Machine Learning Algorithms Deep Dive

Codeium

2025-02-15

Machine Learning Algorithms Deep Dive

1. Support Vector Machine (SVM)

Linear SVM

```
Decision Boundary: w·x + b = 0
Margin Constraints:
  Positive class: w·x + b   1
  Negative class: w·x + b   -1

Optimization Problem:
  Minimize: (1/2)||w||²
  Subject to: y_i(w·x_i + b)   1

Soft Margin SVM (C parameter):
  Minimize: (1/2)||w||² + C
  Subject to: y_i(w·x_i + b)   1 -
```

Kernel SVM

```
Kernel Functions:
1. Linear: K(x,y) = x·y
2. Polynomial: K(x,y) = ( x·y + r)^d
3. RBF: K(x,y) = exp(- ||x-y||²)
4. Sigmoid: K(x,y) = tanh( x·y + r)

Decision Function:
f(x) = sign( ( _i y_i K(x_i,x)) + b)
```

SVM Hyperparameters

```
C: Regularization parameter
  Small C → Larger margin, more violations
  Large C → Smaller margin, fewer violations

  (gamma): Kernel coefficient
  Small   → Larger influence radius
  Large   → Smaller influence radius
```

2. Gradient Descent

Types of Gradient Descent

Batch Gradient Descent

```
For all parameters  :
  =  -  × ( J/ )

Update using entire dataset
Memory: O(n)
```

Stochastic Gradient Descent (SGD)

```
For each training example i:
  =  -  × ( J_i/ )

Update using single example
Memory: O(1)
```

Mini-batch Gradient Descent

```
For each mini-batch B:
  =  -  × ( J_B/ )

Update using batch of b examples
Memory: O(b)
```

Learning Rate Schedules

```
1. Time-based decay:
    (t) =  /(1 + kt)

2. Step decay:
    (t) =   × 0.1^ t/d

3. Exponential decay:
    (t) =   × e^(-kt)
```

Gradient Descent Variants

```
1. Momentum:
   v(t) =  v(t-1) + (1- ) J( )
      =  - v(t)

2. RMSprop:
   s(t) =  s(t-1) + (1- )( J( ))²
      =  -  J( )/√(s(t) +  )

3. Adam:
   m(t) =  m(t-1) + (1- ) J( )
   v(t) =  v(t-1) + (1- )( J( ))²
      =  -  × m(t)/(√v(t) +  )
```

3. Naive Bayes

Types of Naive Bayes

Gaussian Naive Bayes

```
P(x_i|y) = (1/√(2 ²_y))exp(-(x_i- _y)²/(2 ²_y))

For continuous features:
 _y = mean of x for class y
 ²_y = variance of x for class y
```

Multinomial Naive Bayes

```
P(x_i|y) = (count(x_i,y) +  )/(count(y) +  n)

For discrete features (e.g., text):
  = smoothing parameter (Laplace smoothing)
n = number of features
```

Bernoulli Naive Bayes

```
P(x_i|y) = P(i|y)^x_i × (1-P(i|y))^(1-x_i)

For binary features:
P(i|y) = probability of feature i appearing in class y
```

Naive Bayes Decision Rule

```
ŷ = argmax_y P(y) P(x_i|y)

In log space (to prevent underflow):
ŷ = argmax_y log(P(y)) +  log(P(x_i|y))
```

4. K-Means Clustering

Algorithm Steps

```
1. Initialize k centroids randomly
2. Repeat until convergence:
   a. Assign points to nearest centroid
   b. Update centroids as mean of assigned points

Assignment step:
c_i = argmin_j ||x_i -  _j||²

Update step:
 _j = (1/|S_j|) (x_i) for x_i in cluster j
```

Initialization Methods

```
1. Random Initialization:
   Select k points randomly

2. K-means++:
   a. Choose first centroid randomly
   b. For remaining k-1 centroids:
      P(x)  min(D(x)²) to all centroids
```

Choosing K

```
Elbow Method:
Plot inertia vs k
Inertia =  min||x_i -  _j||²

Silhouette Score:
s(i) = (b(i) - a(i))/max(a(i), b(i))
where:
a(i) = mean intra-cluster distance
b(i) = mean nearest-cluster distance
```

5. Polynomial Regression

Model Form

```
y =   +  x +  x² + ... +  x  +

Matrix form:
X = [1  x  x²  ...  x ]
  = [      ...  ]
y = X  +
```

Feature Generation

```
Original: x
Polynomial: [1, x, x², x³, ..., x ]

Example (degree=2):
x = [1, 2, 3]
X = [[1, 1, 1],
     [1, 2, 4],
     [1, 3, 9]]
```

Regularization

```
Ridge (L2):
min ||y - X ||² +  || ||²

Lasso (L1):
min ||y - X ||² +  | |
```

Avoiding Overfitting

1. Cross-validation for degree selection
2. Feature scaling crucial

   ```
   x_scaled = (x -  )/
   ```

3. Regularization parameter tuning

6. Common Implementation Tips

Feature Scaling

```
For all algorithms except Naive Bayes:
- StandardScaler
- MinMaxScaler
- RobustScaler
```

Hyperparameter Selection

```
SVM:
- C: [0.1, 1, 10, 100]
- gamma: ['scale', 'auto', 0.1, 0.01]
- kernel: ['rbf', 'linear', 'poly']

K-Means:
- n_clusters: [2-10]
- init: ['k-means++', 'random']
- n_init: [10, 20, 30]

Polynomial Regression:
- degree: [1-5]
- alpha (regularization): [0.001, 0.01, 0.1, 1]
```

Performance Metrics

```
Clustering:
- Silhouette Score
- Calinski-Harabasz Index
- Davies-Bouldin Index

Regression:
- R² Score
- MSE/RMSE
- MAE

Classification:
- Accuracy
- Precision/Recall
- F1 Score
- ROC-AUC
```

Remember: - Always scale features (except for Naive Bayes) - Use cross-validation - Consider computational complexity - Monitor for overfitting - Validate assumptions