www.math-stockholm.se/cirkel

17 mars 2022

Välkomna!

Idag:

- Approximera en funktion från datapunkter
- Hur tränar man modellen?
- Två olika sorters fel

Figur: Artificiell intelligens

Att approximera en funktion från datapunkter (0)

- Approximera en linjär funktion f sådant att $f(x) = 2 + 3x$
- **Vi har:** punkter X_l och $f(X_l)$, $l = 1, \ldots, N$
- **Vi vill ha:** approximationen av f: $\alpha_\theta(x) = \theta_0 + \theta_1 x$
- Måste minimera $\hat{L}(X)$, medelvärdet av förlustfunktionen (medelkvadratfel) över N exempel, där

$$\hat{L}(X) = \frac{1}{N} \sum_{l=1}^{N} |\alpha_\theta(X_l) - f(X_l)|^2 = \frac{1}{N} \sum_{l=1}^{N} |\theta_0 + \theta_1(X_l) - f(X_l)|^2 \ (*)$$

Att approximera en funktion från datapunkter (0)

- Måste minimera $\hat{L}(X)$, medelvärdet av förlustfunktionen (medelkvadratfel) över N exempel, där

$$\hat{L}(X) = \frac{1}{N}\sum_{l=1}^{N} |\alpha_\theta(X_l) - f(X_l)|^2 = \frac{1}{N}\sum_{l=1}^{N} |\theta_0 + \theta_1(X_l) - f(X_l)|^2 \ (*)$$

- d.v.s. måste hitta θ_0, θ_1 (okända) som minimerar $(*)$
  - f är känd
  - N är känd
  - X_l är kända, $l = 1, \ldots, N$
- Vi ska använda **gradientnedstigning** och **stokastisk gradientnedstigning** för att hitta bästa θ_0, θ_1

Att approximera en funktion från datapunkter (0)

$$\hat{L}(X) = \frac{1}{N} \sum_{l=1}^{N} |\theta_0 + \theta_1(X_l) - f(X_l)|^2 \ (*)$$

- Måste beräkna $\nabla_\theta \hat{L}$
  - Måste hitta θ_0 och θ_1 som är optimala
  - Minimerar $\hat{L}(X)$
- Behöver:
  - Den partiella derivatan med avseende på θ_0 av $\hat{L}(X)$
  - Den partiella derivatan med avseende på θ_1 av $\hat{L}(X)$

Att approximera en funktion från datapunkter (0)

$$\hat{L}(X) = \frac{1}{N}\sum_{l=1}^{N} |\theta_0 + \theta_1(X_l) - f(X_l)|^2 \ (*)$$

- $\nabla_\theta \hat{L}(X) = \left[\frac{\partial \hat{L}}{\partial \theta_0}, \frac{\partial \hat{L}}{\partial \theta_1}\right]^T \in \mathbb{R}^2$ där

$$\frac{\partial \hat{L}}{\partial \theta_0} = \frac{2}{N}\sum_{l=1}^{N} (\theta_0 + \theta_1 X_l - f(X_l))$$

$$\frac{\partial \hat{L}}{\partial \theta_1} = \frac{2}{N}\sum_{l=1}^{N} ((\theta_0 + \theta_1 X_l - f(X_l)X_l))$$

- Gradienten $\nabla_\theta \hat{L}(X)$ är en **vektor**

Att approximera en funktion från datapunkter (0)

$$\frac{\partial \hat{L}}{\partial \theta_0} = \frac{2}{N} \sum_{l=1}^{N} (\theta_0 + \theta_1 X_l - f(X_l))$$

$$\frac{\partial \hat{L}}{\partial \theta_1} = \frac{2}{N} \sum_{l=1}^{N} ((\theta_0 + \theta_1 X_l - f(X_l) X_l))$$

```
def loss_gradient(x, f_of_x, theta):
    res = theta[0] + theta[1] * x - f_of_x
    return 2*(res.mean()), 2*((res * x).mean())
```

