1 Introduction

- Variance: Sensitivity to changes in training data. Bias: Difference between average prediction and actual answer.
- Lazy Learner: Stores training examples and postpones generalising beyond these data until an explicit request is made at test time.
- Eager Learner: Constructs a general, explicit description of target function based on the provided training examples.

2 k-Nearest Neighbours

- Distances: Manhattan (L_1) $= \sum_{d=1}^{D} \left| x_d^{(i)} - x_d^{(q)} \right|$. Euclidean ($L_2$) $= \sqrt{\sum_{d=1}^{D} (x_d^{(i)} - x_d^{(q)})^2}$. Chebyshev ($L_\infty$) $= \max_{d=1}^{D} \left| x_d^{(i)} - x_d^{(q)} \right|$.
- Distance weighted k-NN: Value of k is minor importance as distant neighbours have small weights.
- Weights: Inverse of distance $= \frac{1}{d(x^{(i)}, x^{(q)})}$. Gaussian distribution: $\frac{1}{2\pi} e^{-\frac{d(x^{(i)}, x^{(q)})^2}{2}}$

3 Decision Tree

- Entropy: $H(x) = -\sum_k^K P(x_k) \log_2 P(x_k)$. 0 if all data points in 1 group and 1 if data points split evenly between 2 groups.
- Information Gain: $IG(dataset, subsets) = H(dataset) - \sum_{S \in subsets} \frac{|S|}{|dataset|} H(S)$, $|dataset| = \sum_{S \in subsets} |S|$
- For binary tree, $IG(dataset, subsets) = H(dataset) - (\frac{|S_{left}|}{|dataset|} H(S_{left}) + \frac{|S_{right}|}{|dataset|} H(S_{right}))$, $|dataset| = |S_{left}| + |S_{right}|$
- Ordered Values: For each feature, sort its values and consider only split points that are between 2 data points with different labels.
- Categorical/Symbolic Values: Search for most informative feature and create as many branches as there are different values.
- Overfitting: Early stopping (max depth, min examples) or Pruning (remove internal nodes connected to only leaf nodes if accuracy on validation set increases if node is removed)

4 Evaluation

- Test dataset must NEVER be used to train the model or for hyperparameter tuning.
- Hyperparamater: Model parameters that are chosen before the training. Selected based on accuracy of the validation dataset.
- Held-out test set (Plenty of data): Train on training set, tune hyperparameters on validation set, estimate performance on test set.
- Cross-validation (Limited data): Seperate data into k folds, use 1 fold for testing and k-1 folds for testing+validation, repeat k times using each fold as test set, estimate performance by averaging results across all test folds.

	X Predicted	Y Predicted
X Actual	True Positive	Flase Negative
Y Actual	False Positive	True Negative

Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$, Precision $= \frac{TP}{TP+FP}$, Recall $= \frac{TP}{TP+FN}$, $F_1 = \frac{2 \cdot precision \cdot recall}{precision+recall}$, $F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision)+recall}$

- Macro Averaging: Average on class level. $P_{macro} = \frac{1}{3} \cdot (\frac{1}{3} + \frac{1}{1} + \frac{1}{2})$. Micro Averaging: Average on item level. $P_{micro} = \frac{1+1+1}{3+1+2}$
- Mean Square Error (MSE): $\frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2$. Root Mean Square Error (RMSE) $= \sqrt{MSE}$.
- Imbalance test set: Normalise each row to sum to 1.
- Overfitting: Good performance on training data, poor generalisation to other data. High variance and low bias.
- Overfitting Causes: Model too complex, training set not representative of all possible data, learning performed too long.
- Overfitting Solutions: Use right level of complexity, get more data, stop training earlier, drop out, regularisation.
- Underfitting: Poor performance on training data, poor generalisation to other data. Low variance and high bias.
- True Error: Probability it will misclassify a random example x from Distribution D. $error_D(h) = Pr[f(x) \neq h(x)]$.
- Sample Error: $error_S(h) = \frac{1}{N} \sum_{x \in S}$(1 if f(x)=h(x) else 0).
- Confidence Interal $= error_S(h) \pm Z_N \sqrt{\frac{error_S(h) \cdot (1-error_S(h))}{n}}$, where n is the number of test datapoints and Z_n the desired confidence.

