Kunskapskontroll – Teoretiska Frågor

1.Hur är AI, Maskininlärning och Deep Learning relaterat?

Svar: Artificial Intelligence (AI), Machine Learning (ML) och Deep Learning (DL) är sammanlänkade fält inom datavetenskap, som var och en representerar olika abstraktionsnivåer och komplexitet i utvecklingen av intelligenta system. Medan AI är det breda konceptet för att skapa intelligenta maskiner, är ML ett specifikt tillvägagångssätt för att uppnå AI genom att göra det möjligt för maskiner att lära av data, och DL är ett mer specialiserat tillvägagångssätt inom ML som använder djupa neurala nätverk för att modellera komplexa mönster.

Artificial Intelligence (AI): Det bredaste fältet som omfattar all teknik som gör det möjligt för maskiner att efterlikna mänsklig intelligens.

Machine Learning (ML): En delmängd av AI som involverar träning av algoritmer för att lära av data och göra förutsägelser eller beslut utan explicit programmering.

Deep Learning (DL): En specialiserad delmängd av ML som använder flerskiktiga neurala nätverk för att analysera komplexa mönster i stora datamängder, särskilt användbart för uppgifter som involverar bilder, ljud och text.

2.Hur är Tensorflow och Keras relaterat?

Svar: TensorFlow är en öppen källkodsplattform för maskininlärning och ett symboliskt matematikbibliotek som används för olika maskininlärningsapplikationer. Keras, däremot, är ett neuralt nätverksbibliotek med öppen källkod som fungerar ovanpå TensorFlow. Det är designat för att vara användarvänligt och effektivt, vilket gör det lättare att utveckla algoritmer för djupinlärning. I huvudsak fungerar Keras som ett gränssnitt som utnyttjar TensorFlows kapacitet, vilket effektiviserar utvecklingen och träningen av modeller för djupinlärning.

Så här är TensorFlow och Keras relaterade:

TensorFlow är en robust plattform som tillhandahåller de grundläggande verktygen och operationerna på låg nivå som krävs för att skapa maskininlärningsmodeller från grunden. medan Keras fungerar som ett högnivå-API för TensorFlow, och erbjuder ett enklare och mer intuitivt gränssnitt för att bygga och träna modeller för djupinlärning. Keras förenklar många av

de komplexa uppgifter som är involverade i djupinlärning, vilket gör TensorFlow mer tillgängligt och snabbare att använda för vanliga djupinlärningsuppgifter.

3.Vad är en parameter? Vad är en hyperparameter?

Svar: Parametrar och hyperparametrar är integrerade i modellträningsprocessen i maskininlärning, men de fyller olika roller. Parametrar är variabler inom modellen som lärs från träningsdata. Under utbildningsprocessen uppdateras dessa parametrar av inlärningsalgoritmen för att minimera fel och förbättra modellens prestanda. Till skillnad från hyperparametrar har parametrar inte förinställda optimala värden; de initieras, ibland med relevanta startvärden, och justeras under hela utbildningsprocessen. När träningen är klar har parametrarna optimala värden som gör att modellen kan göra korrekta förutsägelser baserat på nya data. Däremot är hyperparametrar externa i förhållande till modellen och styr själva inlärningsprocessen. De påverkar beteendet och prestanda för inlärningsalgoritmen, vilket påverkar hur modellparametrarna lärs in. Prefixet "hyper_" indikerar att dessa är parametrar på toppnivå som styr den övergripande träningsprocessen. Exempel på hyperparametrar inkluderar inlärningshastigheten, batchstorleken och antalet epoker, som måste ställas in innan träningen börjar och ofta kräver inställning för optimal prestanda. Sammanfattningsvis lär man sig parametrar från data under träning, medan hyperparametrar är fördefinierade och styr inlärningsprocessen.

4.När man skall göra modellval och modellutvärdering så kan man använda ett tränings, validerings och test data. Förklara hur de olika delarna kan användas.

