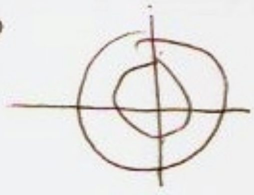
Exame ML 2015/2016

1. - ver lecun-review-sol.pdf

A. 1-NN, nenhum. O 1-NN é caracterizado pela "Voronoi partition of space" que descreve a fronteira produzida pelo 1-NN algorithm.

B - Num Probabilistic Gaussian Discrim com matrix de covariância simétrica resulta numa círculo à volta dos dados

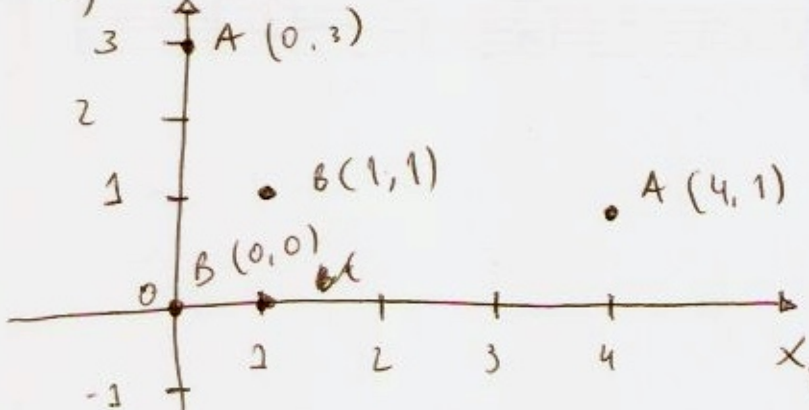


C. Num Probabilistic Gaussian Discriminant, com matrix de covariância diferentes, resulta um elipse à volta do dos dados.

D. Uma decision tree em real-valued inputs cria uma decision boundary que é constituída por retângulos no input space.

2.

a)



Apesar de as classes não serem linearmente separáveis, podia-se utilizar, por exemplo, uma Neural Network, sem hidden layer, com uma função sigmoide na output layer, dado que esta configuração seria um classificador linear, que irá parar de treinar quando o erro chegar a um valor mínimo.

b)
$P \curvearrowright (2; 1)$

Com K=1, a classe de P seria B, dado que o vizinho mais próximo é $B \curvearrowright (1, 1)$.

Com K=3, vizinhos mais próximos: B(1,1), A(4,1), B(0,0), logo a classe seria B $(\underbrace{2B \text{ para } 1A})$

Ganha a maioria!

C. A

$d_1 = \sqrt{(2-0)^2+(1-0)^2} = \sqrt{4+1} = \sqrt{5}$ (P a B(0,0))

$d_2 = \sqrt{(2-0)^2+(1-3)^2} = \sqrt{4+4} = \sqrt{8}$ (P a A(0,3))

$d_3 = \sqrt{(2-4)^2+(1-1)^2} = \sqrt{2^2+0} = 2$ (P a A(4,1))

Com K=10, teríamos como votos: A, B, A, B, B, A, A logo seria classificado como A, dado que existem mais As. Contudo, tem de atentar que o nosso dataset tem apenas 7 pontos e por isso, ao utilizar um K=10, estamos de certa forma a provocar um overfit. Idealmente, o K deve ser inferior ao número de pontos do nosso dataset, para se poder ter uma aproximação o mais fiel possível.

c) Na prática uma regra comum para se escolher o K no algoritmo de K-NN, é fazendo $K = \sqrt{n}$, onde n é o número de instâncias do nosso dataset.

· Em teoria, quando possuímos um número infinito de amostras, quanto maior o valor de K, melhor é a classificação (erro este aproxima-se do Bayes error rate ótimo. O ponto é que todos os K neighbours têm de estar próximos do ponto teste, o que é impossível quando temos um número finito de amostras. Na prática: 1) K deve ser grande para que a taxa de erro seja minimizada (se for demasiado pequeno levará a fronteiras de decisão com muito ruído); 2) K deve ser pequeno o suficiente para que apenas amostras próximas estejam incluídas (um K demasiado grande irá levar a a fronteira demasiado over-smoothed ...). Pode-se por exemplo testar a performance do algoritmo com K-NN diferentes e verificar qual é que ele tem uma melhor performance sem fazer overfit.

d)

