ANNAMACHARYA INSTITUTE OF TECHNOLOGY AND SCIENCES RAJAMPET

(Autonomous)

Department of ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Lecture Notes

Name of the Faculty: D.SAI SHIREESHA

Class: III B.Tech II Sem.

Branch : AIDS

Name of the Course: DEEP LEARNING

Subject Code: 23A3062T

Academic Year: 2025-2026

UNIT-1

Linear Algebra: Scalars, Vectors, Matrices and Tensors, Matrix operations, types of matrices, Norms, Eigen decomposition, Singular Value Decomposition, Principal Components Analysis. Probability and Information Theory: Random Variables, Probability Distributions, Marginal Probability, Conditional Probability, Expectation, Variance and Covariance, Bays' Rule, Information Theory. Numerical Computation:OverflowandUnderflow,Gradient-BasedOptimization,ConstrainedOptimization,Linear Least Squares.

Introduction

Linear Algebra forms the **mathematical foundation of Deep Learning and Machine Learning**. Almost everyoperation in a neural network — such as input representation, weight computation, forward propagation, and gradient calculation — is expressed using **scalars, vectors, matrices, and tensors**. Among these, **scalars and vectors** are the most fundamental buildingblocks.Aclearunderstandingoftheseconceptsisessentialtoanalyzeanddesigndeep learning models effectively.

Scalars

Definition

A**scalar**isasinglenumericalvaluethatrepresents **magnitudeonly**andhas**nodirection**. Scalars belong to the set of real numbers or integers.

a∈Ra\in\mathbb{R}a∈R

CharacteristicsofScalars

- One-dimensional
- Nodirection
- Representconstantsor parameters
- Canbepositive,negative, orzero

ExamplesofScalars

- Temperature=30°C
- Learning rate inneuralnetworks= 0.01
- Biasterminaneuron
- Lossvalue producedbya neuralnetwork

RoleofScalarsinDeep Learning

InDeepLearning,scalarsarewidelyusedin:

- Learningrateduring optimization
- Regularizationparameters(λ)
- Biasvaluesaddedtoneurons
- OutputlossvaluessuchasMeanSquared Error

Thus,scalarscontrolthe**behaviorand performance**ofneuralnetworks.

Vectors

Definition

A**vector**isanorderedcollectionofnumbersarrangedeitherasa **row**ora**column**, representing both **magnitude and direction**.

x=[x1x2⋮xn]∈Rn\mathbf{x}=\begin{bmatrix}x_1 \\x_2\\\vdots\\x_n\end{bmatrix} \in \mathbb{R}^nx=x1x2⋮xn∈Rn

TypesofVectors

1. **RowVector**

x=[x1x2x3]\mathbf{x}=[x_1 \;x_2\;x_3]x=[x1x2x3]

2. **ColumnVector**

x=[x1x2x3]\mathbf{x}=\begin{bmatrix}x_1 \\x_2\\x_3\end{bmatrix}x=x1x2x3

VectorOperations

1. VectorAddition

Two vectors can be added only if they have the same dimension.

a+b=[a1+b1a2+b2]\mathbf{a}+\mathbf{b}=\begin{bmatrix}a_1+b_1\\a_2+b_2 \end{bmatrix}a+b=[a1+b1a2+b2]

Example:

[12]+[34]=[46]\begin{bmatrix}1\\2\end{bmatrix}+\begin{bmatrix}3\\4\end{bmatrix}= \begin{bmatrix}4\\6\end{bmatrix}[12]+[34]=[46]

2. ScalarMultiplication

Avectorcanbemultipliedbyascalar.

cx=[cx1cx2]c\mathbf{x}=\begin{bmatrix}cx_1 \\cx_2\end{bmatrix}cx=[cx1cx2]

3. DotProduct(InnerProduct)

Thedotproductoftwovectorsisdefinedas:

a·b=∑i=1naibi\mathbf{a}\cdot\mathbf{b}=\sum_{i=1}^{n}a_ib_ia·b=i=1∑naibi

Example:

[12]·[34]=1(3)+2(4)=11\begin{bmatrix}1\\2\end{bmatrix}\cdot\begin{bmatrix}3\\
4\end{bmatrix} = 1(3) + 2(4) = 11[12]·[34]=1(3)+2(4)=11

GeometricalInterpretationofVectors

Vectorscanberepresentedgeometricallyas arrows:

- Length→magnitude
- Direction→ orientation

Thedotproductmeasures**similarity**betweenvectorsandisusedextensivelyin:

- Cosinesimilarity
- Attentionmechanisms
- Classificationtask

ImportanceofVectorsinDeepLearning

Vectorsareusedtorepresent:

- Inputfeatures
- Wordembeddings
- Imagepixels(flattened)
- Activationsinneuralnetworks
- Gradientsduring backpropagation

Everylayerina neuralnetworkprocessesvectorstolearnpatternsfromdata.

