min}}{d_{new}} \]
- If \( d_{new} > \rho \), add otherwise

(Carpenter and Grossberg, 1988)

Self-Organizing Maps

- Units have a neighborhood defined; \( m_i \) is “between” \( m_{i-1} \) and \( m_{i+1} \), and are all updated together
- One-dim map: \( ( \text{Kohonen, 1990} ) \)
  \[ \Delta m_i = \eta (y_i - m_i) \]

Training RBF

- Hybrid learning:
  - First layer centers and spreads: Unsupervised k-means
  - Second layer weights: Supervised gradient-descent
- Fully supervised
  (Broomhead and Lowe, 1988; Moody and Darken, 1989)

Radial-Basis Functions

- Locally-tuned units:
  \( \rho_i = \exp \left[ \frac{||x - m_i||^2}{2\sigma_i^2} \right] \)

Classification

- Mixture model: \( p(x) = \sum \alpha_i p_i(x) \)

Rule-Based Knowledge

- IF \( (x = a) \) AND \( (x = b) \) OR \( (x = c) \) THEN \( y = 0.1 \)

Normalized Basis Functions

- Incorporation of prior knowledge (before training)
- Rule extraction (after training) (Tresp et al., 1997)
- Fuzzy membership functions and fuzzy rules

Competitive Basis Functions

- Mixture model: \( p(x) = \sum \alpha_i p_i(x) \)

Regression

- Radial-basis functions, mixture of experts

Rules and Exceptions

- Default rule
**EM for RBF (Supervised EM)**

- **E-step:**
  \[ f'_i = \frac{p(x | h_i)}{\sum_j p(x | h_j)} \]

- **M-step:**
  \[
  m_i = \sum_j \frac{f'_i}{g'_i} y_j \\
  s_i = \sum_j \frac{f'_i}{g'_i} (y_j - y_{r_i}) (y_j - y_{r_i})' \\
  w_i = \sum_j \frac{f'_i}{g'_i} r_i' \]

**Learning Vector Quantization**

- **H units per class prelabeled (Kohonen, 1990)**
  Given \( x, m_i \) is the closest:
  - \( \Delta m_i = \eta (x' - m_i) \) if label \( (x') = \text{label}(m_i) \)
  - \( \Delta m_i = -\eta (x' - m_i) \) otherwise

**Hierarchical Mixture of Experts**

- **Tree of MoE where each MoE is an expert in a higher-level MoE**
- **Soft decision tree:** Takes a weighted (gating) average of all leaves (experts), as opposed to using a single path and a single leaf
- **Can be trained using EM** (Jordan and Jacobs, 1994)

**Margin**

- Distance from the discriminant to the closest instances on either side
- Distance of \( x \) to the hyperplane is \( \frac{w'x + b}{||w||} \)
- We require \( \frac{r(w'x + w_0)}{||w||} \)
- For a unique \( s_0 \), fix \( p \) to \( ||w|| = 1 \), and to max margin
  \[ \min_{0 \leq p} \frac{1}{2} ||w||^2 \text{ subject to } r(w'x + w_0) > 1, \forall x \]

**Kernel Machines**

- **Discriminant-based:** No need to estimate densities first
- Define the discriminant in terms of support vectors
- The use of kernel functions, application-specific measures of similarity
- No need to represent instances as vectors
- Convex optimization problems with a unique solution

**Optimal Separating Hyperplane**

- \( X = \{x_i, r_i \} \) where \( r_i = +1 \) if \( x_i \in C_+ \)
- find \( w \) and \( w_0 \) such that
  - \( w'x + w_0 > 1 \) for \( r_i = +1 \)
  - \( w'x + w_0 < -1 \) for \( r_i = -1 \)
- which can be rewritten as
  \[ r_i (w'x + w_0) \geq 1 \]

(Cortes and Vapnik, 1995; Vapnik, 1995)

**Classification**

- \[ L(m_s, s, w_v | X) = \sum_i \log \sum_v g' \prod \phi(x'_i | v) \]

- \( \gamma'_i = \exp \left( \frac{y_i - y_{r_i}}{2\sigma^2} \right) \)

