CSC520 - Artificial Intelligence

Lecture 21

Dr. Scott N. Gerard

North Carolina State University

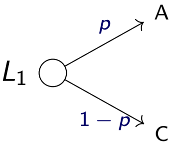
Apr 3, 2025

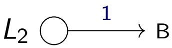
Decision Theory

- *Decision theory* $=$ *Probability theory* $+$ *Utility theory*

Axioms of Utility Theory

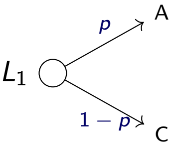
- Lottery L is possible outcomes $S_1, \ldots, S_n$ with probabilities $p_1, \ldots, p_n$ of an action
  - $L = [p_1 : S_1; p_2 : S_2; \ldots p_n : S_n]$
  - S_i can be a state or another lottery
- Orderability: $A \succ B$, $B \succ A$, or $A \sim B$
- Transitivity: $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$
- Continuity: $A \succ B \succ C$, then $\exists p$ s.t. $L_1 \sim L_2$

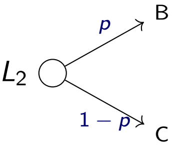




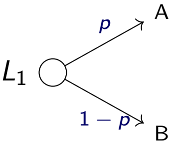
Axioms of Utility Theory

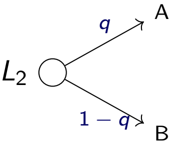
- Substitutability: $(A \sim B)$ then $(L_1 \sim L_2)$





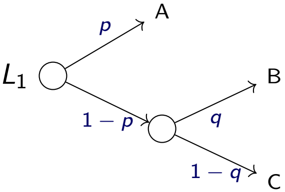
- Monotonicity: $A \succ B$, then $(p > q) \Leftrightarrow (L_1 \succ L_2)$

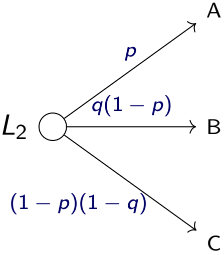




Axioms of Utility Theory

- Decomposability: $L_1 \sim L_2$
  $[p : A; 1-p : [q : B; 1-q : C]] \sim [p : A; (1-p)q : B; (1-p)(1-q) : C]$





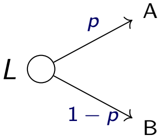
Existence of Utility Function

- Theorem: If an agent's preferences satisfy the axioms, then a utility function exists such that:

$$U(A) > U(B) \Leftrightarrow (A \succ B)$$
$$U(A) = U(B) \Leftrightarrow (A \sim B)$$

- Utility of a lottery is the expected utility of its outcomes

$$U(L) = pU(A) + (1-p)U(B)$$



Agenda

- Machine Learning
- Types of ML algorithms
- Linear regression and gradient descent
- Logistic regression
- Regularization
- Hypothesis evaluation
- K-Means clustering
- Naive Bayes model

Machine Learning

- Field of study that gives computers the ability to learn without being explicitly programmed - Arthur Samuel (1959)
- Drawing conclusions from premises using logical reasoning is *Deduction*
  - Deductive conclusions are always correct
- Primary task of Machine Learning is *Induction*
  - Induction is the process of drawing conclusions from observations (data)
  - Inductive conclusions may be incorrect

Inferencing

	Deduction	**Induction**	**Abduction**
Premise	P	P	???
Rule	$P \Rightarrow Q$	???	$P \Rightarrow Q$
Conclusion	???	Q	Q
	Infer conclusion	Infer rule	Infer premise
	Sound	Approximate	Approximate
	Not ML	**ML**	Likely explanation

Types of Learning Algorithms

- Supervised learning
  - Given data as $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
    learn a hypothesis h such that $\hat{y} = h(x)$
  - Discrete output $\rightarrow$ Classification
    - E.g. learn function to classify email as spam or ham based on email text
  - Continuous output $\rightarrow$ Regression
    - E.g. learn function that predicts house price based on size, location, etc.

Types of Learning Algorithms

- Unsupervised learning
  - Learn patterns in the unlabeled data
  - Typical tasks are: clustering and anomaly detection
    - E.g. group computers into servers, end-user devices, etc. from a network traffic dataset
    - E.g. detect anomalous users from a user activities dataset
- Reinforcement learning
  - Learns optimal actions from the rewards or punishments
    - E.g. robot learns to perform a task in the real world

Linear Regression

- Given data: $(x_1, y_1), (x_2, y_2), \ldots$ where y_i is continuous
- Hypothesis is a linear function: $\hat{y} = h(x) = \theta_0 + \theta_1 x$
- Learn parameters θ_0 and θ_1 such that $\hat{y}$ is close to y
- Minimize the squared difference between $\hat{y}$ and y (loss function)