N%	50%	68%	80%	90%	95%	98%	99%
Z_n	0.67	1.00	1.28	1.64	1.96	2.33	2.58

- Statistical Significance: Less than p% chance performance difference due to sampling noise and systems actually comparable.

5 Neural Networks

- Linear Regression: $y = ax+b$. $E = \frac{1}{2} \sum_{i=1}^{N} (\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{2} \sum_{i=1}^{N} (ax^{(i)} + b - y^{(i)})^2$. $\frac{\delta E}{\delta a} = \sum_{i=1}^{N} (\hat{y}^{(i)} - y^{(i)}) x^{(i)}$. $\frac{\delta E}{\delta b} = \sum_{i=1}^{N} (\hat{y}^{(i)} - y^{(i)})$.
- Gradient Descent: $a := a - \alpha \sum_{i=1}^{N} (ax^{(i)} + b - y^{(i)}) x^{(i)}$. $b := b - \alpha \sum_{i=1}^{N} (ax^{(i)} + b - y^{(i)})$.
- Analytical Single-Variable GD: $X = \begin{bmatrix} x^1 & 1.0 \\ \vdots & \vdots \\ x^N & 1.0 \end{bmatrix}$ $y = \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix}$ $\theta = \begin{bmatrix} a \\ b \end{bmatrix}$, $\nabla_\theta E(\theta) = X^T (X\theta - y) = 0 \implies \theta^* = (X^T X)^{-1} X^T y$
- GD (whole dataset) vs Stochastic GD (each datapoint) vs Mini-batched GD (batch of datapoint)
- Normalisation: Min-max ($X = a + \frac{(X - X_{min})(b-a)}{X_{max} - X_{min}}$), Standardisation ($X = \frac{X - \mu}{\sigma}$). Normalise on training set only.

- L2 regularisation: $J(\sigma) = Loss(y, \hat{y}) + \lambda \sum_w w^2$, $w \leftarrow w - \alpha(\frac{\delta Loss}{\delta w} + 2\lambda w)$
- L1 regularisation: $J(\sigma) = Loss(y, \hat{y}) + \lambda \sum_w |w|$, $w \leftarrow w - \alpha(\frac{\delta Loss}{\delta w} + \lambda sign(w))$

Perceptron: only learn linearly separable functions and is non-differentiable (not suitable for neural networks).

$$h(x) = f(W^T x) = \begin{cases} 1 & \text{if } W^T x > 0 \\ 0 & \text{otherwise} \end{cases} \text{ with learning rule } \theta_i \leftarrow \theta_i + \alpha(y - h(x))x_i$$

Activation Function	Formula	Range	Derivitive	Notes
Linear	x	$(-\infty, \infty)$	1	
Logistic/Sigmoid	$\frac{1}{1+e^{-z}}$	$(0, 1)$	$g(z)(1-g(z))$	
Tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$(-1, 1)$	$1 - g(z)^2$	Steeper Gradient
ReLU	x if $x > 0$ else 0	$(0, \infty)$	1 for $x > 0$ else 0	
Softmax	$\frac{e^{z_i}}{\sum_k e^{z_k}}$	(0,1]		Sums to 1

Activation Function	Formula	Derivative
MSE	$\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$	$\frac{1}{N}\sum_{i=1}^{N} 2(\hat{y} - y)$
Binary Cross-entropy	$-\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)}))$	
Categorical Cross-entropy	$-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}(y_c^{(i)}\log(\hat{y}^{(i)})$	

Type	Output Activation	Loss
Regression	Linear	MSE
Binary	Sigmoid	Binary Cross-entropy
Multi-Class	Softmax	Categorical Cross-entropy
Multi-Label	Sigmoid	Binary Cross-entropy

For $Z = XW + B$ and $A = g(Z)$,

1. $\frac{\delta Loss}{\delta Z} = \frac{\delta Loss}{\delta A} \circ g'(Z)$, where $\circ$ is the element wise multiplication
2. For softmax with cross entropy loss, $\frac{\delta L}{\delta Z} = \frac{1}{N}(\hat{y} - y)$, where y is a matrix.
3. $\frac{\delta Loss}{\delta X} = \frac{\delta Loss}{\delta Z} \cdot \frac{\delta Z}{\delta X} = \frac{\delta Loss}{\delta Z} \cdot W^T$
4. $\frac{\delta Loss}{\delta W} = X^T \cdot \frac{\delta Loss}{\delta Z}$. Then $W = W - \alpha \frac{\delta L}{\delta W}$.
5. $\frac{\delta Loss}{\delta b} = 1^T \cdot \frac{\delta Loss}{\delta Z}$, where 1 is a column vector of ones.