Calcular "class means":

$$\bar{A}_{x_1}: \frac{0+0+4+4}{4} : \frac{8}{4} : 2$$

$$\bar{A}_{x_2}: \frac{3+(-3)+1+(-2)}{4} : \frac{-1}{4} : -0.25$$

$\bar{A} \hookrightarrow (2, -0.25)$

$$\bar{B}_{x_1} : \frac{0+0+1}{3} : \frac{1}{3} : 0.33$$

$$\bar{B}_{x_2} : \frac{0+(-2)+1}{3} : -\frac{1}{3} : -0.33$$

$\bar{B} \hookrightarrow (0.33, -0.33)$

Para fazer o arranjo de cada ponto há que calcular a distância a cada classe:

Ponto	$d_{\overline{A}}$; $\overline{A} \curvearrowright (2,-0.25)$	$d_{\overline{B}}$; $\overline{B} \curvearrowright (0.33,-0.33)$
(0, 3)	$\sqrt{(0-2)^2+(3+0,25)^2}$ = 3.81	$\sqrt{(0-0.33)^2+(3+0.33)^2}$ = 3.55 Ⓑ
(0, -3)	$\sqrt{(0-2)^2+(-3+0,25)^2}$ = 3.40	$\sqrt{(0-0.33)^2+(-3+0.33)^2}$ = 2.69 Ⓑ
(4, 1)	$\sqrt{(4-2)^2+(1+0,25)^2}$ = 2.36 Ⓐ	$\sqrt{(4-0.33)^2+(1+0.33)^2}$ = 3.90
(4, -2)	$\sqrt{(4-2)^2+(-2+0,25)^2}$ = 2.66 Ⓐ	$\sqrt{(4-0.33)^2+(-2+0.33)^2}$ = 4.03
(0, 0)	$\sqrt{(0-2)^2+(0+0,25)^2}$ = 2.02	$\sqrt{(0-0.33)^2+(0+0.33)^2}$ = 0.47 Ⓑ
(0, -2)	$\sqrt{(0-2)^2+(-2+0,25)^2}$ = 2.66	$\sqrt{(0-0.33)^2+(-2+0.33)^2}$ = 1.70 Ⓑ
(1, 1)	$\sqrt{(1-2)^2+(1+0.25)^2}$ = 1.60	$\sqrt{(1-0.33)^2+(1+0.33)^2}$ = 1.49 Ⓑ

Atribuir a cada classe o ponto com o valor de distância menor!

3.

a) $J(c) = \sum_{k=1}^{K} \sum_{i=1}^{N} r_{ik} \| x_i - \mu_k \|^2$

1ª Iteração:

Centroids e Distân.

Ponto	$c_1 (2, -0.25)$	$c_2 (1/3, -\frac{1}{3})$
$(0, 3)$	3.81	3.35 $\boxed{c_2}$
$(0, -3)$	3.40	2.69 $\boxed{c_2}$
$(4, 1)$	2.36 $\boxed{c_1}$	3.90
$(4, -2)$	2.66 $\boxed{c_1}$	4.03
$(0, 0)$	2.02	0.47 $\boxed{c_2}$
$(0, -2)$	2.66	1.70 $\boxed{c_2}$
$(1, 1)$	1.60	1.49 $\boxed{c_2}$

Nesta iteração, a função de custo tem o seguinte valor:

$$J(c) = \|2.36\|^2 + \|2.66\|^2 + (3.35)^2 + (2.69)^2 + (0.47)^2 + (1.70)^2 + (1.49)^2 = \underline{\underline{36.4348}}$$

2ª Iteração

1º Passo: Calcular Novos centróides:

$C_{1_{x_1}} = \frac{4 + 4}{2} = 4$, $C_{1_{x_2}} = \frac{-2+1}{2} = -0.5$

$C_1 \to (4, -0.5)$

$C_{2_{x_1}} = \frac{0+0+0+0+1}{5} = 0.20$, $C_{2_{x_2}} = \frac{3-3+0-2+1}{5} = -0.20$

2º Passo: Calcular distâncias:

	C_1 (4, -0.5)		C_2 (0.20, -0.20)	
(0, 3)	5.32		3.21	C_2
(0, -3)	4.72		2,81	C_2
(4, 1)	1.5	C_1	3,48	
(4, -2)	1.5	C_1	4,20	
(0, 0)	4.03		0,28	C_2
(0, -2)	4.27		1.81	C_2
(1, 1)	3.35		1.44	C_2

cálculo do valor de $J(c)$, nesta iteração:

$$J(c) = (1.5)^2 + (1.5)^2 + (3.21)^2 + (2.81)^2 + (0.28)^2 + (1.81)^2$$

$$+ (1.44)^2 = 28.1283$$

3)