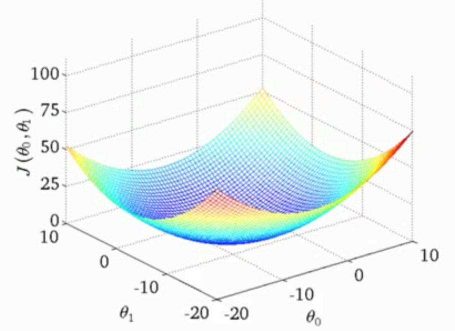
$$
\begin{aligned}
J(\theta_0, \theta_1) &= \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \\
&= \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)^2
\end{aligned}
$$

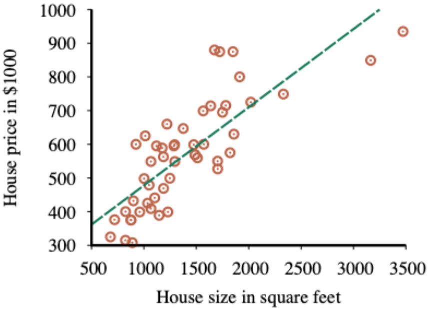
- Goal of the learning is to: $\min\limits_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Gradient Descent

Start with initial values for θ_0,θ_1
while not converged **do**
$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

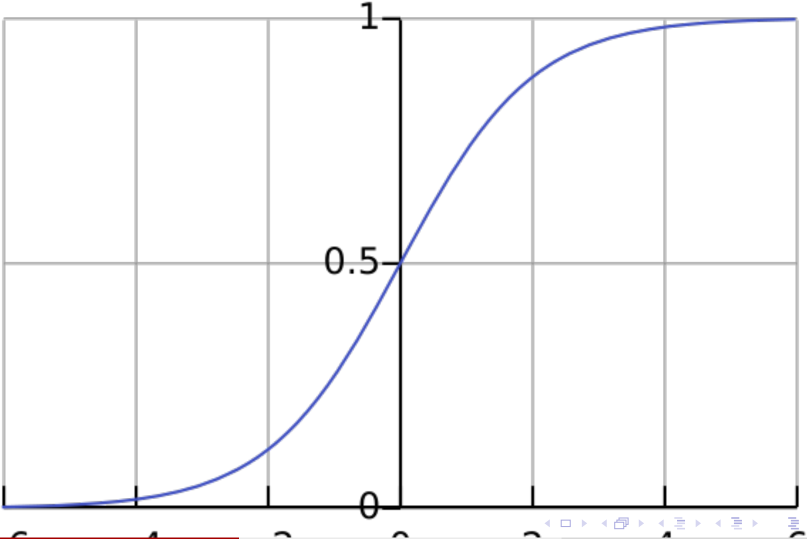
$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_1^m (\hat{y}_i - y_i)$$
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_1^m (\hat{y}_i - y_i) x_i$$





Logistic Function

$$g(z) = \frac{1}{1 + e^{-z}}$$



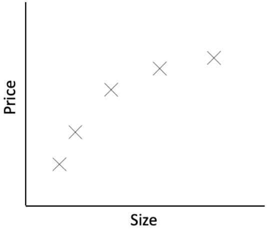
Logistic Regression

- Given data: $(x_1, y_1), (x_2, y_2), \ldots$ where y_i is either 0 or 1
- Hypothesis is a logistic function: $h(x) = \hat{y} = g(\theta_0 + \theta_1 x)$, where $g(z) = \frac{1}{1 + e^{-z}}$, the logistic function
- Loss function

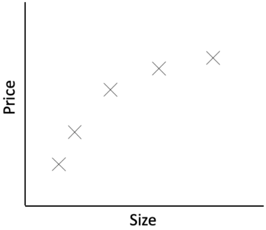
$$J(\theta_0, \theta_1) = -\frac{1}{m} \sum_{i=1}^{m} (y_i log \hat{y}_i + (1 - y_i) log(1 - \hat{y}_i))$$

- Goal of the learning is to: $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$
- Use gradient descent like in linear regression