Svar : När man utvecklar och utvärderar maskininlärningsmodeller delas datasetet vanligtvis in i tre delar: utbildning, validering och testuppsättningar. Varje del tjänar ett distinkt syfte i modellvalet och utvärderingsprocessen. Genom att dela upp data i dessa tre delar säkerställer du att modellen är korrekt tränad, hyperparametrar är optimalt inställda och den slutliga utvärderingen återspeglar modellens verkliga prestanda på ny data.

Träningsdata

Syftet med träningsdata är att träna modellen. Modellen lär sig av dessa data genom att justera sina interna parametrar.

Valideringsdata

Syftet med datavalidering är att justera hyperparametrar och förhindra överanpassning. Den utvärderar och justerar modellen och säkerställer att den generaliserar väl.

Testdata

Syftet med testdata är att utvärdera den slutliga modellens prestanda. Och den används för att bedöma hur väl modellen presterar på osynliga data, vilket ger en opartisk utvärdering.

5. Förklara vad nedanstående kod gör

```
n_cols = X_train.shape[1]

nn_model = Sequential()
nn_model.add(Dense(100, activation = 'relu', input_shape = (n_cols, )))
nn_model.add(Dropout(rate=0.2))
nn_model.add(Dense(50, activation = 'relu'))
nn_model.add(Dense(1, activation = 'sigmoid'))

nn_model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

early_stopping_monitor = EarlyStopping(patience = 5)
nn_model.fit(X_train, y_train, validation_split = 0.2, epochs = 100, callbacks = [early_stopping_monitor])
```

Förklara vad nedanstående kod gör:

Svar: This code snippet defines and trains a neural network model using the Keras library. Here's a theoretical overview of what each part of the code does:

Determine the Number of Input Features:

The line n_cols = X_train.shape[1] calculates the number of input features from the training data X_train. This value is used to define the input shape for the neural network.

Initialize the Model:

nn_model = Sequential() initializes a Sequential model, which allows stacking layers sequentially.

Add Layers to the Model:

nn_model.add(Dense(100, activation='relu', input_shape=(n_cols,))) adds an input layer with 100 neurons and a ReLU activation function. The input_shape parameter specifies the number of input features.

nn_model.add(Dropout(rate=0.2)) adds a Dropout layer with a dropout rate of 20%, which helps prevent overfitting by randomly setting 20% of the input units to zero during each training step.

nn_model.add(Dense(50, activation='relu')) adds a hidden layer with 50 neurons and a ReLU activation function.

nn_model.add(Dense(1, activation='sigmoid')) adds an output layer with 1 neuron and a sigmoid activation function, suitable for binary classification tasks.

Compile the Model:

nn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']) compiles the model, specifying the Adam optimizer for training, binary cross-entropy as the loss function, and accuracy as the performance metric to be monitored during training.

Set Up Early Stopping:

early_stopping_monitor = EarlyStopping(patience=5) sets up an early stopping mechanism. Early stopping monitors the validation loss and stops the training process if the validation loss does not improve for 5 consecutive epochs, helping to prevent overfitting and save computational resources.

Train the Model:

nn_model.fit(X_train, y_train, validation_split=0.2, epochs=100, callbacks=[early_stopping_monitor]) trains the model using the training data (X_train and y_train). It splits 20% of the training data for validation, runs the training for up to 100 epochs, and uses the early stopping mechanism to halt training if the model's performance on the validation data stops improving.

In summary, this code builds and trains a neural network for a binary classification problem, incorporating techniques such as dropout for regularization and early stopping to prevent overfitting.