- Soft decision tree:

  \[ x \] near the boundary is \( f'_i = \frac{p(x | h_i)}{\sum_j p(x | h_j)} \)

  \[ s_i = \sum_j \frac{f'_i}{g'_i} (y_j - y_{r_i}) (y_j - y_{r_i})' \]

  \[ w_i = \sum_j \frac{f'_i}{g'_i} r_i' \]

**Mixture of Experts**

- In RBF, each local fit is a constant, \( w_o \), second layer weight
- In MoE, each local fit is a linear function of \( x \), a local expert, \( \{w_k \} \)

(Jacobs et al., 1991)
Defining good measures of similarity

Polynomials of degree $\theta$:

- Radial-basis functions:
  \[
  K(x, x') = \exp(-\|x - x'|^2 / \sigma^2)
  \]

- String kernels, graph kernels, image kernels, ...

- Kernel “engineering”
  
  Defining good measures of similarity

Empirical kernel maps: Define a set of templates $m_i$ and score function $s(x, m_i) = \phi(x)^T \phi(m_i)$

\[ K(x, x') = \sum_i s(x, m_i) s(x', m_i) \]

SVM for Regression

- Use a linear model (possibly kernelized)
  \[ f(x) = \langle w, x \rangle + w_0 \]

- Use the $\epsilon$-sensitive error function
  \[ e_t(x, f(x)) = \begin{cases} 0 & |f(x) - y| \leq \epsilon \\ |f(x) - y| - \epsilon & \text{otherwise} \end{cases} \]

\[ \min_{w, b} \frac{1}{2} \langle w, w \rangle + C \sum_i e_t(x_i, y_i) \]

Soft Margin Hyperplane

- Not linearly separable
  \[ r^t (w' x' + w_s) \leq 1 - \xi^t \]

- Soft error
  \[ \sum_i \xi_i^t \]

- New primal is
  \[ \min_{w, b} \frac{1}{2} \langle w, w \rangle + C \sum_i e_t(x_i, y_i) \]

Kernel Trick

- Preprocess input $x$ by basis functions
  \[ z = \phi(x) \]
  \[ g(z) = \langle w, z \rangle \]

- The SVM solution
  \[ w = \sum_i a_i^t r_i^t x_i \]
  \[ g(x) = \sum_i a_i^t \phi(x_i)^T \phi(x) \]

- Adaptive kernel combination
  \[ K_{\text{total}}(x, x') \]

- Localized kernel combination
  \[ K_{\text{localized}}(x, x') \]

Multiclass Kernel Machines

- 1-vs-all
- Pairwise separation
- Error-Correcting Output Codes (section 17.5)
- Single multiclass optimization

\[ \min_{w, b} \frac{1}{2} \langle w, w \rangle + C \sum_i e_t(x_i, y_i) \]

Kernel Machines for Ranking

- We require not only that scores be correct order
  but at least +1 unit margin.

- Linear case:
  \[ \min_{w, b} \frac{1}{2} \langle w, w \rangle + C \sum_i \xi_i^t \]

- Pairwise separation
  \[ w^T x > w^T x' + 1 - \xi_i^t \]

- Error-Correcting Output Codes (section 17.5)
Consider a sphere with center $a$ and radius $R$:

$$\min r^2 + C \sum a'$$
subject to
$$l_k = \sum a' \alpha' \leq 1$$

Case 1: Head-to-Head

$$P(X, Y|Z) = P(X|Y)P(Z|Y)$$

Case 2: Tail-to-Tail

$$P(X, Y, Z) = P(X, Y)P(Z)$$

Case 3: Head-to-Tail

Conditional Independence

- $X$ and $Y$ are independent if $P(X, Y) = P(X)P(Y)$
- $X$ and $Y$ are conditionally independent given $Z$ if $P(X, Y|Z) = P(X|Z)P(Y|Z)$
- Three canonical cases: Head-to-tail, Tail-to-tail, head-to-head

Graphical Models

- Aka Bayesian networks, probabilistic networks
- Nodes are hypotheses (random vars) and the probability corresponds to our belief in the truth of the hypothesis
- Arrows are direct influences between hypotheses
- The structure is represented as a directed acyclic graph (DAG)
- The parameters are the conditional probabilities in the arcs (Pearl, 1988, 2000; Jensen, 1996; Lauritzen, 1996)