PythonExample(VectorOperations)

```
importnumpyasnp

x=np.array([1,2,3])
```

```
y=np.array([4,5,6])

print("Addition:", x + y)
print("DotProduct:",np.dot(x,y))
```

CommonMistakesbyStudents

- Mixingrowandcolumn vectors
- Ignoringvectordimensions
- Applyingdotproductonincompatiblevectors

Scalarsand vectorsarethe **basicmathematicalentities**usedinDeepLearning. Scalarscontrol learningbehavior,while vectorsrepresent dataandparameters.Astrongunderstandingofthese concepts enables students to understand more advanced topics such as matrices, tensors, eigen decomposition, and optimization.

After understanding scalars and vectors, the next fundamental concepts in Linear Algebra are **matrices and tensors**. These structures allow us to represent **multiple vectors together** and handle **high-dimensional data** efficiently. In Deep Learning, almost every operation — from storingdatasetstorepresentingneuralnetworkweights —reliesheavilyonmatricesandtensors.

2. Matrices

Definition

A**matrix**isatwo-dimensionalrectangulararrayofnumbersarranged in **rowsandcolumns**.

A=[a11a12…a1na21a22…a2n⋮⋮⋱⋮am1am2…amn]A= \begin{bmatrix} a_{11} &a_{12} & \dots&a_{1n}\\a_{21}&a_{22}&\dots &a_{2n}\\\vdots&\vdots &\ddots &\vdots \\ a_{m1}&a_{m2}&\dots&a_{mn}\end{bmatrix}A=a11a21⋮am1a12a22⋮am2……⋱…a1na2n ⋮amn

Where:

- mmm=numberofrows
- nnn= numberofcolumns

Matrix Dimensions

Represented as **m×n**

- Example:
  - 3×23× 23×2→3rows,2columns

ExamplesofMatrices

- Studentmarkstable
- Imagepixels(2Dgrid)
- Weightmatrixinneuralnetworks

3. TypesofMatrices(Overview)

- RowMatrix
- ColumnMatrix
- SquareMatrix
- ZeroMatrix
- DiagonalMatrix
- IdentityMatrix
- SymmetricMatrix

4. MatrixOperations(Basic)

Matrix Addition

Twomatricescanbeadded

onlyiftheyhavethesamedimensions. A+B=[aij+bij]A + B =

[a_{ij} + b_{ij}]A+B=[aij+bij] **Example**:

[1234]+[5678]=[681012]\begin{bmatrix}1 &2\\3&4 \end{bmatrix}+\begin{bmatrix}5 &6 \\7 &8\end{bmatrix} = \begin{bmatrix} 6&8\\10 &12\end{bmatrix}[1324]+[5768]=[610 812]

ScalarMultiplication

Eachelementofthe matrixismultiplied byascalar.

cA=[c·aij]cA = [c \cdot a_{ij}]cA=[c·aij]

MatrixMultiplication

If:

- AAAism×nm×nm×n
- BBBisn×pn×pn×p

Then:

AB=C(m×p)AB=C\quad(m× p)AB=C(m×p)

Matrixmultiplicationis**notcommutative**.

5. ImportanceofMatricesinDeepLearning

Matricesareusedto represent:

- Inputdatabatches
- Weight parameters
- Biasvalues
- Activationoutputs

Example:

Fora neurallayer:

y=Wx+b\mathbf{y}=W\mathbf{x}+by=Wx+b

Where:

- WWW→ weightmatrix
- x\mathbf{x}x→inputvector
- bbb→biasvector

Thisshowsthat**matrixmultiplication isthecoreoperation inneuralnetworks**.

6. Tensors

Definition

A**tensor** isageneralizationofscalars,vectors,andmatricesto**higherdimensions**.