The $$\mathcal{L}(\Delta) = \sum_{i=1}^{n} \| x_i - \mu_{k(i)} \|^2 = \sum_{k=1}^{K} \sum_{i \in C_k} \| x_i - \mu_k \|^2$$

K-means is fundamentally a coordinate descent algorithm. Coordinate descent serves to minimize a multivariate function along one direction at a time. The inner-loop of k-means repeatedly minimizes the function with respect to k while holding μ fixed and then minimizes with respect to μ while holding k fixed. This means the function must monotonically decrease and that values must converge.

4.

i) False. SVMs will always try to improve the margin between points in the input space, so, it will achieve less accuracy than Perceptron, due to the fact that the Perceptron algorithm only stops until it has correctly classified all the training data. i.e., it overfits in the training data.

ii) False. They actually achieve higher accuracy than Perceptron in the test set, regarding the fact that, during training, the SVM algorithm only maximizes the margin between classes, while the Perceptron overfits on the training set which will lead to a worse performance on the test set. Kernels, however are useful to solve non-linear separation problems, but will not greatly better performance.

5.

a) We can use the normal equations method:

Write the cost function in matrix form:

$$\mathcal{L}(w, s) = \frac{1}{2}\sum_{n=1}^{N}(y_i - f(x_i))^2$$

$$= \frac{1}{2}(Xw - y)^T(Xw - y) = \frac{1}{2}(w^T X^T X w - w^T X^T y - y^T X w + y^T y)$$

To minimize $\mathcal{L}(w, s)$, take derivative and set to zero:

$$\frac{\partial \mathcal{L}}{\partial w} = -X^T y + X^T X w = 0$$

$$\boxed{X^T X w = X^T y}$$

Portanto: $X = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 2 \end{bmatrix}$, $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$, $y = \begin{bmatrix} 2.5 \\ 4 \\ 3.5 \end{bmatrix}$

So, the normal equations:

$$X^T X w = X^T y$$

$$\underbrace{\begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}}_{2\times 3} \underbrace{\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 2 \end{bmatrix}}_{3\times 2} \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{2\times 1} = \underbrace{\begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}}_{2\times 3} \underbrace{\begin{bmatrix} 2.5 \\ 4 \\ 3.5 \end{bmatrix}}_{3\times 1}$$

$$\underbrace{\begin{bmatrix} 6 & 5 \\ 5 & 6 \end{bmatrix}}_{2\times \cancel{4}} \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{\cancel{2}\times 1} = \underbrace{\begin{bmatrix} 14 \\ 13.5 \end{bmatrix}}_{2\times 1}$$

$$\underbrace{\begin{bmatrix} 6w_1 + 5w_2 \\ 5w_1 + 6w_2 \end{bmatrix}}_{2\times 1} = \underbrace{\begin{bmatrix} 14 \\ 13.5 \end{bmatrix}}_{2\times 1}$$

Resolvendo o sistema de Equações:

$$\begin{cases} 6w_1 + 5w_2 = 14 \\ 5w_1 + 6w_2 = 13,5 \end{cases} \Leftrightarrow \begin{cases} 6w_1 = 14 - 5w_2 \\ \text{———} \end{cases} \Leftrightarrow \begin{cases} w_1 = \frac{14 - 5w_2}{6} \\ 5\left(\frac{14 - 5w_2}{6}\right) + 6w_2 = 13.5 \end{cases}$$

$$\Leftrightarrow \begin{cases} \text{———} \\ \frac{70 - 25w_2}{6} + 6w_2 = 13.5 \end{cases} \Leftrightarrow \begin{cases} \text{———} \\ 70 - 25w_2 + 36w_2 = 81 \end{cases} \Leftrightarrow \begin{cases} \text{———} \\ 11w_2 = 11 \end{cases}$$

$$\Leftrightarrow \begin{cases} w_1 = \frac{14 - 5(1)}{6} \\ w_2 = 1 \end{cases} \Leftrightarrow \begin{cases} w_1 = \frac{9}{6} \\ w_2 = 1 \end{cases} \Leftrightarrow \begin{cases} w_1 = 1.5 \\ w_2 = 1 \end{cases}$$

Logo, o modelo é: $y = w_1 x_1 + w_2 x_2$

$$\Rightarrow y = 1.5\,(x_1) + (1)\,x_2$$

$$\boxed{y = 1.5\,x_1 + x_2}$$

b)

