CS2109S Midtrm

Wxy2003-xy

Search	Time	Space	Complete?	Optimal?
BFS	Exp	Exp	Yes	Yes
UCS	Exp	Exp	Yes	Yes
DFS	Exp	Poly	No	No
DLS(D)	Exp	Poly	No	No
DLS(B)	Exp	Exp	No	Yes
IDS(D)	Exp	Exp	Yes	Yes
IDS(B)	Exp	Exp	Yes	Yes

Informed Search: Uses heuristics.

- **A***: $f(n) = g(n) + h(n)$.
- **Hill Climbing**: Greedy local search.

Heuristics

Admissibility: $h(n) \leq h^*(n)$. Never overestimates cost.
Consistency: $h(n) \leq h(n') + c(n, n')$. Guarantees optimality.
Dominance: $\forall n, h_1(n) \geq h_2(n) \implies h_1$ dominates h_2

Adversarial Search

Minimax Algorithm:

$$\text{max-value}(s) = \max \text{min-value}(s')$$
$$\text{min-value}(s) = \min \text{max-value}(s')$$

Alpha-beta pruning: Prunes unnecessary branches to optimize Minimax.

```
max_value(state, $\alpha$, $\beta$):
    if is_terminal(state): return utility(state)
    v = -$\infty$
    for next_state in expand(state):
        v = max(v, min(v, $\alpha$, $\beta$))
    return v

min_value(state, $\alpha$, $\beta$):
    if is_terminal(state): return utility(state)
    v = $\infty$
    for next_state in expand(state):
        v = min(v, max(v, $\alpha$, $\beta$))
    return v

alpha-beta search(state):
    v = max_value(state, -$\infty$, $\infty$)
    // initialized $\alpha$ to be -$\infty$, $\beta$ to $\infty$
    return action in expand(state) with value v
```

Supervised Learning: Learns from labeled data.
Unsupervised Learning: Finds patterns in unlabeled data.
Reinforcement Learning: Learns via rewards.

Regression

Linear Model: $h_w(x) = w^T x$. **Loss Function (MSE):**

$$J(w) = \frac{1}{2N} \sum_{i=1}^{N} (h_w(x^{(i)}) - y^{(i)})^2 \tag{1}$$

Gradient Descent:

$$w_j \leftarrow w_j - \gamma \frac{\partial J}{\partial w_j} \tag{2}$$

$$\frac{\partial}{\partial w_0} J_{MSE}(w) = \frac{1}{N} \sum_{i=1}^{N} ((w_0 + w_1 x^{(i)}) - y^{(i)}) \quad (x_0 = 1)$$

$$\frac{\partial}{\partial w_1} J_{MSE}(w) = \frac{1}{N} \sum_{i=1}^{N} ((w_0 + w_1 x^{(i)}) - y^{(i)})(x^{(i)})$$

Normal Equation: $w = (X^T X)^{-1} X^T y$.

Logistic Regression

Sigmoid Function: $\sigma(x) = \frac{1}{1+e^{-x}}$.
Binary Cross Entropy Loss:

$$BCE(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \tag{3}$$

Classification Metrics

- **Accuracy**: $\frac{TP+TN}{TP+FN+FP+TN}$.
- **Precision**: $\frac{TP}{TP+FP}$.
- **Recall**: $\frac{TP}{TP+FN}$.
- **F1-score**: $\frac{2}{\frac{1}{P}+\frac{1}{R}}$.

Decision Trees

```
DTL(examples, attributes, default):
    if (examples = $\emptyset$): return default
    if ($\forall e \in example$, e has the same classification $c$): return c
    if (attributes = $\emptyset$): return mode(examples)

    best = choose_attribute(attributes, examples)
    tree = new decision tree with root test $best$
    for $v_i$ of $best$ do:
        examples_i = $\{e | e \in examples, e.best = v_i\}$
        subtree = DTL(examples, attributes \ best,
                      mode(examples))
        tree.add($v_i$: subtree)


choose_attribute(attributes, examples):
    best_gain = -$\infty$
    best_attr = None

    for attr in attributes:
        gain = information_gain(attr, examples)
        if gain > best_gain:
            best_gain = gain
            best_attr = attr
    return best_attr
```

For data set contains boolean outputs,

$$I(P(+), P(-)) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

where $0 \leq \mathbb{R}_I \leq 1$. However for non-binary variables the entropy can be greater than 1

Information gain

Information gain = entropy of this node - entropy of children nodes

$$IG(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - remainder(A)$$

Initial $I = 1$

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i})$$

Feature Transformation

Modify the original features of a dataset to make them more suitable for modeling.