6. Vad är syftet med att regularisera en modell?

Svar: Att reglera en modell syftar till att förhindra överanpassning och förbättra dess generalisering till nya, osynliga data. Regulariseringstekniker lägger till begränsningar eller påföljder för inlärningsprocessen, vilket hjälper till att skapa en modell som fungerar bra inte bara på träningsdata utan även på validerings- och testdatauppsättningar. Här är anledningen till att reglering är viktigt. Sammanfattningsvis är regularisering avgörande för att utveckla maskininlärningsmodeller som inte bara presterar bra på träningsuppsättningen utan också generaliserar effektivt till ny, osynlig data.

Regularisering hjälper till att förhindra att modellen passar in buller och extremvärden i träningsdatan, vilket kan leda till dålig prestanda på ny data.

Förbättra generalisering: Det säkerställer att modellen presterar bra på osynliga data genom att uppmuntra enklare modeller som fångar de verkliga underliggande mönstren.

Vanliga tekniker:

L1 Regularization (Lasso): Lägger till ett straff för stora koefficienter, uppmuntrar sparsitet och funktionsval.

L2 Regularization (Ridge): Lägger till ett straff för den kvadratiska storleken av koefficienter, stabiliserar modellen och förhindrar stora svängningar i parametervärden.

Bortfall: Slumpmässigt tappar neuroner under träning för att förhindra att modellen blir alltför beroende av specifika neuroner, vilket främjar robusthet.

Tidig stopp: Stoppar träningen när prestandan på valideringsdata börjar försämras, vilket förhindrar att modellen överpassar träningsdatan.

Dataökning: Skapar mer träningsdata genom att modifiera befintliga data, hjälpa modellen att lära sig mer allmänna funktioner och förbättra dess förmåga att generalisera.

7. ”Dropout” är en regulariseringsteknik, vad är det för något?

Svar: Dropout är en regleringsteknik som används i neurala nätverk för att förhindra överanpassning. Det fungerar genom att slumpmässigt "ta bort" en delmängd av neuroner under träningsprocessen, vilket innebär att tillfälligt ta bort dem tillsammans med deras anslutningar. Detta tvingar nätverket att inte förlita sig på några specifika neuroner och uppmuntrar det att lära sig mer robusta funktioner som generaliserar bra till osynliga data. Så här fungerar det

Slumpmässigt urval: Under varje träningsiteration väljs en bråkdel (vanligtvis mellan 20 % och 50 %) av neuroner i nätverket slumpmässigt ut för att ignoreras (bortfaller).

Tillfälligt avlägsnande: De valda neuronerna tas tillfälligt bort från nätverket, och deras utsignaler beaktas inte för just den fram- och bakåtpassningen.

Skalade utgångar: Under träning skalas de återstående neuronerna upp med en faktor för att bibehålla den totala uteffekten. Vid testtillfället är bortfallet avstängt, men vikterna skalas ner med bortfallet för att ta hänsyn till de saknade enheterna under träningen.

Fördelar med Dropout

Minskar överanpassning: Genom att förhindra neuroner från att anpassa sig för mycket till träningsdata, hjälper bortfallet att minska överanpassningen.

Främjar robusta funktioner: Det tvingar nätverket att lära sig redundanta representationer, vilket gör nätverket mer robust och kapabelt att generalisera bättre till ny data.

Dropout är en enkel men effektiv teknik för att förbättra prestandan hos neurala nätverk genom att göra dem mer robusta och förhindra överanpassning.

8. ”Early stopping” är en regulariseringsteknik, vad är det för något?

Svar: Early stopping" är en regleringsteknik som används för att träna maskininlärningsmodeller, särskilt neurala nätverk, för att förhindra överanpassning. Tanken bakom tidig stopp är att övervaka modellens prestanda på en separat valideringsdatauppsättning under träningsprocessen och stoppa träningen när prestandan på valideringsdataset börjar försämras, vilket indikerar att ytterligare träning kan leda till överanpassning.. Tidig stopp är en regulariseringsteknik som används för att förhindra överanpassning genom att övervaka modellens prestanda på en valideringsdatauppsättning under träning och stoppa träningsprocessen när prestandan inte längre förbättras. Det hjälper till att säkerställa att modellen generaliserar väl till nya, osynliga data.

