

**Lösningar**  
**Introduktion till maskininläring**  
**19 april 2021 8–14**

---

**Uppgift 1**

- a.) Låg bias, hög varians.
- b.) Låg bias, låg varians.
- c.) Låg bias, hög varians.
- d.) Låg bias, hög varians.
- e.) Låg bias, hög varians.
- f.) Lägre varians.

**Uppgift 2**

- a.) Låt  $\pi(x) = \mathbb{P}(y = 1|x) = \sigma(\alpha + \beta x)$ , då ges cross-entropyn ges av

$$-\log \mathbb{P}_\theta(\vec{y}) = - \left( \sum_{i=1}^n y_i \log \pi(x_i) + (1 - y_i) \log(1 - \pi(x_i)) \right)$$

- b.) Att minska bias. Om boosting ska ha (stor) positiv effekt bör modellerna ha hög bias, och vara snabba att träna.
- c.) Primära syftet med PCA är dimensionsreducering.
- d.) Policyn  $\pi$  är Markov om den *endast* tar hänsyn till vilket tillstånd systemet (MDPn) befinner sig i, och inte någon historik. Alltså,  $\pi : S \rightarrow A$ . Dessa policies är viktiga då

$$\mathbb{E}_\pi \left( \sum_{t=1}^{\infty} R_t \right)$$

maximeras av en *Markov* policy.

e.) Om tillståndsrummet  $S$  är för stort, eller om man inte har full kunskap om den underliggande MDPn (man kan ej beräkna  $p(s'|s, a)$ ,  $r(s, a)$  osv).

f.) Parametern  $w^T$  väljs så att det minsta avståndet mellan punkterna  $\{\phi(x_i), i = 1, \dots, n\}$  och hyperplanet  $f(x) = 0$  blir så stort som möjligt, givet att  $y_i f(x_i) > 0$ . Kan också skrivas formellt,

$$\begin{aligned} & \arg \min \|w\|^2 \\ \text{s.t.} \quad & y_i f(x_i) \geq 1. \end{aligned}$$

### Uppgift 3

a.)

```
Call:
lm(formula = y ~ x1 + x2 + x3, data = data)

Residuals:
    Min       1Q   Median       3Q      Max
-3.4674 -0.5742  0.0798  0.6786  1.6798

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.3521     0.2949   7.976 3.21e-10 ***
x1           0.8955     0.1332   6.721 2.39e-08 ***
x2          -0.5241     0.2908  -1.803  0.078 .
x3           0.3515     0.2560   1.373  0.176
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9804 on 46 degrees of freedom
Multiple R-squared:  0.5299,    Adjusted R-squared:  0.4992
F-statistic: 17.28 on 3 and 46 DF,  p-value: 1.177e-07
```

b.)

```
set.seed(1990)
train_data <- data %>%
sample_frac(0.75)

test_data <- data %>%
anti_join(train_data)

m1 <- lm(y ~ x1 + x2 + x3, data = train_data)

pred <- predict(m1, test_data)

mean(abs(pred - test_data$y)) %>% print()
```

Följande seed ger **mae = 0.8536**.

c.) Man kan, till exempel, undersöka detta genom att göra en korsvalidering.

```
folds <- seq(1,5)
data <- data %>%
mutate(fold = sample(folds,
nrow(data),
replace = TRUE))

for (x in c("x1", "x2", "x3", "x1+x2+x3")) {
mae <- 0
for (current_fold in folds) {
train_data <- data %>%
filter(fold != current_fold)

test_data <- data %>%
  filter(fold == current_fold)

f <- as.formula(paste0("y~", x))
m <- lm(f, data = train_data)
pred <- predict(m, test_data)
mae <- mae + mean(abs(pred - test_data$y))

}
print(f)
print(mae / length(folds) )
}
```

Detta ger

```
y ~ x1
[1] 0.764278
y ~ x2
[1] 1.117038
y ~ x3
[1] 1.138266
y ~ x1 + x2 + x3
[1] 0.7704743
```

Alltså, modellen  $y = \alpha + \beta_1 x_1$  verkar generalisera bäst till ny data.