Object	**Dimension**
Scalar	0-D
Vector	1-D
Matrix	2-D
Tensor	≥3-D

ExamplesofTensors

1. **3-D Tensor**
   - Colorimage:(Height×Width×Channels)
2. **4-D Tensor**

- Batchofimages:
  (Batchsize×Height×Width×Channels)

TensorRepresentation

Tijk(3-Dtensor)T_{ijk} \quad\text{(3-Dtensor)}Tijk(3-D tensor)

7. TensorinDeepLearning

Tensorsarethe**primarydatastructure**indeeplearningframeworkssuchas:

- TensorFlow
- PyTorch

Used for:

- Inputdata
- Featuremaps inCNNs
- Gradients
- Modelparameters

8. PythonExample(Matrix&Tensor)

```
importnumpyasnp


#Matrix
A=np.array([[1,2],[3, 4]])

#3DTensor
T=np.array([

    [[1,2],[3,4]],

    [[5,6],[7,8]]

])


print("MatrixA:\n",A)
print("TensorT:\n",T)
```

9. CommonMistakesbyStudents

- Confusingmatrixandtensordimensions
- Invalid matrix multiplication
- Ignoringshapecompatibility

10. ApplicationsinDeepLearning

- CNNsuse4-D tensors

- RNNsprocess 3-Dtensors
- Backpropagationusestensoroperations

Matrices and tensors enable efficient representation and computation of high-dimensionaldata. Theyformthe**structuralbackboneofDeepLearningmodels**, makingthemindispensable for modern AI systems.

Matrices are the **core computationalstructures** inLinear Algebra and Deep Learning. Once matrices are defined, various **matrix operations** are performed on them to manipulate data, transformfeatures, andtrainneuralnetworks. Additionally, **specialtypesofmatrices**possess unique properties that simplify computations and improve efficiency. Understanding matrix operationsandtypesisessentialforbuilding,analyzing,andoptimizingdeeplearningmodels.

2. MatrixOperations

Matrixoperationsallowustocombine, transform,andanalyzematrices mathematically.

Matrix Addition

Definition

Twomatricescanbeadded

onlyiftheyhavethesamedimensions. A+B=[aij+bij]A + B =

[a_{ij} + b_{ij}]A+B=[aij+bij] **Example**

A=[1234],B=[5678]A=\begin{bmatrix} 1 &2\\3 &4 \end{bmatrix},\quad B= \begin{bmatrix}5 &6\\ 7 &8 \end{bmatrix}A=[1324],B=[5768]A+B=[681012]A+B= \begin{bmatrix} 6 &8\\ 10&12\end{bmatrix}A+B=[610812]

Properties

- Commutative:A+B=B+AA+B=B+AA+B=B+A
- Associative
- Identityelement: Zeromatrix

ScalarMultiplication

Definition

Eachelementofthematrixismultipliedbyascalarvalue. cA=[c·aij]cA = [c

\cdot a_{ij}]cA=[c·aij]

Example

2[1234]=[2468]2\begin{bmatrix}1 &2\\ 3&4\end{bmatrix} =\begin{bmatrix} 2 &4\\ 6 & 8\end{bmatrix}2[1324]=[2648]

MatrixMultiplication **Definition**

Matrixmultiplicationispossiblewhen:

- AAAisofsizem×nm\timesnm×n
- BBBisofsizen×pn\timespn×p

Resulting matrix:

AB=C(m×p)AB=C\quad(m\timesp)AB=C(m×p)cij=∑k=1naikbkjc_{ij}= \sum_{k=1}^{n} a_{ik}b_{kj}cij=k=1∑naikbkj

Example

A=[1234],B=[5678]A=\begin{bmatrix} 1 &2\\3 &4 \end{bmatrix},\quad B= \begin{bmatrix}5&6\\7&8\end{bmatrix}A=[1324],B=[5768]AB=[19224350]AB= \begin{bmatrix} 19 &22\\ 43&50\end{bmatrix}AB=[19432250]

ImportantProperties

- **Notcommutative**:AB≠BAAB\neqBAAB 偃=BA
- Associative
- Distributiveover addition

TransposeofaMatrix

Definition

ThetransposeofmatrixAAAisobtainedbyinterchangingrowsandcolumns. AT=[aji]A^T =

[a_{ji}]AT=[aji]