Large Margin Nearest Neighbor

- Learns the matrix $M$ of Mahalanobis metric
  $$D(x', x) = (x' - x)M(x' - x)$$
- For three instances $i, j,$ and $l$, where $i$ and $j$ are of the same class and $l$ different, we require
  $$D(x', x) > D(x', x) + 1$$
and if this is not satisfied, we have a slack for the difference and we learn $M$ to minimize the sum of such slacks over all $i,j,l$ triples ($i$ and $l$ being one of $k$ neighbors of $i$, over all $i$)

Learning a Distance Measure

- LMNN algorithm (Weinberger and Saul 2009)
  $$(1 - \mu)\sum P(x') + \sum (1 - \mu)\xi_x$$
subject to
$$D(x', x) > D(x', x) \pm \xi_x$$
- LMCA algorithm (Torresani and Lee 2007) uses a similar approach where $M=L$ and learns $L$

Class Kernel Machines

- Kernel PCA does PCA on the kernel matrix (equal to canonical PCA with a linear kernel)
- Kernel LDA, CCA

Kernel Dimensionality Reduction

- Kernel PCA does PCA on the kernel matrix (equal to canonical PCA with a linear kernel)
- Kernel LDA, CCA

Causal vs Diagnostic Inference

- Causal inference: If the sprinkler is on, what is the probability that the grass is wet?
- Diagnostic inference: If the grass is wet, what is the probability that the sprinkler is on?
Exploiting the Local Structure

In a Markov random field, dependencies are symmetric, for example, pixels in an image.

In an undirected graph, A and B are independent if removing C makes them unconnected.

Potential function \( v_i(X) \) shows how favorable is the particular configuration \( X \) over the clique \( C \).

The joint is defined in terms of the clique potentials

\[
p(X) = \frac{1}{Z} \prod_i v_i(X) 
\]

where normalizer \( Z = \sum v_i(X) \)

Junction Trees

If \( X \) does not separate \( E^+ \) and \( E^- \), we convert it into a junction tree and then apply the polytree algorithm.

Tree of moralized, clique nodes

Influence Diagrams

A path from node A to node B is blocked if

- The directions of edges on the path meet head-to-head (case 1) or tail-to-tail (case 2) and the node is in \( C \), or
- The directions of edges meet head-to-tail (case 3), and neither node nor any of its descendants is in \( C \).

If all paths are blocked, A and \( B \) are d-separated [conditionally independent] given \( C \).

Belief Propagation (Pearl, 1988)

Chain Propagation (Pearl, 1988)

- Chain:

  \[
  P(X) = \frac{1}{Z} \prod_i P(X_i | \text{parents}(X_i)) 
  \]

- Factor Graphs

  \[
  p(x) = \frac{1}{Z} \prod_i v_i(x) 
  \]

  where normalizer \( Z = \sum v_i(x) \)

- Learning a Graphical Model

  Learning the conditional probabilities, either as tables (for discrete case with small number of parents), or as parametric functions.

  Learning the structure of the graph: Doing a state-space search over a score function that uses both goodness of fit to data and some measure of complexity.
Introduction

Modeling dependencies in input; no longer iid

Sequences:
- Temporal: In speech, phenomena in a word (dictionary); words in a sentence (syntax, semantics of the language).
- In handwriting, pen movements
- Spatial: In a DNA sequence; base pairs

Discrete Markov Process

- N states: $S_1, S_2, \ldots, S_N$. State at "time" $t$, $q_t = S_i$.
- First-order Markov

$$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_{j-1}) = P(q_{t+1} = S_j | q_t = S_i)$$

- Transition probabilities

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \quad a_{ij} \geq 0 \text{ and } \sum_{j=1}^{N} a_{ij} = 1$$

- Initial probabilities

$$\pi_i = P(q_1 = S_i) \quad \sum_{i=1}^{N} \pi_i = 1$$

Balls and Urns: Learning

- Given K example sequences of length T

$$\delta_t(i) \equiv \max_{j=1}^{N} \sum_{l=1}^{t} (\delta_{l-1}(j)a_{ij}b(O_l))$$

Three Basic Problems of HMMs

1. Evaluation: Given $A$ and $O$, calculate $P(O | A)$

2. State sequence: Given $A$ and $O$, find $Q^*$ such that

3. Learning: Given $X = \{Q^*_T, \ldots, Q^*_1\}$ are recorded; a probabilistic function of the state

Hidden Markov Models

- States are not observable
- Discrete observations $\{O_1, O_2, \ldots, O_T\}$ are recorded; a probabilistic function of the state

$$P(O_T, \ldots, O_1 | A)$$

- Emission probabilities

$$b_j(m) = P(O = o_m | q_t = S_j)$$

- Example: In each urn, there are balls of different colors, but with different probabilities.

