Cheat Sheet

1. Entropy and Information Gain

- **Entropy (H)**: Measures uncertainty in data.

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

  Where p_i is the probability of class i.
- **Conditional Entropy**:

$$H(S|T) = \sum_{t \in T} P(t) H(S_t)$$

- **Information Gain (IG)**: Used for feature selection in decision trees.

$$IG(S, A) = H(S) - H(S|A)$$

2. ID3 Algorithm

- **Steps**:
  1. Compute entropy for target attribute.
  2. Calculate information gain for each feature.
  3. Split on the feature with the highest information gain.
  4. Recursively split until leaves are pure.
- **Pruning**: Remove branches that don't improve accuracy using validation data.

3. k-NN Algorithm

- **Distance Calculation (Euclidean)**:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{i,j})^2}$$

- **1-NN**: Classify based on the closest training instance.
- **k-NN**: Classify based on majority label of the k nearest neighbors.

4. Cross Validation

- **k-fold Cross Validation**: Split data into k folds, train on $k-1$ folds, test on the remaining one.
- Average error from all folds is the estimated performance.

5. Neural Networks and Perceptrons

- **Perceptron**: A linear classifier.

$$O = g\left(\sum_{i=1}^{n} w_i x_i\right), \quad g(h) = \text{step function}$$

- **Delta Rule** (for weight update):

$$\Delta w_i = \eta (y - O) x_i$$

  Where η is the learning rate.
- **Multi-Layer Perceptron (MLP)**: Contains input, hidden, and output layers.

6. Backpropagation

- **Forward Propagation**: Calculate output layer activations.
- **Error Calculation**:

$$E = \frac{1}{2} \sum (y_i - O_i)^2$$

- **Weight Update**:

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta \frac{\partial E}{\partial w_{ij}}$$

- **Sigmoid Activation Function**:

$$g(h) = \frac{1}{1 + e^{-h}}, \quad g'(h) = g(h)(1 - g(h))$$

7. Steepest Descent

- **Gradient Descent Update Rule**:

$$x_{new} = x_{old} - \eta \nabla f(x)$$

- If derivatives are unavailable, approximate using finite differences:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_i + \delta) - f(x_i - \delta)}{2\delta}$$

8. CNF and DNF in Neural Networks

- **CNF (Conjunctive Normal Form)**: AND of ORs.
- **DNF (Disjunctive Normal Form)**: OR of ANDs.
- Perceptrons can compute 1-term CNF or DNF, but cannot compute XOR.

9. Cross-Entropy and Softmax

- **Cross-Entropy (Binary)**:

$$H(p,q) = -\left[p \log q + (1-p)\log(1-q)\right]$$

- **Softmax Function** (for multi-class classification):

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

10. Deep Learning

- **Motivation**: Deep networks allow learning of hierarchical representations. Lower layers learn simple patterns, while higher layers capture more abstract features.
- **Enhancements**:
  - ReLU activation function: $g(h) = \max(0, h)$
  - Dropout: Randomly drop neurons during training to reduce overfitting.
  - ADAM optimization: Adaptive learning rate based on first and second moments.

11. ADAM Optimization Algorithm

- **ADAM Update Rule**:

$$\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\
W_t &= W_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}
\end{aligned}$$

12. Batch Normalization

- **Batch Normalization Formula**:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \quad y_i = \gamma \hat{x}_i + \beta$$

  Where μ_B and σ_B^2 are the mean and variance of the batch.

13. Image-Related Neural Network Layers

- **Convolution (Conv2D)**:

$$O_{ij} = \sum_{m,n} I_{i+m,j+n} K_{m,n}$$

  Where I is the input image, and K is the convolution kernel.
- **Max Pooling (MaxPool2D)**: Reduces the size of feature maps by taking the maximum value in each pool.

14. Overfitting and Regularization

- **L2 Regularization (Weight Decay)**:

$$E_r(W) = E(W) + \lambda \sum_i w_i^2$$

- **L1 Regularization**:

$$E_r(W) = E(W) + \lambda \sum_i |w_i|$$

- **Dropout**: Randomly drop neurons during training with probability p, and scale weights during testing by multiplying by p.