Att approximera en funktion från datapunkter (0)

```
def gradient_descent(
    gradient, x, f_of_x, start, gamma=0.1, m=50, tol=1
    e-06
):

    vector_nu = start
    for i in range(m):
        diff = -gamma * np.array(gradient(x, f_of_x,
    vector_nu))
        if np.all(np.abs(diff) <= tol):
            break
        vector_nu += diff
    return vector_nu
```

Notera: gradientnedstigning använder alla punkter i beräkningen av gradienten

Att approximera en funktion från datapunkter (0)

```
def sgd(
    gradient, x, f_of_x, start, gamma, batch_size, m,
    tol
):
```

- Stokastisk gradientnedstigning tar `batch_size` antal punkter i beräkningen av gradienten i varje steg
- Kompendiet: `batch_size = 1`
- Vi ska programmera i Python

Att approximera en funktion från datapunkter (0)

Sammanfattning

Vi har:

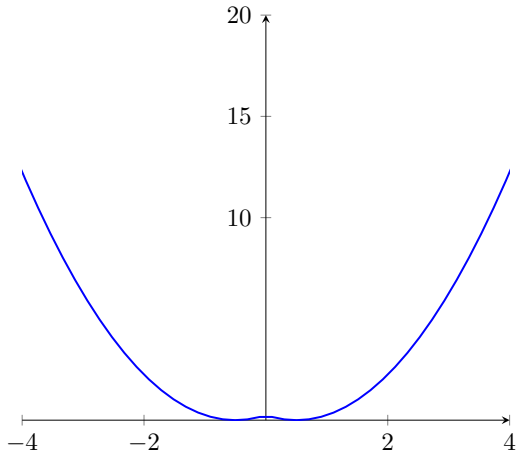
- Hittat θ_0 och θ_1 sådant att

$$\hat{L}(X) = \frac{1}{N}\sum_{l=1}^{N} |\theta_0 + \theta_1(X_l) - f(X_l)|^2$$

  minimerades
- Samma θ_0 och θ_1 sådant att $\theta_0 + \theta_1 x \approx 2 + 3x$
- Nu tar vi något lite mer komplicerat
- Ska diskutera maskininlärning

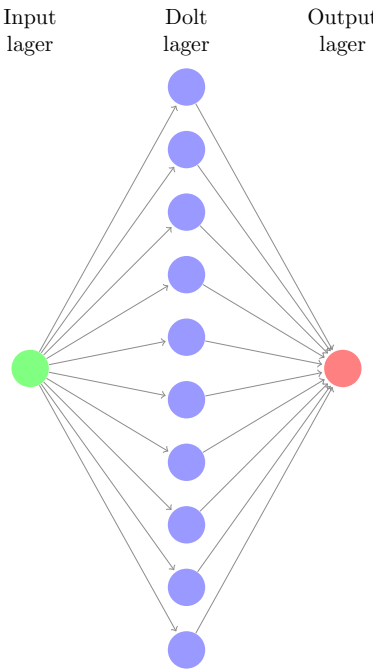
Att approximera en funktion från datapunkter (1)

- Approximera en kvadratisk funktion f sådant att $f(x) = \left|x - \frac{1}{2}\right|^2$
- Ska använda de verktyg vi har studerat för att approximera denna funktion
- Idag används maskininlärning för att lösa mycket mer komplicerade problem
- Dessa problem kräver mycket mer avancerade verktyg än vi har tid för



Att approximera en funktion från datapunkter (2)

- För att approximera f ska vi använda ett neuralt nätverk med ett dolt lager
- Vi har:
  - En ingångsnod
  - 10 noder i det dolda lagret
  - En utgångsnod



Att approximera en funktion från datapunkter (3)