9.Din kollega frågar dig vilken typ av neuralt nätverk som är populärt för bildanalys, vad svarar du?

Svar: Convolutional Neural Networks (CNN) är populära för bildanalys. De är speciellt utformade för att effektivt hantera den rumsliga strukturen av bilder genom att använda faltningslager, poolande lager och andra specialiserade tekniker. CNN har uppnått anmärkningsvärd framgång i uppgifter som bildklassificering, objektdetektering, bildsegmentering och ansiktsigenkänning. Deras förmåga att automatiskt lära sig hierarkiska representationer av funktioner från råa pixelvärden gör dem väl lämpade för ett brett utbud av bildanalysuppgifter.

10. Förklara översiktligt hur ett ”Convolutional Neural Network” fungerar.

Svar: A Convolutional Neural Network (CNN) är en typ av neuralt nätverk som är särskilt effektivt för att bearbeta och analysera visuella data, såsom bilder. Här är en kort förklaring av hur CNN fungerar:

Konvolutionella lager: CNN använder faltningslager för att extrahera funktioner från inmatningsbilden. Varje lager består av en uppsättning inlärbara filter (även kallade kärnor) som glider över inmatningsbilden och utför faltningsoperationer.

Konvolution involverar elementvis multiplikation av filtret med en liten del av ingångsbilden, följt av summering av resultaten för att producera en enda utdatapixel.

Genom att använda flera filter i varje faltningslager kan CNN fånga olika funktioner på olika rumsliga platser i inmatningsbilden.

Aktiveringsfunktion: Efter faltning appliceras en aktiveringsfunktion (vanligen ReLU - Rectified Linear Unit) elementvis för att introducera icke-linjäritet i modellen, vilket gör att den kan lära sig komplexa mönster.

Poolande lager: Poolningslager är varvade mellan faltningslager för att reducera de rumsliga dimensionerna (bredd och höjd) på funktionskartorna samtidigt som viktig information behålls.

Vanliga poolningsoperationer inkluderar maximal pooling och genomsnittlig pooling, som nedsamplar funktionskartorna genom att ta det maximala eller genomsnittliga värdet inom varje poolningsfönster.

Tillplattning: Efter flera faltnings- och sammanslagningslager plattas de resulterande särdragskartorna till en endimensionell vektor. Denna utjämningsprocess förbereder data för inmatning i de fullt anslutna lagren.

Fullt anslutna lager:De tillplattade egenskapsvektorerna matas in i ett eller flera helt sammankopplade lager, som är traditionella neurala nätverkslager där varje neuron är kopplad till varje neuron i det föregående lagret.

Dessa helt anslutna lager utför resonemang och beslutsfattande på hög nivå baserat på de extraherade funktionerna.

Output Layer: Det sista lagret av CNN är vanligtvis ett softmax-aktiveringslager för klassificeringsuppgifter, vilket ger en sannolikhetsfördelning över de olika klasserna. För uppgifter som objektdetektering eller segmentering kan utdatalagret ha en annan arkitektur för att passa de specifika kraven för uppgiften.

Träning:

CNN: er tränas med gradientbaserade optimeringsalgoritmer som stokastisk gradient descent (SGD) eller dess varianter.

Under träning uppdateras modellens parametrar (filtrets vikter, fördomar, etc.) iterativt för att minimera en förlustfunktion, vanligtvis skillnaden mellan de förutsagda utsignalerna och de sanna etiketterna.

11.Din vän har ett album med 100 olika bilder som innehåller t.ex. tennisbollar och zebror. Hur hade han/hon kunnat klassificera de bilderna trots att han/hon inte har någon mer data att träna en modell på?