- For each observation sequence, there are multiple state sequences

HMM Unfolded in Time

- Forward variable:

$$\alpha_t(i) = P(O_1, \ldots, O_t, q_t = S_i | A)$$

- Initialization:

$$\alpha_0(i) = \pi_i, \beta_T(i) = 1$$

- Recursion:

$$\alpha_t(i) = \sum_{j=1}^{N} \alpha_{t-1}(j)a_{ji}b(O_t)$$

- Backward variable:

$$\beta_T(i) = 1$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}b(O_t)\beta_{t+1}(j)$$

Finding the State Sequence

- For each time step $t$:

$$\hat{q}_t = \arg \max_i \beta_t(i)$$

Continuous Observations

- Discrete: $P(0_i | q_t = S_j) \prod_{k=1}^{K} b_k(m_i)$, $c_{jk} = 1$ if $O_i = v_k$

- Gaussian mixture (Discretize using k-means):

$$P(0_i | q_t = S_j) \sum_k b_k P(0_i | v_k)$$

Choices: $P(0_i | q_t = S_j) \sim N(\mu_j, \sigma_j^2)$

Use EM to learn parameters, e.g.
HMM with Input

- Input-dependent observations:
  \[ p(q_t | q_{t-1}, x_t) = \mathcal{N}(q_t | \mu, \sigma^2) \]
- Input-dependent transitions (Mello and Jordan, 1996; Bengio and Frasconi, 1996):
  \[ p(q_{t+1} = q_t | x_t) \]
- Time-delay input:
  \[ x_t' = f(q_{t-1}, ..., q_1) \]

HMM as a Graphical Model

Bayesian Approach

1. Prior \( p(\theta) \) allows us to concentrate on region where \( \theta \) is likely to lie, ignoring regions where it’s unlikely
2. Instead of a single estimate with a single \( \theta \), we generate several estimates using several \( \theta \) and average, weighted by how their probabilities Even if prior \( p(\theta) \) is uninformative, (2) still helps. MAP estimator does not make use of (2):
   \[ \hat{\theta}_{MAP} = \arg\max_{\theta} p(X | \theta) \]

Estimating the Parameters of a Distribution: Continuous case

- \( p(x) \sim \mathcal{N}(\mu, \sigma^2) \)
- Gaussian prior for \( \mu, p(\mu) \sim \mathcal{N}(\mu_0, \sigma_0^2) \)
- Posterior is also Gaussian \( p(\mu|x) \sim \mathcal{N}(\mu_0 + \frac{N \mu}{\sigma^2}, \frac{N}{\sigma^2}) \)

Gaussian: Prior on Variance

- Let's define a prior (gamma) on precision \( \lambda = 1/\sigma^2 \)
  \[ p(\lambda) = \text{gamma}(\alpha, \beta) = \frac{1}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta \lambda} \]
  \[ p(\lambda) = \prod_{i=1}^{N} \lambda^{\alpha_i/2} \exp \left[ -\frac{\lambda}{2} \left( \frac{N-1}{\beta} - n \right) \right] \]
  \[ = \mathcal{N}(\lambda | \alpha_0/\beta, \frac{1}{\alpha_0}) \]
  \[ p(X) \propto p(X | \lambda) \propto \text{gamma}(\alpha_0, \beta_0) \]
  \[ \alpha_0 = \alpha_0 + N/2 \]
  \[ \beta_0 = \beta_0 + \frac{n}{2} \]

Joint Prior and Making a Prediction

- \( p(X) = \prod_{i=1}^{N} \mathcal{N}(x_i | \mu, \sigma^2) \)
  \[ = \mathcal{N}(X | \mu, \sigma^2) \]

Conjugate prior: Posterior has the same form as prior
- Uniform prior: Posterior has the same form as likelihood
- Laplace approximation: Use a Gaussian
- Variational approximation: Split the multivariate density into a set of simpler densities using dependencies

