Machine Learning Exam Cheat Sheet

Key Concepts

Biases in Machine Learning

Inductive Bias

Inductive bias refers to the assumptions a learning algorithm makes to generalize beyond the training data. It enables the algorithm to choose among multiple hypotheses consistent with the data, making learning possible.

Different learning algorithms can be differentiated/compared based on this bias (p.64):

- **Search/Preference Bias**: This follows from the search strategy. There is a preference for certain hypotheses over others (e.g. shorter hypotheses), with no hard restriction on the hypohteses that can eventually be enumarated. ID3 searches a **complete hypothesis space**, but **searches it incompletely**. Typically, this bias is more desirable (p.64 for justification).
- **Restriction/Language bias**: This follows from the defenition of the search space. There is a categorical restriction on the set of hypotheses considered. Candidate Elimination searches an **incomplete search space**, but **searches it completely**
- Ocam's razor argument for the preference of shorte hypotheses (p.65)

Bias-Variance Trade-off

- **Bias**: Error from overly simplistic assumptions.
- **Variance**: Error from sensitivity to fluctuations in the training set.
- **Trade-off**: Aim for low total error:

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}.$$

Evaluation Metrics

$$\begin{aligned}
\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{Total Instances}}, \\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\
\text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\
\text{F1-Score} &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.
\end{aligned}$$

Overfitting vs. Underfitting

- **Overfitting**: High training accuracy but poor generalization.
- **Underfitting**: Poor performance on both training and test data.
- **Solutions**: Use regularization, simplify the model, or increase data.

Chapter 1: Introduction

Types of Learning

- **Supervised Learning**: Labels provided (e.g., regression, classification).
- **Unsupervised Learning**: No labels (e.g., clustering, dimensionality reduction).
- **Reinforcement Learning**: Learning via rewards and penalties.

Linear Regression

- Hypothesis: $h(x) = \theta_0 + \theta_1 x$.
- Cost Function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$.
- Gradient Descent Update: $\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$.

Logistic Regression

- Hypothesis: $h(x) = \frac{1}{1+e^{-\theta^T x}}$.
- Cost Function:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))].$$

Chatper 2: Concept Learning

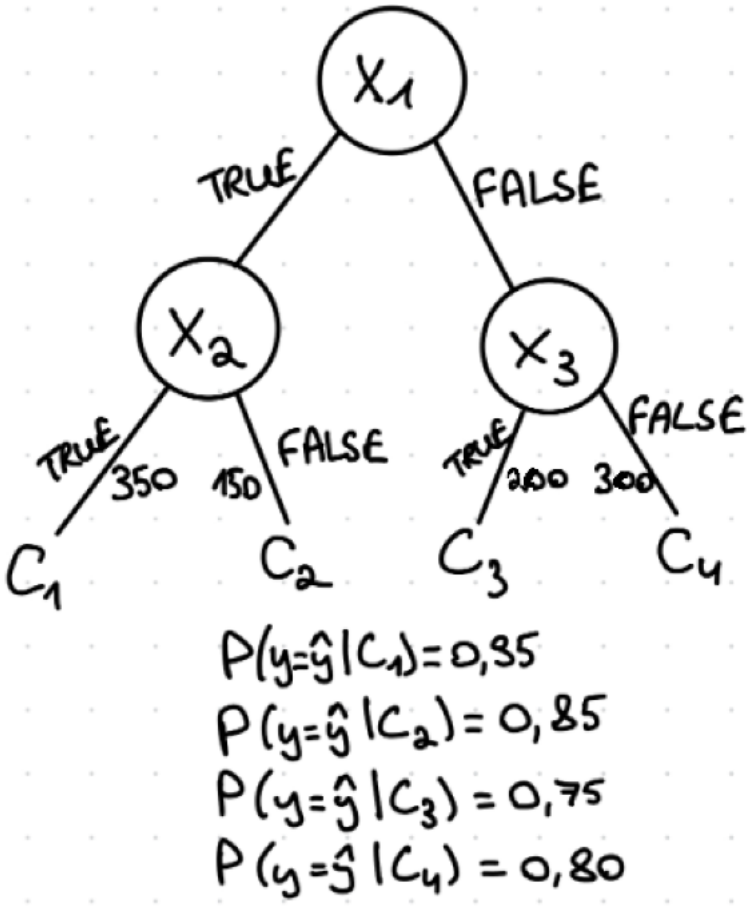
- Defenition of concept learning (p. 21)
- Calculating the instance space, syntactically and semantically distinct hypotheses (p. 23-24)
- **General-to-Specific order** and the importance of **partial order** (p.24-25)
- **Find-S** pseudocode (p.26)
- **negative exaples in Find-S:** Why negative examples are not revised in given certain assumptions (p.27)
- Issues with Find-S: impact of ignoring negative cases on sensitivity to noise (p.28)
- Difference between **consistency** and **satisfiability** of hypotheses (p.29)
- **Candidate-Elimination Algorithm** pseudocode (p.33)
- Useful consequence of general-to-specific ordering:
  - If **positive** instance for h in **S** then surely positive for every h that is more general
  - if **negative** instance for h in **G** then surely negative for every h that is more specific (p.35) Linked with why we first remove inconsistensies in Candidate-Elimination
- Consequences of **violating assumptions** of **Candidate-Elimination** (p.37)
- Best next query strategy (p.38)
- Relation between "best next query strategy" and ambiguity in not fully learned concepts: *"those instances whose classification is most ambiguous are precisely the instances whose true classification would provide the most new information for refingn the vversions space".* (p.39)