- Approximerar f med en funktion $\alpha_{\boldsymbol{\theta}}$ som ges av

$$\begin{aligned} \alpha_{\boldsymbol{\theta}}(x) &= \sum_{k=1}^{K} \tilde{w}_{1,k}^{(0)} \sigma\left(w_{1,k}^{(0)} x + b_k^{(0)}\right) \\ \boldsymbol{\theta} &= [\tilde{w}_{1,k}^{(0)}, w_{1,k}^{(0)}, b_k^{(0)}], k = 1, \ldots, K \end{aligned}$$

  där $\boldsymbol{\theta} \in \mathbb{R}^{K \times 3}$

- x är ingångsnoden
- σ är aktiveringsfunktion
- $\boldsymbol{\theta}$ representerar vikter och bias
- $K = 10$ eftersom vi har 10 noder i det dolda lagret

Att approximera en funktion från datapunkter (4)

- Nya parametrar: $\tilde{w}_{1,k}^{(0)},\ k = 1, \ldots, K$
- Att implementera just den här modellen i Keras är faktiskt enklare
- Keras tar hand om de extra parametrarna, d.v.s. det finns inte något mer vi behöver göra för att hantera dem
- Målet: välja $\boldsymbol{\theta}$ som minimera värdet av förlustfunktionen (medelkvadratfel) över N, d.v.s., minimera

$$\frac{1}{N}\sum_{l=1}^{N}|\alpha_{\boldsymbol{\theta}}(X_l) - f(X_l)|^2,\ X = \{X_l\}_{l=1}^{N}$$

där $X_l, l = 1, \ldots, N$ ett stickprov från $\mathcal{U}(-4, 4)$ (inget utfall i $(-4, 4)$ är mer eller mindre sannolikt än något annat)

Att approximera en funktion från datapunkter (5)

- Vi formulerar vårt problem som

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{K\times 3}} \frac{1}{N}\sum_{l=1}^{N}|\alpha_{\boldsymbol{\theta}}(X_l)-f(X_l)|^2,\ X=\{X_l\}_{l=1}^{N}\ (*)$$

- Slumpmässigt generera N datapunkter på intervallet $(-4,4)$
- Dessa datapunkter kallas **träningsdata** eftersom vi kommer att träna modellen på dem
- Beräknar $f(X_l),\ l=1,\ldots,N$
- Motsvarar märkning av träningsdata (etiketter)
- Approximerar vi $\boldsymbol{\theta}$ som minimerar $(*)$ med en iterativ metod
- Detta är vad som kallas **träning**

Att approximera en funktion från datapunkter (6)

Algoritm 2: Approximera (7.5) med stokastisk gradientnedstigning

input : Startgissning $\boldsymbol{\theta}_0 \in \mathbb{R}^{3\times K}$
$X_1, \ldots, X_N$ stickprov från $\mathcal{U}(-4, 4)$
$\gamma \in \mathbb{R}$
Parameter $m \in \mathbb{Z}$ (antal steg)

output: $\boldsymbol{\theta}_{m+1}$ som approximerar (7.5)

1 Beräkna $f_l = f(X_l)$, $l = 1, \ldots, N$ där f är som i (7.1)
2 **for** $j = 0, 1, \ldots, m$ **do**
3 Välj slumpmässigt $i \in \{1, \ldots, N\}$
4 $\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - \gamma \nabla_{\boldsymbol{\theta}} \left(|\alpha_{\boldsymbol{\theta}_j}(X_i) - f(X_i)| \right)^2$
5 **end**

- Från kompendiet
- Notera: iterativ metod är stokastisk gradientnedstigning

Att evaluera ett neuralt nätverk (1)

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{K\times 3}} \frac{1}{N}\sum_{l=1}^{N}|\alpha_{\boldsymbol{\theta}}(X_l) - f(X_l)|^2,\ X = \{X_l\}_{l=1}^{N}\ (*)$$