Model Selection in HMM

- In classification, for each \( C_i \) estimate \( P(O | \lambda_i) \) by a separate HMM and use Bayes’ rule
  \[ P(\lambda_i | O) \propto P(O | \lambda_i) P(\lambda_i) \]

Rationale

- Parameters \( \theta \) not constant, but random variables with a prior, \( p(\theta) \)
- Bayes’ Rule: \( p(O | X) = \frac{p(O, X | \theta) p(\theta)}{p(X)} \)
Defining a prior is subjective

Marginal likelihood of a model:

Posterior probability of model given data:

Customers arrive and either join one of the existing tables or start a new one, based on the table occupancies:

Nonparametric Bayes

Gaussian Processes

Dirichlet Processes

Nonparametric Gaussian Mixture

Latent Dirichlet Allocation
Nonparametric Bayesian approach for feature extraction

Matrix factorization:

\[ X = ZA \]

\[ A = \begin{cases} 1 \text{ with probability } \mu_j, \\ 0 \text{ with probability } 1 - \mu_j \end{cases} \]

Nonparametric version: Allow \( j \) to increase with more data probabilistically

Indian buffet process: Customer can take one of the existing dishes with prob \( \mu_j \) or add a new dish to the buffet

With reasonable \( L \), find \( W \) such that the Hamming distance btw rows and columns are maximized.

Voting scheme

\[ y = \sum_{j=1}^{K} w_j d_j \]

Subproblems may be more difficult than one-per-\( K \)

Bayesian perspective:

\[ p(x|\theta) = \sum_{\mathcal{M}_j} p(x|\mathcal{M}_j) p(\mathcal{M}_j) \]

If \( d_j \) are iid

\[ \mathbb{E}[d_j] = \frac{1}{L} \left( \mathbb{E}[d_j^2] - \mathbb{E}[d_j] \right) \]

Bias does not change, variance decreases by \( L \)

If dependent, error increase with positive correlation

\[ \mathbb{V}[d_j^2] = \frac{1}{L} \left( \mathbb{V}[d_j^2] + 2 \mathbb{V}[d_j] \right) \]

Error-Correcting Output Codes

<table>
<thead>
<tr>
<th>( K ) classes; ( L ) problems (Dietterich and Bakiri, 1995)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code matrix ( W ) codes classes in terms of learners</td>
</tr>
<tr>
<td>One per class ( L=K )</td>
</tr>
<tr>
<td>( W = \begin{pmatrix} 1 &amp; -1 &amp; -1 &amp; -1 \ -1 &amp; 1 &amp; -1 &amp; -1 \ -1 &amp; -1 &amp; 1 &amp; -1 \ -1 &amp; -1 &amp; -1 &amp; 1 \end{pmatrix} )</td>
</tr>
<tr>
<td>With reasonable ( L ), find ( W ) such that the Hamming distance btw rows and columns are maximized.</td>
</tr>
<tr>
<td>Voting scheme</td>
</tr>
<tr>
<td>( y = \sum_{j=1}^{K} w_j d_j )</td>
</tr>
<tr>
<td>Subproblems may be more difficult than one-per-( K )</td>
</tr>
</tbody>
</table>

Bagging

Use bootstrapping to generate \( L \) training sets and train one base-learner with each (Breiman, 1996)

Use voting (Average or median with regression)

Unstable algorithms profit from bagging

AdaBoost

Generate sequence of base-learners each focusing on previous one's errors

Use voting (Average or median with regression)

Unstable algorithms profit from bagging

Stacking

Voting where weights are input-dependent (gating)

\[ y = \sum_{j=1}^{K} w_j d_j \]

Experts or gating can be nonlinear

Combining another learner (Walperton, 1992)

Fine-Tuning an Ensemble

Given an ensemble of dependent classifiers, do not use it as is, try to get independence

- Subset selection: Forward (growing)/Backward (shrinking) pruning approaches to improve accuracy/diversity/independence
- Train metaclassifiers: From the output of correlated classifiers, extract new combinations that are uncorrelated. Using PCA, we get “eigenlearners.”
- Similar to feature selection vs feature extraction