Svar: Min vän kan använda en teknik som kallas transfer learning för att klassificera bilder även utan att ha ytterligare data för att träna en modell från grunden. Överföringsinlärning utnyttjar kunskapen som en förtränad modell lärt sig på en stor datamängd (t.ex. ImageNet) och tillämpar den på en ny, mindre datamängd med liknande egenskaper.

Här är några steg genom vilka min vän kan använda transfer learning:

Förutbildat modellval:

Välj en förtränad modell för djupinlärning som har tränats på en stor datamängd, som VGG, ResNet eller Inception. Dessa modeller har lärt sig att extrahera meningsfulla egenskaper från bilder.

Särdragsextraktion:

Ta bort utgångsskiktet från den förtränade modellen och lämna modellens faltningsskikt intakta.

Använd den förtränade modellen för att extrahera funktioner från bilderna i albumet. Denna process innebär att varje bild passerar genom den förtränade modellen och registrerar aktiveringarna av det sista faltningsskiktet.

Finjustering eller tillägg av klassificerare:

Beroende på storleken och likheten hos den nya datamängden kan din vän välja att antingen:

Finjustera: Finjustera parametrarna för den förtränade modellen på den nya datamängden genom att lägga till ett nytt klassificeringsskikt och träna modellen med en liten inlärningshastighet. Detta gör att modellen kan anpassa sina inlärda funktioner till den nya datamängden.

Lägg till en klassificerare: Träna en ny klassificerare (t.ex. ett helt anslutet lager) ovanpå den förtränade modellens funktioner. Denna klassificerare tränas med hjälp av de extraherade funktionerna som indata och motsvarande etiketter från den nya datamängden.

Utvärdering:

När modellen har tränats eller finjusterats, utvärdera dess prestanda på ett separat valideringseller testset för att bedöma dess noggrannhet och generaliseringsförmåga.

Genom att använda överföringsinlärning kan din vän dra nytta av kunskapen som den förtränade modellen lärt sig på en stor datamängd och använda den för att klassificera bilder i sitt album, även utan att ha ytterligare data för att träna en modell från grunden. Detta tillvägagångssätt kan ge effektiva resultat, särskilt när man hanterar begränsad data.

12.Vad gör nedanstående kod?

```
model.save('model_file.h5')
```

```
my_model = load_model('model_file.h5')
```

Svar: model.save('model_file.h5'): Detta kommando sparar hela modellen till en fil som heter model_file.h5. Den sparade modellen inkluderar modellarkitekturen, vikter och träningskonfiguration (som förlustfunktionen och optimeraren). Detta gör att modellen kan laddas om senare och användas exakt som den var vid tidpunkten för sparandet.

my_model = load_model('model_file.h5'): Detta kommando laddar modellen från model_file.h5-filen tillbaka till ett Keras-modellobjekt. Den laddade modellen (my_model) är

identisk med den ursprungliga modellen, inklusive dess arkitektur, vikter och träningskonfiguration. Detta gör det möjligt att återuppta träningen eller använda modellen för att göra förutsägelser utan att behöva träna om den från början.

13.Deep Learning modeller kan ta lång tid att träna, då kan GPU via t.ex. Google Colab skynda på träningen avsevärt. Läs följande artikel: https://blog.purestorage.com/purely-informational/cpu-vs-gpu-for-machine-learning/ och skriv mycket kortfattat vad CPU och GPU är.

Svar: En CPU (Central Processing Unit) är den primära komponenten i en dator som utför det mesta av bearbetningen inuti datorn. Den är designad för sekventiell uppgiftsbearbetning och utmärker sig för att hantera en mängd olika allmänna uppgifter.

En GPU (Graphics Processing Unit), ursprungligen designad för att rendera bilder, utmärker sig nu vid parallella bearbetningsuppgifter, vilket gör den mycket effektiv för specialiserade beräkningar som de som behövs för maskininlärning och djupinlärning. GPU:er kan bearbeta flera uppgifter samtidigt, vilket avsevärt snabbar upp träningstiderna för modeller för djupinlärning.