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = a_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + a_2 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + a_3 \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Resolvendo o Sistema de Equações:

$$\begin{cases} w_1 = a_1 + 2a_2 + a_3 \\ w_2 = a_1 + a_2 + 2a_3 \end{cases} \Leftrightarrow \begin{cases} 1.5 = a_1 + 2a_2 + a_3 \\ 1 = a_1 + a_2 + 2a_3 \end{cases} \Leftrightarrow \begin{cases} a_1 = 1.5 - 2a_2 - a_3 \\ \text{———} \end{cases}$$

$$\begin{cases} \text{———} \\ 1 = (1.5 - 2a_2 - \cancel{a_3}) + \cancel{a_2} + 2a_3 \end{cases} \Leftrightarrow \begin{cases} \text{———} \\ 1 = 1.5 - a_2 + a_3 \end{cases} \Leftrightarrow \begin{cases} a_1 = \frac{3}{2} - 2\left(\frac{1}{2} + a_3\right) - a_3 \\ a_2 = \frac{1}{2} + a_3 \end{cases}$$

$$\Leftrightarrow \begin{cases} a_1 = \frac{3}{2} - 1 - 2a_3 - a_3 \\ a_2 = \frac{1}{2} + a_3 \end{cases} \Leftrightarrow \begin{cases} a_1 = \frac{3}{2} - \frac{2}{2} - 3a_3 \\ \text{———} \end{cases} \Leftrightarrow \begin{cases} a_1 = +\frac{1}{2} - 3a_3 \\ a_2 = \frac{1}{2} + a_3 \end{cases}$$

Arbitrando a_3 ; $a_3 = 1$

$a_1 = +\frac{1}{2} - 3(1) = +\frac{1}{2} - \frac{6}{2} = -\frac{5}{2}$ //

$a_2 = \frac{1}{2} + 1 = \frac{1}{2} + \frac{2}{2} = \frac{3}{2}$ //

$$\boxed{\begin{array}{l} a_1 = -\frac{5}{2} \\ a_2 = \frac{3}{2} \\ a_3 = 1 \end{array}}$$

Testando

$w_1 = a_1 + 2a_2 + a_3 \Leftrightarrow 1.5 = -\frac{5}{2} + 2\left(\frac{3}{2}\right) + 1 \Leftrightarrow 1.5 = -\frac{5}{2} + \frac{6}{2} + \frac{2}{2}$

$\Leftrightarrow 1.5 = \frac{3}{2}$ (V)

$w_2 = a_1 + a_2 + 2a_3 \Leftrightarrow 1 = -\frac{5}{2} + \frac{3}{2} + 2(1) \Leftrightarrow 1 = -\frac{2}{2} + \frac{4}{2} \Leftrightarrow 1 = \frac{2}{2}$ (✓)

e) Matematicamente, $\text{span}(v_1, v_2, \dots v_n) = \{c_1v_1 + c_2v_2 + \dots c_nv_n\},$

$c_i \in K, \ i \in \{1 \dots n\}$

se desenvolvermos a equação

$$\min_w \underbrace{\sum_{i=1}^{n} L(w^T x_i, y_i) + \lambda \|w\|^2}_{} =$$

$$\min_w L(W^T X^T, y) + \lambda \|w\|^2$$

em que X tem de dimensão $n \times d$, X^T tem dim $d \times n$

y tem dimensão $n \times 1$

W tem dim $d \times 1$

o Span dos vectores é dado por:

$$\text{Span}(x_1, x_2, x_3, x_n) = \{a_1x_1 + a_2x_2 + a_3x_3 \dots a_3x_n\}$$

$a_i \in K, \ i \in \{1 \dots n\}$

Em cada iteração

o que pretende é encontrar

um w' que fique com $x_i w_i = y_i$

ou $a_i \bar{x}_i = y_i$ | Em isto que significa que consoante o número do ano n, o vetor w só tem n entradas [illegible] de $1 — n$

feito por um coeficiente $a_i x_i = a_i \begin{bmatrix} 1 \\ \vdots \\ d \end{bmatrix}$; ou constante que multiplica por cada as dimensões do vetor.

d. Sabendo o resultado do cin, se figuram sempre a_i o resultado mic do tipo:

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} = \sum_{i=1}^{n} a_i x_i$$

$$\Leftrightarrow \begin{bmatrix} w_1 \\ w_d \end{bmatrix} = a_1 \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} + \dots a_n \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