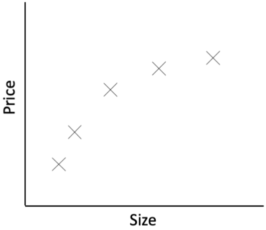
Regularization



$\theta_0 + \theta_1 x$



$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

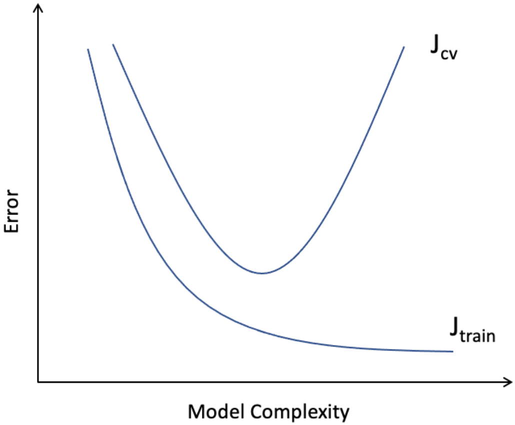
- Manually select which features to keep
- Regularization: Keep all features but reduce values of parameters θ_j

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

Hypothesis Training and Evaluation

- Partition available data into:
  - Train dataset ($\approx 60\%$)
  - (Cross-)Validation (dev) dataset ($\approx 20\%$)
  - Test dataset ($\approx 20\%$)
- Use training dataset to learn the hypothesis parameters (θ_j)
- Use cross-validation dataset to tune hyperparameters such as learning rate, regularization parameter, etc.
- Use test dataset "once" to estimate generalization error

Bias vs Variance



J_{cv}: Cross-validation error
J_{train}: Training error

Confusion Matrix for Classification

	Predicted Positive	Predicted Negative
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

Confusion Matrix Metrics

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

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

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

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

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

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

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

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

Confusion Matrix Metrics

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

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

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

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

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

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

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

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

- red square(s) are in both numerator and denominator. blue is only in denominator. Gray squares are for value used anywhere in calculation.
- Accuracy is useful when test cases are well-balanced between positives and negatives, but not useful when TNs are MUCH larger than all other values.
- When looking for suitable answer out of many possibilities, precision is more important. You don't want to sift through lots of FPs.
- If you're predicting cancer, recall is more important. Additional doctor's appointments will weed out FNs. But the FPs are never investigated.
- F_1 is the harmonic mean: $\frac{1}{F_1} = \frac{\frac{1}{precision} + \frac{1}{recall}}{2}$
- F_1 equally blends precision and recall. The smaller values pulls F_1 below the arithemetic mean.
- F_β adjusts recall and precision weights. Common variants are F_2 (recall weighted twice precision) and $F_{0.5}$ (precision weighted twice recall).

Confusion Matrix Metrics

Multiple ways to calculate F_1

$$\frac{1}{F_1} = \frac{\frac{1}{precision} + \frac{1}{recall}}{2}$$

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

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Confusion Matrix Metrics

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

$$Sensitivity = \frac{TP}{TP+FN} = \frac{TP}{P}$$

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

$$FNR = \frac{FN}{TP+FN} = \frac{FN}{P}$$

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

$$FPR = \frac{FP}{FP+TN} = \frac{FP}{N}$$

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

$$Specificity = \frac{TN}{FP+TN} = \frac{TN}{N}$$

Confusion Matrix Metrics

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

Positive Predictive Value
$$PPV = \frac{TP}{TP+FP}$$

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

False Omission Rate
$$FOR = \frac{FN}{FN+TN}$$

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

False Discovery Rate
$$FDR = \frac{FP}{TP+FP}$$

	Pred P	Pred N
Act P	TP	FN
Act N	FP	TN

Negative Predictive Value
$$NPV = \frac{TN}{FN+TN}$$

Spam Email Classifier Hypothesis Evaluation

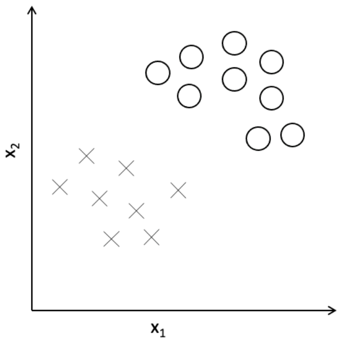
- Confusion matrix

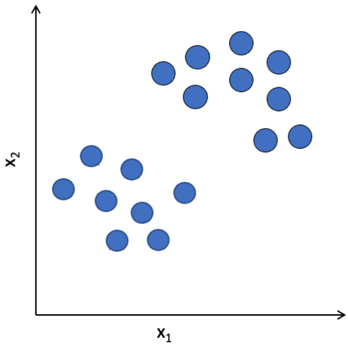
	Predicted Spam	Predicted Ham
Actual Spam	100 (TP)	10 (FN)
Actual Ham	20 (FP)	200 (TN)

- Accuracy $= \dfrac{\text{Correct Predictions}}{\text{All Predictions}} = (100 + 200)/330 \approx 0.90$
- Precision $= \dfrac{\text{Correctly Predicted Spam}}{\text{Total Predicted Spam}} = 100/(100 + 20) \approx 0.83$
- Recall $= \dfrac{\text{Correctly Predicted Spam}}{\text{Total Actual Spam}} = 100/(100 + 10) \approx 0.90$
- F1-score $= \dfrac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = 2 * 0.83 * 0.9/(.83 + .9) \approx 0.86$

