

# General Notes

CS, ML and Stats

Patrick Daly

# Contents

<b>1</b>	<b>Computer Science</b>	<b>4</b>
1.1	Algorithms . . . . .	4
1.2	Data Structures . . . . .	9
1.3	Linux . . . . .	9
<b>2</b>	<b>Machine Learning</b>	<b>9</b>
2.1	Supervised . . . . .	9
2.1.1	Ordinary Least Squares (OLS) . . . . .	9
2.1.2	Generalized Linear Model(GLM) . . . . .	9
2.1.3	Logistic Regression . . . . .	9
2.1.4	Linear Discriminant Analysis . . . . .	9
2.1.5	Support Vector Machines . . . . .	9
2.1.6	K-Nearest Neighbors . . . . .	9
2.1.7	Gaussian Process . . . . .	9
2.1.8	K-Nearest Neighbors . . . . .	9
2.1.9	Decision Trees . . . . .	9
2.1.10	Random Forest . . . . .	9
2.1.11	Gaussian Process . . . . .	9
2.1.12	Naive Bayes . . . . .	9
2.1.13	Kalman Filter? dunno where this should go yet... maybe estimation section? . . . . .	9
2.2	Unsupervised . . . . .	9
2.2.1	Gaussian Mixture Models . . . . .	9
2.2.2	K-Means . . . . .	9
2.2.3	Density-Based Spatial Clustering of Applications with Noise (DBSCAN) . . . . .	9
2.2.4	Spectral Clustering . . . . .	9
2.2.5	Hierarchical Clustering . . . . .	9
2.2.6	Factor Analysis . . . . .	9
2.2.7	Independent Component Analysis . . . . .	9
2.2.8	Principal Component Analysis . . . . .	9
2.2.9	Non-Negative Matrix Factorization (NMF) . . . . .	9
2.2.10	Latent Dirichlet Allocation (LDA) . . . . .	9
2.2.11	Outlier Detection? . . . . .	9
<b>3</b>	<b>Deep Learning</b>	<b>9</b>
3.1	Convolutional Networks . . . . .	9
3.2	Recurrent Networks . . . . .	9

3.3	Long Short-Term Memory (LSTM)	9
3.4	Autoencoders	9
3.5	Reinforcement Learning	9
<b>4</b>	<b>Linear Algebra</b>	<b>9</b>
4.1	Norms	9
4.1.1	Euclidean / Frobenius	9
4.1.2	Manhattan	9
4.1.3	Infinity	9
4.1.4	Nuclear	9
4.1.5	Spectral	9
4.1.6	Symmetric	9
4.1.7	Positive Definite	9
4.1.8	Positive Semi-Definite	9
4.1.9	Negative Definite	9
4.1.10	Negative Semi-Definite	9
4.2	Eigendecomposition	9
4.3	Singular Value Decomposition (SVD)	9
4.4	Principal Component Analysis (PCA)	9
4.5	Independent Component Analysis (ICA)	9
4.6	Canonical Component Analysis (CCA)	9
4.7	Factor Analysis	9
<b>5</b>	<b>Statistics</b>	<b>9</b>
5.1	Probability Theory	9
5.2	Distributions	9
5.3	Combinatorics	9

# 1 Computer Science

## 1.1 Algorithms

**DFS** Time:  $O(n)$ , Space:  $O(n)$   
Solution exists far away.

Recursive

```
1 def dfs(node):
2     if node:
3         # do stuff if pre-order
4         if node.left:
5             dfs(node.left)
6         # do stuff if in-order
7         if node.right:
8             dfs(node.right)
9         # do stuff if post-order
```

Iterative

```
1 def dfs(node): # if bst, may need to swap search left/right
2     visited = set()
3     stack = [node]
4     while stack:
5         current = stack.pop(-1)
6         print(current.val)
7         if current not in visited:
8             visited.add(current)
9             if current.left and current.left not in visited:
10                stack.append(current.left)
11            if current.right and current.right not in visited:
12                stack.append(current.right)
13    return visited
```