Partial derivitives for vectors & matrices:

1. $z = Wx \Leftrightarrow \frac{\delta z}{\delta z} = W$
2. $z = x \Leftrightarrow \frac{\delta z}{\delta z} = I$
3. $z = xW \Leftrightarrow \frac{\delta z}{\delta z} = W^T$
4. $z = Wx, \Delta = \frac{\delta J}{\delta z} \Leftrightarrow \frac{\delta J}{\delta W} = \Delta^T x$
5. $z = xW, \Delta = \frac{\delta J}{\delta z} \Leftrightarrow \frac{\delta J}{\delta W} = x^T \Delta$

6 Unsupervised Learning

- K Means: Simple, Popular and Efficient (O(TKN) with $K, T << N$) but have to define k and only for hyper-ellipsoids.
- Assignment: $c^{(i)} = argmin_{k\in\{1,\ldots,K\}}||x^{(i)} - \mu_k||^2$. Update: $\mu_k = \frac{\sum_{i=1}^{N} 1(c^{(i)}=k)\cdot x^{(i)}}{\sum_{i=1}^{N} 1(c^{(i)}=k)}$. Convergence: $\forall_k \left|\mu_k^t - \mu_k^{(t-1)}\right| < \epsilon$.
- Selecting K: Elbow method - select K where the rate of decrease sharply shifts, or cross validation.
- Normal Distribution: $N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{(x-\mu)^2}{2\sigma^2}$ or $N(x|\mu, \Sigma^2) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} \exp -\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)$
- Kernel/Parzen Density Estimation: Non-Gaussian vs Gaussian

$$\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h^D}H(\frac{x - x^{(i)}}{h}), H(u) = \begin{cases} 1 & |u_j| < \frac{1}{2}; j = 1, \ldots, D \\ 0 & otherwise \end{cases} \text{ or } \hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi h^2)^{\frac{D}{2}}} \exp -\frac{||x - x^{(i)}||^2}{2h^2}$$

$$L = -\log p(\chi|\theta) = -\sum_{i=1}^{N}\log p(x^{(i)}|\theta) = \frac{N}{2}\log 2\pi + \frac{N}{2}\log\sigma^2 + \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x^{(i)} - \mu)^2, \text{ lower is better}$$

- Gausian Mixture Model:

$$p(x|\theta) = \sum_{k=1}^{K}\pi_k N(x|\mu_k, \Sigma_k)$$

$$r_{ik} = \frac{\pi_k N(x^{(i)}|\mu_k, \Sigma_k)}{\sum_j^K \pi_j N(x^{(i)}|\mu_j, \Sigma_j)} \quad N_k = \sum_{i=1}^{N} r_{ik} \quad \hat{\mu}_k = \frac{1}{N_k}\sum_{i=1}^{N} r_{ik}x^{(i)} \quad \Sigma_k = \frac{1}{N_k}\sum_{i=1}^{N} r_{ik}(x^{(i)} - \hat{\mu}_k)(x^{(i)} - \hat{\mu}_k)^T \quad \pi_k = \frac{N_k}{N}$$

- Bayesian Information Criterion: $BIC_K = L(K) + \frac{P_K}{2}log(N)$, P = no. of parameters and N = no. of examples. For 2D: $P_K = 6K - 1$ (2 for mean, 3 for covariance, 1 for mixing proportion), where K = no. of components.

7 Evolutionary Algorithm

- Genetic Algorithm: Selection (Biased Roulette Wheel/Tournament + Elitism), Cross-over, Mutation
- Evolutionary Strategies: Select best μ from $\mu + \lambda$ individuals, mutate $y_i = x_rand + N(0, \sigma)$ then union parents and offspring.
- Novelty Search: $Novelty(x) = \frac{1}{N}\sum_{k=0}^{N} d_i(x)$. Requires behavioural descriptors. Optimises for novelty instead of quality.
- MAP-Elite: Discretise the behavioural descriptors into cells and keep best solution for each cell. Finds diverse yet high performing solutions. Diversity (archive size/max size), Performance (mean or max fitness), QD-Score (sum of fitness of all solutions in archive)