Chapter 3: Decision Trees

- **Entropy**: $H(D) = -\sum_i p_i \log_2 p_i$.
- **Information Gain**:
$$IG(D, A) = H(D) - \sum_{v \in \text{Values}(A)} \frac{|D_v|}{|D|} H(D_v).$$
- Prone to overfitting; use pruning techniques.

EXAMPLE Decision Tree and Information Gain Calculation

Tree Representation



Below is the representation of a decision tree:

Explanation of Information Gain Calculation

1. Parent Node Entropy $H(D)$: The dataset contains 1000 examples in total:

- Correctly classified: 740,
- Incorrectly classified: 260.

The probabilities are:

$$P(\text{Correct}) = \frac{740}{1000} = 0.74, \quad P(\text{Incorrect}) = \frac{260}{1000} = 0.26.$$

The entropy of the parent node is:

$$H(D) = -\left(0.74 \log_2(0.74) + 0.26 \log_2(0.26)\right)$$

$$H(D) \approx -\left(0.74 \cdot (-0.432) + 0.26 \cdot (-1.943)\right) \approx 1.247.$$

2. Weighted Child Node Entropies: After splitting on X_1, we calculate the weighted entropy of the child nodes.

a. For $C_1 + C_2$ (TRUE branch):

- Total examples: 500,
- Correct: 350,
- Incorrect: 150.

The probabilities are:

$$P(\text{Correct}) = \frac{350}{500} = 0.7, \quad P(\text{Incorrect}) = \frac{150}{500} = 0.3.$$

The entropy is:

$$H(C_1, C_2) = -\left(0.7 \log_2(0.7) + 0.3 \log_2(0.3)\right)$$
$$H(C_1, C_2) \approx 0.881.$$

b. For $C_3 + C_4$ (FALSE branch):

- Total examples: 500,
- Correct: 390,
- Incorrect: 110.

The probabilities are:

$$P(\text{Correct}) = \frac{390}{500} = 0.78, \quad P(\text{Incorrect}) = \frac{110}{500} = 0.22.$$

The entropy is:

$$H(C_3, C_4) = -\left(0.78 \log_2(0.78) + 0.22 \log_2(0.22)\right)$$
$$H(C_3, C_4) \approx 0.811.$$

c. Weighted Entropy After Split:

$$H_{\text{split}} = \frac{500}{1000} H(C_1, C_2) + \frac{500}{1000} H(C_3, C_4)$$
$$H_{\text{split}} = 0.5 \cdot 0.881 + 0.5 \cdot 0.811$$
$$H_{\text{split}} \approx 0.846.$$

3. Information Gain: The information gain of the top-level node is:

$$IG(D, X_1) = H(D) - H_{\text{split}}$$
$$IG(D, X_1) = 1.247 - 0.846 \approx 0.401.$$

Conclusion

The information gain for splitting on the top-level node X_1 is approximately:

$$IG(D, X_1) \approx 0.401.$$

Chapter 4: Neural Networks

- **Forward Propagation**: Compute activations layer by layer.
- **Backpropagation**: Update weights using gradients from the chain rule.
- **Activation Functions**:
  - Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$.
  - ReLU: $\text{ReLU}(x) = \max(0, x)$.
  - Tanh: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

Chapter 5: Evaluating Hypotheses

Chapter 6: Bayesian Learning

Bayes' Theorem

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

where:

$$P(E) = \sum_i P(E|H_i)P(H_i)$$

MAP (Maximum A Posteriori)

$$H_{\text{MAP}} = \arg\max_H P(H|E) = \arg\max_H P(E|H)P(H)$$

Naive Bayes Classifier

$$P(C|X) = \frac{P(C)\prod_{i=1}^{n} P(x_i|C)}{P(X)}$$

$$P(X|C) = \prod_{i=1}^{n} P(x_i|C)$$

Bayesian Network Example (from WPO8)

$$P(a|c) = \frac{P(c|a)P(a)}{P(c)}$$

where:

$$P(c) = P(c|a)P(a) + P(c|\neg a)P(\neg a)$$

Sum Rule

$$P(A) = P(A \cap B) + P(A \cap \neg B)$$
$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Product Rule

$$P(A \cap B) = P(A|B)P(B)$$

Theorem of Total Probability

$$P(E) = \sum_i P(E|H_i)P(H_i)$$

Joint and Marginal Probabilities (from WPO8)

$$P(a, d|\neg b) = P(d|a, \neg b)P(a|\neg b)$$

$$P(d) = P(d|a, b)P(a, b) + P(d|a, \neg b)P(a, \neg b) + P(d|\neg a, b)P(\neg a, b) + P(d|\neg a, \neg b)P(\neg a, \neg b)$$