Sample 1 ——— Sample n

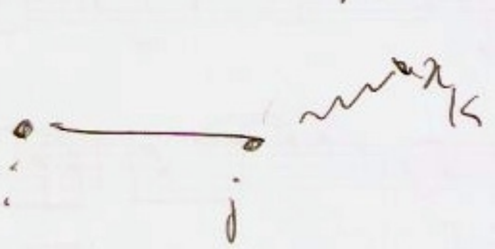
Com esta opção seria uma opção interessante quando:

- d é pequeno, isto é, quando os atributos existentes de cada amostra são em número relativamente pequeno, fazendo com que os vetores também não sejam demasiado extensos;
- n é pequeno, dado que se n for um número exageradamente grande, ter-se-ia de calcular n coeficientes e resolver um sistema com n coeficientes para cada peso.
- Nas duas opções acima, atuam claramente em situações em que os sistemas de equações são impossíveis e/ou indeterminados.

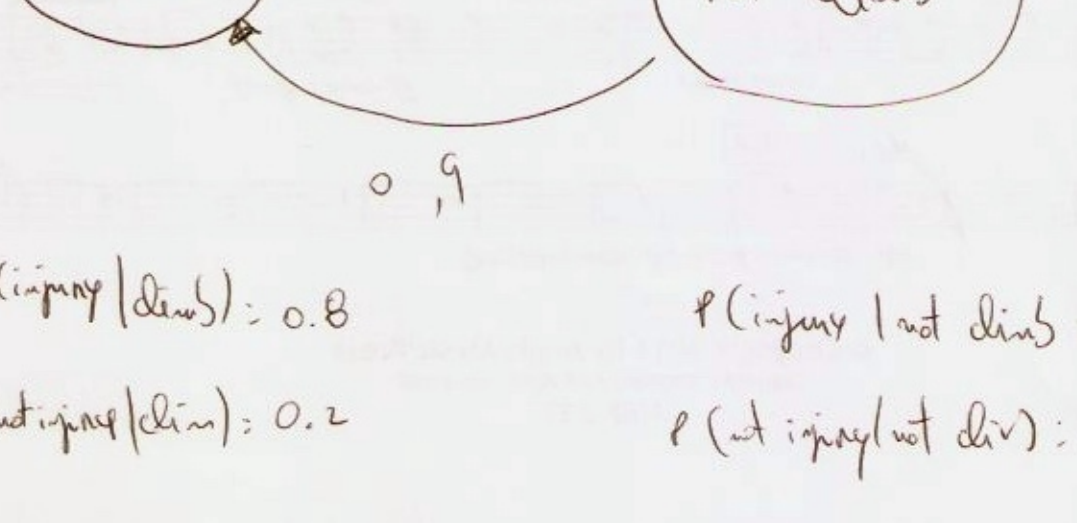
um estado interessante é quando n = 1, pois ficam parte no sistema de equação final e determinado. Se pudermos tornar, no caso que estes valores sejam devidamente conhecidos, que computacionalmente pode tornar - expressivo!

8.

a) Estamos perante dados sequenciais e cuja probabilidade de acontecer é dependente dos estados anteriores. Portanto, temos de assumir que cada sequência observada é produzida através do resultado de diversas transições entre estados, a depender de um estado:



b)



$P(\text{injury} | \text{climb}) : 0.8$

$P(\text{not injury} | \text{climb}) : 0.2$

$P(\text{injury} | \text{not climb} : 0.1$

$P(\text{not injury} | \text{not climb}) : 0.9$

The observation:

$$\text{Monday} \longrightarrow \text{Tuesday} \longrightarrow \text{Wednesday}$$

injury injury ~ injury

Injury Not injur

$$\underline{\overline{(\ 0\ ,\ 1\ )}}$$

Resolution i python:

No entanto:

Python code

Transition Matrix:

$$\begin{array}{c|cc} & \text{State} & \text{State}_2 \\ \text{state1} & 0.4 & 0.9 \\ \text{te}_2 & 0.9 & 0.4 \end{array}$$

Initial probs:

$$\begin{array}{cc} 0{,}98 & 0{,}02 \\ \text{State1} & \text{State}_2 \end{array}$$

Emission Probabilities:

$$\begin{array}{c|cc} & (li\text{-}) & \sim \text{dhn} \\ \text{injur} & 0{,}8 & 0{,}1 \\ \text{oti} & 0.2 & 0.9 \end{array}$$

$$\begin{array}{c|cc} & \text{injury} & \sim \text{Not injur} \\ \text{State1} & 0{,}8 & 0.2 \\ \text{State2} & 0{,}1 & 0.9 \end{array}$$

Then we compute o forward algorithm: Slide 53/60 lecture 11

The given probability is: 0,20

7.