## Uppgift 4

a.) Fördelar: neurala nätverk kan modellera icke-monotona beroenden mellan  $\mathbb{P}(y = 1|x)$  och  $x$ , samt att dessa beroenden kan modelleras *indirekt* modellen, givet tillräckligt med data. I en logistisk regression måste sådana beroenden modelleras explicit. Nackdelar: neurala nätverk kräver mer data; neurala nätverk är mycket svårare att tolka.

b.) Antal parametrar i modellen,

$$(10 \cdot 10 + 10) + (12 \cdot 10 + 12) + (14 \cdot 12 + 14) + (16 \cdot 14 + 16) + (18 \cdot 16 + 18) + (1 \cdot 18 + 1) = 989.$$

Givet  $h^{(5)}$  ges  $f_\theta(x)$  av

$$f_\theta(x) = f_\theta(h^{(5)}) = \sigma(W^{(6)}h^{(5)} + b)$$

c.) Early stopping är en metod som används för att undvika/mitigera overfitting, ofta i samband med backpropagation. Man vill minimera en lossfunktion  $L(\theta; D)$  på någon data  $D$ , utan att overfitta  $D$ . En variant av early stopping ser ut såhär. Dela upp data i träningsdata  $D_T$  och valideringsdata  $D_V$ . Idén är att minimera  $L(\theta)$  på träningsdata, men stanna så fort en ny (backprop-) iteration gör så att  $L(\theta; D_V)$  ökar.

1. Initiera  $\theta_0$

2. Slumpa  $(x_i, y_i)$  ifrån  $D_T$  och sätt  $\theta_{n+1} = \theta_n - \lambda \nabla L(\theta_n; (x_i, y_i))$  om  $L(\theta_{n+1}; D_V) < L(\theta_n; D_V)$

## Uppgift 5

a.)

$$p(x) = \pi_1 N(x|\mu_1, \Sigma_1) + (1 - \pi_2) N(x|\mu_2, \Sigma_2).$$

b.) Eftersom klusterna är sfäriska kommer båda algoritmerna att framgångsrikt separera båda klusterna och hitta samma klustercenter, alltså samma kluster indelning. Skillnaden är att GMM kommer göra en *soft clustering* och tilldela varje datapunkt en sannolikhet att tillhöra varje kluster.

c.)  $\mu_1$  kommer röra sig åt vänster och  $\mu_2$  åt höger.

d.) Likelihooden ökar alltid i en EM-iteration.

e.) Värdet på  $\pi_1$  kommer minska. Då

Låt  $z_n \in \{1, 2\}$  vara det kluster  $x_n$  tillhör.

$$\pi_1^{\text{old}} = 0.5$$

$$\mathbb{P}(z_n = 1|x_n) = \frac{\pi_1^{\text{old}} N(x_n|\mu_1, \Sigma_1)}{\pi_1^{\text{old}} N(x_n|\mu_1, \Sigma_1) + (1 - \pi_1^{\text{old}}) N(x_n|\mu_2, \Sigma_2)} = \frac{N(x_n|\mu_1, \Sigma_1)}{N(x_n|\mu_1, \Sigma_1) + N(x_n|\mu_2, \Sigma_2)}$$

$$N_1 = \sum_{k=1}^n \mathbb{P}(z_n = 1|x_n)$$

$$\pi_1^{\text{new}} = N_1/n.$$

Notera att  $N_1 = \sum_{k=1}^n \mathbb{P}(z_n = 1|x_n) < 0.5n$  då fler punkter ligger närmare  $\mu_2$  än  $\mu_1$ . Så  $\pi_1^{\text{new}} \leq \pi_1^{\text{old}}$ .