Chapter 7: Computational Learning Theory

Version Space and Candidate Elimination

- General-to-Specific Ordering: Hypotheses are refined iteratively using training examples.
- Version Space:
$$VS_{H,D} = \{h \in H : h \text{ consistent with } D\}.$$
- Find-S Algorithm:
  1. Initialize S to the most specific hypothesis.
  2. Generalize S only as needed to be consistent with positive examples.

Input data
Nearest Neighbors
Linear SVM
RBF SVM
Gaussian Process
Decision Tree
Random Forest
Neural Net
AdaBoost
Naive Bayes
QDA
.97
.88
.97
.97
.95
.95
.90
.93
.88
.85
.93
.40
.88
.90
.78
.75
.88
.85
.70
.72
.95
.93
.95
.93
.95
.95
.95
.95
.95
.93

Chapter 8: Instance-Based Learning

k-Nearest Neighbors (k-NN)

- Lazy learner: Stores training data and computes distance during prediction.
- Distance Metric: Euclidean $\sqrt{\sum(x_i - y_i)^2}$.

Chapter 10: Learning Sets of Rules

Chapter 13: Reinforcement Learning

Bellman Equation

$$V(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s') \right].$$

Q-Learning Update Rule

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right].$$

Policy Iteration

1. Policy evaluation.
2. Policy improvement.

Machine Learning Questions and Answers

- **Increasing the depth of a decision tree cannot increase the training error: True.** Increasing the depth allows the tree to fit the training data perfectly, reducing the training error to zero.
- **It is generally a good idea to initiate the weight at 0 when doing gradient descent on the weights of a Neural Network: False.** Initializing weights to zero leads to symmetry breaking issues, making neurons in the same layer identical.
- **When two variables are independent, there is no correlation between them: True.** Independence implies no statistical relationship, leading to zero correlation.
- **A KNN model with N data samples and K dimensionality with $K >> N$ (using Euclidean distance) will perform well in general: False.** The curse of dimensionality makes distances less meaningful, degrading performance.
- **No algorithm exists for a linear regressor that can reach a local optimum by minimizing the mean squared error using gradient descent: False.** Linear regression with mean squared error has a convex loss function, ensuring convergence to the global optimum.
- **In Expectation Maximization, the choice of the initial step (E or M) does not matter for convergence: True.** The algorithm converges regardless of whether it starts with the E-step or the M-step.
- **It is always a bad idea to add noise to your training data: False.** Adding noise can regularize models and prevent overfitting, improving generalization.
- **In reinforcement learning $R(s,a,s') \leq Q^*(s,a)\ \forall s,a,s'$: True.** $Q^*(s,a)$ accounts for immediate and future rewards, always being greater than or equal to the immediate reward.
- **A model that only uses a linear activation function always results in a linear regression model: True.** The composition of linear transformations results in a linear model.

- **If we have a dataset with N elements with K attributes and $K >> N$, then K-nearest neighbor is a good algorithm to classify this: False.** High dimensionality causes equidistance between points, reducing KNN's effectiveness.
- **Adding training examples in a dataset increases the variance: False.** More data reduces the model's dependence on specific instances, decreasing variance.
- **The quality of k-clustering is independent of the choices of the centroids: False.** Poor initial centroids can result in suboptimal clustering outcomes.
- **PAC bounds are guaranteed to be efficient if the data is chosen adversarially: False.** Adversarial data violates the assumptions of PAC learning, making the bounds inefficient.
- **A smaller hypothesis space means a higher chance of overfitting: False.** Overfitting typically occurs when the hypothesis space is too large and flexible.
- **The number of parameters in a Bayesian Network is exponential in the total number of arcs in the graph: False.** The number of parameters is exponential in the number of parents per node, not the total arcs.
- **Using an ensemble of models always leads to better performance compared to using a single model: False.** If individual models are weak or highly correlated, the ensemble may perform worse.
- **Q-learning always finds the optimal policy, if convergence is ensured: True.** Proper exploration and learning rate schedules guarantee convergence to the optimal policy.
- **In an MLP, if all neurons only use a linear activation function, the network can still model non-linear relationships in the data as long as it has enough layers: False.** A network with only linear activations is equivalent to a single linear model, regardless of depth.
- **In the k-means algorithm, the number of clusters (k) must be determined after the algorithm runs and the data points are assigned to clusters: False.** The algorithm requires k as an input to initialize centroids and assign data points.