**BFS** Time:  $O(n)$ , Space:  $O(n)$   
Solution exists nearby.

Iterative

```
1 def bfs(node):
```

```

2     stack = [node]
3     while stack:
4         current = stack.pop(-1)
5         if current.left:
6             stack.append(current.left)
7         if current.right:
8             stack.append(current.right)

```

Mergesort Time:  $O(n \log n)$ , Space:  $O(n)$

```

1 def mergesort(array, start, end):
2     if start < end:
3         mid = (start+end) // 2
4         mergesort(array, start, mid)
5         mergesort(array, mid+1, end)
6         merge(array, start, mid, end)

1 def merge(array, start, mid, end):
2     left = array[start: mid+1]
3     right = array[mid+1: end+1]
4     i, j, k = 0, 0, start
5     while i < len(left) and j < len(right):
6         if left[i] < right[j]:
7             array[k] = left[i]
8             i += 1
9         else:
10            array[k] = right[j]
11            j += 1
12            k += 1
13    if j == len(right):
14        array[k: end+1] = left[i:]

1 def binarysearch(array, val, low, high):
2     if high < low:
3         # can also return high or low index
4         return 'Not found!'
5     mid = (low + high) // 2
6     if array[mid] > val:
7         return binarysearch(array, val, low, mid-1)

```

```

8     elif array[mid] < val:
9         return binarysearch(array, val, mid+1, high)
10    return mid

```

**Greatest Common Divisor (GCD)** Time: ?, Space: ?

Euclid: Time:  $O(\ln^2 \min(a, b))$ , Space:  $O(1)$

```

1 def gcd(a, b):
2     return gcd(b, a % b) if b else a

```

Stein: Time: 60% faster than Euclid in theory, similar in practice Space:  $O(1)$

```

1 def stein(a, b):
2     # binary gcd
3     if a == b: return a
4     if a == 0: return v
5     if b == 0: return a
6     if ~a & 1: # a is even
7         if b & 1: # b is odd
8             return stein(a >> 1, v)
9         else: # b is even
10            return stein(a >> 1, b >> 1) << 1
11    else:
12        if ~b & 1: # b is even
13            return stein(a, b >> 1)
14        else: # b is odd
15            if a > b:
16                return stein((a-b) >> 1, b)
17            else:
18                return stein(u, (b-a) >> 1)

```

**Fibonacci** Time: ?, Space: ?

Iterative

```

1 def fib_iterative(n):
2     if n == 0: return 0
3     if n == 1: return 1
4     fibs = [0]*(n+1)
5     fibs[0] = 0

```

```
6     fibs[1] = 1
7     for i in range(2, n+1):
8         fibs[i] = fibs[i-2] + fibs[i-1]
9     return fibs[n]
```

Recursive

```
1 def fib_recursive(n, dp={0: 0, 1: 1}):
2     if n in dp:
3         return dp[n]
4     return fib_recursive(n-2, dp) + fib_recursive(n-1, dp)
```





## 1.2 Data Structures

## 1.3 Linux

# 2 Machine Learning

## 2.1 Supervised

### 2.1.1 Ordinary Least Squares (OLS)

### 2.1.2 Generalized Linear Model (GLM)

### 2.1.3 Logistic Regression

### 2.1.4 Linear Discriminant Analysis

### 2.1.5 Support Vector Machines

### 2.1.6 K-Nearest Neighbors

### 2.1.7 Gaussian Process

### 2.1.8 K-Nearest Neighbors

### 2.1.9 Decision Trees

### 2.1.10 Random Forest

### 2.1.11 Gaussian Process

### 2.1.12 Naive Bayes

### 2.1.13 Kalman Filter? dunno where this should go yet... maybe estimation section?

## 2.2 Unsupervised

### 2.2.1 Gaussian Mixture Models

### 2.2.2 K-Means

### 2.2.3 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

### 2.2.4 Spectral Clustering

### 2.2.5 Hierarchical Clustering

### 2.2.6 Factor Analysis

### 2.2.7 Independent Component Analysis

### 2.2.8 Principal Component Analysis

### 2.2.9 Non-Negative Matrix Factorization (NMF)

### 2.2.10 Latent Dirichlet Allocation (LDA)

### 2.2.11 Outlier Detection?

# 3 Deep Learning