- Två olika sorters fel
- **Träningsfelet** är vad vi minimerar med stokastisk gradientnedstigning
  - Tar in träningsdata $\{X_l, f(X_l)\}$, $l = 1, \ldots, N$ och hittar $\boldsymbol{\theta}_{m+1}$ för att approximera $(*)$
  - Träningsfelet ges av

$$\frac{1}{N}\sum_{l=1}^{N}|\alpha_{\boldsymbol{\theta}_{m+1}}(X_l) - f(X_l)|^2$$

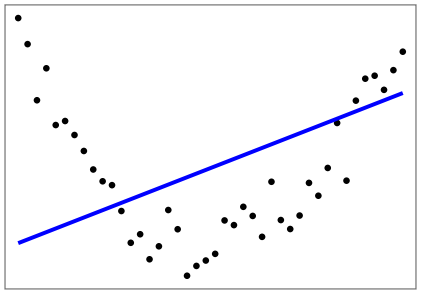
Att evaluera ett neuralt nätverk (2)

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{K\times 3}} \frac{1}{N}\sum_{l=1}^{N}|\alpha_{\boldsymbol{\theta}}(X_l) - f(X_l)|^2,\ X = \{X_l\}_{l=1}^{N}\ (*)$$

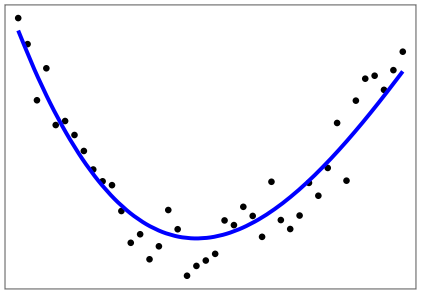
- Två olika sorters fel
- **Generaliseringsfelet** mäter hur stor förlustfunktionen är för data som modellen inte har tränat på
  - Ta ett stickprov av $\tilde{N}$ punkter $\tilde{X}_1, \ldots, \tilde{X}_{\tilde{N}}$ från $\mathcal{U}(-4, 4)$
  - Stickprovet heter **testmängden** eftersom vi bara använder det för att testa modellen
  - Beräknar generaliseringsfelet på testmängden som

$$\frac{1}{\tilde{N}}\sum_{l=1}^{\tilde{N}}|\alpha_{\boldsymbol{\theta}_{m+1}}(\tilde{X}_l) - f(\tilde{X}_l)|^2$$

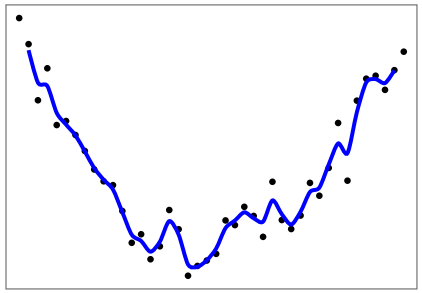
En datamängd, tre olika modeller



Underanpassad



Bra modell



Överanpassad

- **Underanpassning** innebär att en modell är för enkel och inte kan lära sig från träningsdata
  - Stort värde på träningsfelet
  - Ännu större värde på generaliseringsfelet
- **Överanpassning** innebär att en modell funkar bra på träningsdata men fungerar dåligt på osedd data (på testmängden)

Andra viktiga begrepp inom djupinlärning

- **Övervakad inlärning**: modellen tränas genom att behandla en datamängd med etiketter
- **Oövervakad inlärning**: modellens uppgift är att hitta mönster och avvikelser i en datamängd utan att ha fått instruktioner om vad som är rätt och vad som är fel
- **Backpropagation**: en algoritm som beräknar gradienten vid träning av det neurala nätverket
- **Framåtkopplade nätverk**: ett nätverk där informationen bara rör sig i en riktning (framåt) d.v.s. från ingångsnoderna, genom dolda noder (om sådana finns) och till utgångsnoderna

Nästa gången

- Jobba med att klassificera handskrivna siffror
- Jag är på en konferens i USA
- Ska spela in hur man gör (vecka 13)
- Frågor? Zoom rum 7 april, 16:00-17:30 (vecka 14)