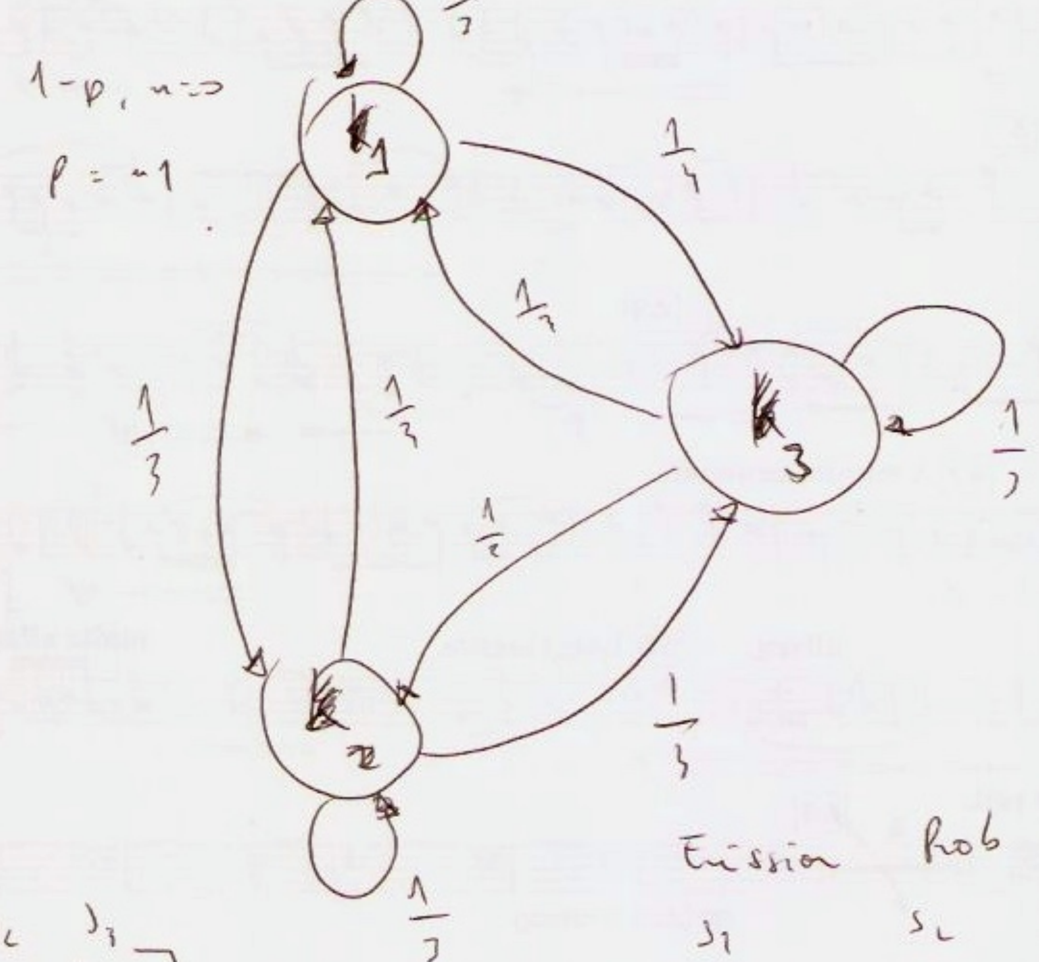
Benulli Distibution

$$D(n)\begin{cases} 1-p & \text{for } n=0 \\ 1 & \text{for } n=1 \end{cases}$$

$$p(n) = \begin{cases} 1-p, & n=0 \\ p, & n=1 \end{cases}$$

Initial Probs

$$\begin{array}{ccc} \left[\frac{1}{3}\right. & \frac{1}{3} & \left.\frac{1}{3}\right] \\ s_1 & s_2 & s_3 \end{array}$$



T Matrix

$$\begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ s_1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ s_2 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ s_3 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array}$$

Emission Prob Matrix

$$\begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ 0 & 1-p & 1-p & 1-p \\ 1 & p & p & p \end{array}$$