- Feature engineering
  - Polynomial features: $z = x^k, k$ is the polynomial degree.
  - log feature: $z = \log(x)$
  - Exp. feature: $z = e^x$
- Feature scale
  - Min-max scaling: $z_i = \frac{x_i - min(x_i)}{max(x_i) - min(x-i)}$, scales to [0, 1]
  - standardization: $z_i = \frac{x_i - \mu_i}{\sigma_i}$, transformed data has mean of 0 and SD of 1
  - robust scaling (not in syl)
- Feature encoding (not in syl)

Normal equation

$$w = (X^T X)^{-1} X^T y$$

Gradient Descent

```
    def gradient_descent_multi_variable(X, y, lr = 1e-5,
     number_of_epochs = 250):
        bias:number = 0
        weights = np.full((X.shape[1], 1), 0).astype(float)
        loss:List[number] = []
        N:number = X.shape[0]
        pred = X @ weights + bias
# pred:$\mathbb{R}^{m\times 1}$ = $X:\mathbb{R}^{m\times n} \times w:\mathbb{R}^{n\times 1}$+ bias:number
        for e in range(number_of_epochs):
            pred = X @ weights + bias
            g_w = (1 / N) * (X.T @ (pred - y))
# $\frac{\partial}{\partial w_1} J_{MSE}(w) = \frac{1}{N} X^T((Xw + bias) - y)$
            g_b = (1 / N) * np.sum(pred - y)
# $\frac{\partial}{\partial w_0} J_{MSE}(w) = \frac{1}{N} \sum_{i=1}^{N}((Xw + bias) - y)$
            bias -= lr * g_b
# $w_0 \leftarrow w_0 - \gamma \frac{\partial J_{MSE}(w)}{\partial w_0}$
            weights -= lr * g_w
# $w_1 \leftarrow w_1 - \gamma \frac{\partial J_{MSE}(w)}{\partial w_1}$
            loss.append(mean_squared_error(y, pred))
        return bias, weights, loss
```

Multiclass Classification

One-vs-One: Train $C(C-1)/2$ classifiers. **One-vs-Rest:** Train C classifiers.

Binary Cross Entropy

$$BCE(y, \hat{y}) = -ylog(\hat{y}) - (1-y)log(1-\hat{y})$$

$$J_{BCE}(w) = \frac{1}{N} \sum_{i=1}^{N} BCE(y^{(i)}, h_w(x^{(i)}))$$

$\sigma(x) = \frac{1}{1+e^{-x}}$ is Not convex, but $-log(\sigma(x))$ is convex
Hypothesis function

$$h_w(x) = \sigma(w_0, w_1 x_1 + w_2 x_2)$$

Weight Update

$$w_j \leftarrow w_j - \gamma \frac{\partial J_{BCE}(w_0, w_1, \dots)}{\partial w_j}$$

Loss function derivative

$$\frac{\partial J_{BCE}(w_j)}{\partial w} = \frac{1}{N} \sum_{i=1}^{N} (h_w(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

0.1 Logistic regression cost function

$$\begin{cases} -log(h_w(x)), \text{ if y = 1} \\ -log(1 - h_w(x)), \text{if y = 0} \end{cases} \tag{4}$$

Generalization

- In supervised learning, and machine learning in general, the model's performance on unseen data is all we care about. This ability to perform well on new, unseen data is known as the model's generalization capability.
- Measuring a model's error is a common practice to quantify the performance of the model. This error, when evaluated on unseen data, is known as the generalization error.
- There are two factors that affect generalization:
  - Dataset quality
    - Relevance: Dataset should contain relevant data, i.e., features that are relevant for solving the problem.
    - Noise: Dataset may contain noise (irrelevant or incorrect data), which can hinder the model's learning process and reduce generalization.
    - Balance (for classification): Balanced datasets ensure that all classes are adequately represented, helping the model learn to generalize well across different classes.
  - Data quantity
    - In general, having more data typically leads to better model performance, provided that the model is expressive enough to accurately capture the underlying patterns in the data
    - Extreme case: if the dataset contains every possible data point, the model would no longer need to "guess" or make predictions. Instead, it would only need to simply memorize all the data!
  - Model complexity
    - Refers to the size and expressiveness of the hypothesis class.
    - Indicates how intricate the relationships between input and output variables that the model can capture are.
    - Higher model complexity allows for more sophisticated modeling of input-output relationships

Hyperparameter

Hyperparameters are settings that control the behavior of the training algorithm and model but are not learned from the data. They need to be set before the training process begins. Such as

- Learning rate
- Feature transformations
- Batch size and iterations in mini-batch gradient descent