Cascading

Use \( d_j \) only if preceding ones are not confident

Cascade learners in order of complexity
Combining Multiple Sources/Views

- Early integration: Concat all features and train a single learner
- Late integration: With each feature set, train one learner, then either use a fixed rule or stacking to combine decisions
- Intermediate integration: With each feature set, calculate a kernel, then use a single SVM with multiple kernels
- Combining features vs decisions vs kernels

Introduction

- Game-playing: Sequence of moves to win a game
- Robot in a maze: Sequence of actions to find a goal
- Agent has a state in an environment, takes an action and sometimes receives reward and the state changes
- Credit-assignment
- Learn a policy

Single State: K-armed Bandit

- Among K levers, choose the one that pays best Q(a): value of action a
- Reward is \( r \)
- Set Q(a) = \( r \)
- Choose a* if \( Q(a^*) = \max_a Q(a) \)
- Rewards stochastic (keep an expected reward): \( Q(s,a) = -Q(s,a) + \eta[r(s,a) - Q(a)] \)

Elements of RL (Markov Decision Processes)

- \( s_t \): State of agent at time t
- \( a_t \): Action taken at time t
- In \( s_t \), action \( a_t \) is taken, clock ticks and reward \( r_{t+1} \) is received and state changes to \( s_{t+1} \)
- Next state prob: \( P(s_{t+1} \mid s_t, a_t) \)
- Reward prob: \( p(r_{t+1} \mid s_t, a_t) \)
- Initial state(s), goal state(s)
- Episode (trial) of actions from initial state to goal
- (Sutton and Barto, 1998; Koelbling et al., 1996)

Model-Based Learning

- Environment, \( P(s_{t+1} \mid s_t, a_t) \) and \( p(r_{t+1} \mid s_t, a_t) \) known
- There is no need for exploration
- Can be solved using dynamic programming
- For all states \( s, a \in A \)
- \( Q(s,a) = E_{r} E_{s'} P(s',a,s) V(s') \)
- Optimal policy \( \pi^* = \arg \max_{\pi} E_{r} E_{s'} P(s',a,s) V(s') \)

Value Iteration

- Initialize \( V(s) \) to arbitrary values
- Repeat
  - For all \( s \in S \)
    - For all \( a \in A \)
      - \( Q(s,a) = E_{r} E_{s'} P(s',a,s) V(s') \)
      - Until \( V(s) \) converges

Policy Iteration

- Initial policy \( \pi \) arbitrarily
- repeat
  - Compute the values using \( \pi \) by solving the linear equations
    - \( V(s) = E_{r} E_{s'} P(s',a,s) V(s') \)
  - Improve the policy at each state
    - \( \pi'(s) = \arg \max_{\pi} E_{r} E_{s'} P(s',a,s) V(s') \)
  - Until \( \pi \approx \pi' \)

Exploration Strategies

- E-greedy: With pr \( \epsilon \), choose one action at random uniformly, and choose the best action with pr 1-\( \epsilon \)
- Probabilities: \( P(a) = \frac{\exp(Q(s,a))}{\sum_{b} \exp(Q(s,b))} \)
- Move smoothly from exploration/exploration.
- Decrease \( \epsilon \)
- Annealing: \( P(a) = \frac{\exp(Q(s,a)/\epsilon)}{\sum_{b} \exp(Q(s,b)/\epsilon)} \)

Deterministic Rewards and Actions

- Q(s,a) = \( E_{r} E_{s'} P(s',a,s) \max_{\pi} Q(s',a) \)
- Deterministic single possible reward and next state \( Q(s,a) = r + \gamma \max_{\pi} Q(s',a) \)
- used as an update rule (backup)
- \( Q(s,a) = r + \gamma \max_{\pi} Q(s',a) \)
- Starting at zero, Q values increase, never decrease
Nondeterministic Rewards and Actions

- When next states and rewards are nondeterministic (there is an opponent or randomness in the environment), we keep averages (expected values) instead as assignments.
- Q-learning (Watkins and Dayan, 1992):
  \[ \hat{Q}(s,a) = \hat{Q}(s,a) + \alpha \left( r + \gamma \max' Q(s',a') - \hat{Q}(s,a) \right) \]
- Off-policy vs on-policy (Sarsa)
- Learning V (TD-learning: Sutton, 1988)
  \[ V(s) \leftarrow V(s) + \alpha \left( r + \gamma V(s') - V(s) \right) \]