K-Means Clustering

- Data has clusters but it is not labeled





- Objective here is to group the data into clusters

K-Means Clustering

- K-Means is a simple clustering algorithm
- Algorithm: Randomly initialize k cluster centers
- Then repeat these two steps until convergence
  - Cluster assignment: Assign a data point to a cluster whose center is closest to it
  - Update cluster centers: Update the cluster centers to the average of the data points assigned to the cluster

K-Means Clustering

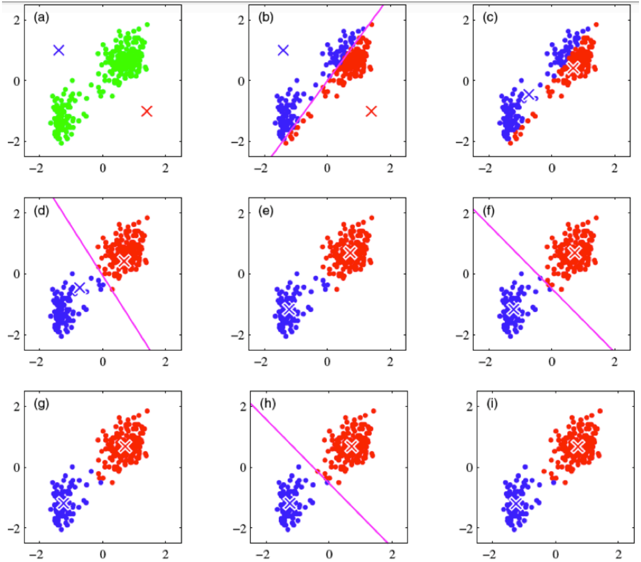


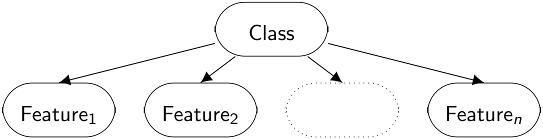
Figure from Roger Grosse, et.al., Univ. of Toronto

K-Means Clustering Optimization Function

- c_i = index of cluster to which example x_i is assigned
- μ_k = cluster center of k
- μ_{c_i} = cluster center of the cluster to which x_i is assigned
- $J(c_1, \ldots, c_m, \mu_1, \ldots, \mu_k) = \frac{1}{m} \sum_{i=1}^{m} \|x_i - \mu_{c_i}\|^2$
- K-Means minimizes the above function over $c_1, \ldots, c_m, \mu_1, \ldots, \mu_k$
- K-Means can get stuck at a local minima
  - Run K-Means multiple times with different initial cluster centers
  - For each run, compute J and use the results of the run with lowest J

Naive Bayes Model

- Most commonly used Bayesian model in machine learning
- Typically used as a baseline for comparison



- Assumes that features are conditionally independent of each other given the class

$$\begin{aligned} P(C|f_1,\ldots,f_n) &= \alpha P(f_1,\ldots,f_n|C)P(C) \\ &= \alpha P(C)\prod_i P(f_i|C) \end{aligned}$$

Naive Bayes Model Example

Age	Prescription	Astigmatism	TearRate	Lenses
Young	Myope	No	Reduced	Noncontact
Young	Myope	No	Normal	Softcontact
Young	Myope	Yes	Reduced	Noncontact
Young	Myope	Yes	Normal	Hardcontact
Young	Hypermetrope	No	Reduced	Noncontact
Young	Hypermetrope	No	Normal	Softcontact
Young	Hypermetrope	Yes	Reduced	Noncontact
Young	Hypermetrope	Yes	Normal	Hardcontact
Prepresbyopic	Myope	No	Reduced	Noncontact
Prepresbyopic	Myope	No	Normal	Softcontact

$$P(Noncontact|Prespresbyopic, Hypermetrope, No, Reduced) =$$
$$\alpha P(Prepresbyopic|Noncontact)P(Hypermetrope|Noncontact)$$
$$P(No|Noncontact)P(Reduced|Noncontact)P(Noncontact)$$
$$= \alpha * 1/5 * 2/5 * 3/5 * 5/5 * 5/10$$
$$= \alpha * .024$$

Class Exercise

- Below is the confusion matrix for a model that classifies user activities as fraudulent or benign.

	Predicted Fraudulent	Predicted Benign
Actual Fraudulent	20	50
Actual Benign	10	5000

- Compute the accuracy, precision, recall and F1-score.
- In this case, is accuracy is a good measure of performance?