Q-learning

- Initialize all \( Q(s,a) \) arbitrarily
- For all episodes
- Initialize \( s \)
- Choose \( a \) using policy derived from \( Q \), e.g., \( \epsilon \)-greedy
- Take action \( a \), observe \( r \) and \( s' \)
- Update \( Q(s,a) \):
  \[ Q(s,a) \leftarrow Q(s,a) + \alpha (r + \gamma \max' Q(s',a') - Q(s,a)) \]
- Until \( s \) is terminal state

Sarsa (\( \lambda \))

- Initialize all \( Q(s,a) \) arbitrarily, \( \epsilon(s,a) = 0 \), \( \epsilon(a) \)
- For all episodes
- Initialize \( s \)
- Choose \( a \), using policy derived from \( Q \), e.g., \( \epsilon \)-greedy
- Take action \( a \), observe \( r \) and \( s' \)
- Choose \( \lambda' \) using policy derived from \( Q \), e.g., \( \epsilon \)-greedy
- Update \( Q(s,a) \):
  \[ Q(s,a) \leftarrow Q(s,a) + \alpha [r + \gamma \max' Q(s',a') - Q(s,a)] \]
- Until \( s \) is terminal state

Sarsa

- Initialize all \( Q(s,a) \) arbitrarily
- For all episodes
- Initialize \( s \)
- Choose \( a \) using policy derived from \( Q \), e.g., \( \epsilon \)-greedy
- Take action \( a \), observe \( r \) and \( s' \)
- Choose \( \lambda' \) using policy derived from \( Q \), e.g., \( \epsilon \)-greedy
- Update \( Q(s,a) \):
  \[ Q(s,a) \leftarrow Q(s,a) + \alpha (r + \gamma \max' Q(s',a') - Q(s,a)) \]
- Until \( s \) is terminal state

Eligibility Traces

- Keep a record of previously visited states (actions)
  \[ e_{t}(s,a) = \begin{cases} 1 & \text{if } s = s_{t} \text{ and } a = a_{t} \\ 0 & \text{otherwise} \end{cases} \]
- \( \delta_{t} = r_{t} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_{t},a_{t}) \)
- \( Q(s_{t},a_{t}) \leftarrow Q(s_{t},a_{t}) + \beta e_{t}(s,a) \delta_{t} \)

Generalization

- Tabular: \( Q(s,a) \) or \( V(s) \) stored in a table
- Regressor: Use a learner to estimate \( Q(s,a) \) or \( V(s) \)
  \[ \Delta = \beta \sum_{i=1}^{n} e_{i} (Q_{i}(s,a) - Q(s,a)) \]

Partially Observable States

- The agent does not know its state but receives an observation \( p(a_{t+1}|s_{t+1}) \) which can be used to infer a belief about states
- Partially observable MDP

The Tiger Problem

- Two doors, behind one of which there is a tiger
- \( p \): prob that tiger is behind the left door

The MIT Press, 2014
alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml3e
Questions:

- Assessment of the expected error of a learning algorithm is the error rate of 1-NN less than 2%?
- Comparing the expected errors of two algorithms is k-NN more accurate than MLP?
- Training/validation/test sets
- Resampling methods: K-fold cross-validation

Criteria (Application-dependent):
- Misclassification error, or risk (loss functions)
- Training time/space complexity
- Testing time/space complexity
- Interpretablity
- Easy programmability
- Cost-sensitive learning

Response function based on output to be maximized:
- Depends on controllable factors
- Uncontrollable factors introduce randomness
- Find the configuration of controllable factors that maximizes response and minimally affected by uncontrollable factors

1. Aim of the study
2. Selection of the response variable
3. Choice of factors and levels
4. Choice of experimental design
5. Performing the experiment
6. Statistical Analysis of the Data
7. Conclusions and Recommendations

Performance Measures

- Error rate = # of errors / # of instances = (FN+FP) / N
- Recall = # of ... / # of found = TP / (TP+FN) = sensitivity = hit rate
- Specificity = TN / (TN+FP)
- False alarm rate = FP / (FP+TN) = 1 - Specificity

Interval Estimation

- X = {x}| where x~N (μ, σ²)
- m = N (μ, σ²/N)

Hypothesis Testing

- Reject a null hypothesis if not supported by the sample with enough confidence

\[
H_0: \mu = \mu_0 \quad \text{vs.} \quad H_1: \mu > \mu_0
\]

Accept H_0 with level of significance α if \(\mu_0\) is in the 100(1-α) confidence interval

\[
\frac{\sqrt{(m-\mu)} - 1.64}{\sigma} < 1-\alpha
\]

100(1-α) percent one-sided confidence interval

\[1-\alpha = \frac{\sqrt{(m-\mu)} - 1.64}{\sigma}
\]

Two-sided test

\[
\frac{\sqrt{(m-\mu)} - 1.64}{\sigma} < 1-\alpha
\]

100(1-α) percent two-sided confidence interval
Assessing Error: $H_0: p \leq p_0$ vs. $H_1: p > p_0$

- Single training/validation set: Binomial Test
  - If error prob is $p_0$, prob that there are $e$ errors less in $N$ validation trials is
  $$P(X \leq e) = \sum_{j}^N \binom{N}{j} p_0^j (1-p_0)^{N-j}$$
  - Accept if this prob is less than $1 - \alpha$

K-Fold CV Paired $t$ Test

- Use K-fold cv to get K training/validation folds
- $p_1, \ldots, p_K$: Errors of classifiers 1 and 2 on fold $i$
- $p_i = p_i^1 - p_i^2$: Paired difference on fold $i$
- The null hypothesis is whether $p_i$ has mean 0
  $$H_0: \mu_x = 0 \text{ vs. } H_1: \mu_x \neq 0$$
- $m = K$:
  $$s^2_i = \frac{\sum (p_i - \mu_i)}{K-1}$$
  - Thus an estimator of $\sigma^2$ is $s^2_i$'s average:
  $$\bar{s}^2 = \frac{\sum s^2_i}{K}$$
  - So when $H_i$ is true, we have
  $$\frac{s^2}{\bar{s}^2} \sim \chi^2_{K-1}$$

Regardless of $H_i$, our secondestimator to $\sigma^2$ is the average of group variances $s^2_j$:

$$s^2_j = \frac{\sum (x_j - \mu_j)^2}{K-1}$$

- $\bar{s}^2 = \frac{\sum s^2_j}{K}$
- $SSw = \sum (x_j - \mu_j)^2$
- $K\bar{s}^2_{x} = \sum s^2_j$
- $K\bar{s}^2_{x} / \bar{s}^2_{x}$
- $SSw / \bar{s}^2_{x}$
- $SSw / (K-1)$
- $SSw / (K-1)$
- $F_{a,b}$

### Multivariate Tests

- Instead of testing using a single performance measure, e.g., error, use multiple measures for better discrimination, e.g., [fp-rate, fn-rate]
- Compare p-dimensional distributions
- Parametric case: Assume p-variate Gaussians
- $H_i: \mu_1 = \mu_2 \text{ vs. } H_i: \mu_1 \neq \mu_2$

### Multivariate Pairwise Comparison

- Paired differences: $d_i = X_i - X_{2i}$
- $H_0: \mu_1 = 0 \text{ vs. } H_1: \mu_1 \neq 0$
- Hotelling's multivariate $T^2$ test
- $T^2 = \frac{Km}{S^2} \sum d_i^2$
- For $p=1$, reduces to paired $t$ test

### Multivariate ANOVA

- Comparison of L>2 algorithms
  - Sign test: Count how many times A beats B over N datasets, and check if this could have been by chance if A and B did have the same error rate
  - Comparing multiple algorithms
    - Kruskal-Wallis test: Calculate the average rank of all algorithms on N datasets, and check if this could have been by chance if all had equal error rate
    - If KW rejects, we do pairwise posthoc tests to find which ones have significant rank difference

### Comparing Classifiers: $H_0: \mu_0 = \mu_1$ vs. $H_1: \mu_0 \neq \mu_1$

- Single training/validation set: McNemar's Test
- $e_{0j}$: Number of examples misclassified by both
- $e_{ij}$: Number of examples misclassified by 2 but not 1
- Under $H_0$, we expect $e_{0j} = e_{ij} = \frac{k^2}{2}$

Under $H_0$, we expect $e_{0j} = e_{ij} = \frac{k^2}{2}$

Accept if < $\chi^2_{0.05}$