Example

A=[1234]⇒AT=[1324]A= \begin{bmatrix}1 &2\\3&4 \end{bmatrix} \RightarrowA^T= \begin{bmatrix}1 &3\\ 2&4\end{bmatrix}A=[1324]⇒AT=[1234]

InverseofaMatrix

Definition

The inverseofasquare matrixAAAisdenotedasA−1A^{-1}A−1suchthat:

AA−1=IAA^{-1} = IAA−1=I

Conditions

- Matrixmustbesquare
- Determinantmustbenon-zero

3. TypesofMatrices

RowMatrix

- Single row
- Dimension:1×n1\timesn1×n

ColumnMatrix

- Singlecolumn
- Dimension:m×1m\times1m×1

SquareMatrix

- Samenumberofrowsand columns
- Requiredfordeterminantand inverse

Zero(Null)Matrix

- Allelements arezero
- Additive identity

DiagonalMatrix

- Non-zeroelementsonlyon main diagonal

D=[d100d2]D= \begin{bmatrix} d_1&0\\0 &d_2\end{bmatrix}D=[d100d2]

IdentityMatrix

I=[1001]I=\begin{bmatrix}1&0\\0&1\end{bmatrix}I=[1001] Acts as

multiplicative identity.

SymmetricMatrix

A=ATA= A^TA=AT

Used incovariancematricesandPCA.

OrthogonalMatrix

ATA=IA^TA=IATA=I

Used inrotationandPCA.

SingularandNon-SingularMatrix

- Singular:determinant=0
- Non-singular:determinant≠0

4. ImportanceinDeepLearning

- Weightmatricesdefineneuralconnections
- Transposeused inbackpropagation
- Symmetric matricesincovariancecomputation
- Identitymatrixininitializationand normalization
- Matrixmultiplicationdrivesforward propagation

5. PythonExample

```
importnumpyasnp

A=np.array([[1,2],[3, 4]])

B=np.array([[5,6],[7, 8]])

print("Addition:\n", A + B)
print("Multiplication:\n",np.dot(A,B))
print("Transpose:\n", A.T)
```

6. CommonStudentMistakes

- Invalid matrix multiplication
- Assumingcommutativity
- Forgettingdimensionrules

Matrixoperationsandtypes formthe **computationalengine** ofDeepLearning. Correct usage ensures efficient learning, numerical stability, and accurate model implementation.

InLinearAlgebra, **norms** measurethe**sizeormagnitude**ofvectorsandmatrices. Normsare widely used in Deep Learning to:

- Measure**distancebetween vectors**
- Regularizeneuralnetworks(L1,L2regularization)
- Ensurenumericalstability
- Controloverfitting

Formally,normsprovidea**mathematicalframeworkfor"length"or"size"** inmulti- dimensional space.

2. VectorNorms

Definition

A**vectornorm** isa function||·||:Rn→R||\cdot||:\mathbb{R}^n\to\mathbb{R}||·||:Rn→Rthat satisfies:

1. **Non-negativity**:||x||≥0||\mathbf{x}||\ge0||x||≥0
2. **Definiteness**:||x||=0⟺x=0||\mathbf{x}||=0\iff\mathbf{x}=0||x||=0⟺x=0
3. **Homogeneity**:||αx||=|α|·||x||||\alpha\mathbf{x}||=|\alpha|\cdot ||\mathbf{x}||||αx||=|α|·||x||
4. **Triangleinequality**:||x+y||≤||x||+||y||||\mathbf{x}+\mathbf{y}||\le||\mathbf{x}||+ ||\mathbf{y}||||x+y||≤||x||+||y||

CommonVectorNorms

a) L1Norm(Manhattan norm)

||x||1=∑i=1n|xi|||\mathbf{x}||_1=\sum_{i=1}^n|x_i|||x||1=i=1∑n|xi|

- Measures**sumofabsolutevalues**
- Usedin**Lassoregression/L1regularization**

Example:

x=[3,−4,1]⟹||x||1=3+4+1=8\mathbf{x}=[3,-4, 1]\implies||\mathbf{x}||_1=3+4+1= 8x=[3,−4,1]⟹||x||1=3+4+1=8

b) L2Norm(Euclideannorm)

||x||2=∑i=1nxi2||\mathbf{x}||_2=\sqrt{\sum_{i=1}^nx_i^2}||x||2=i=1∑nxi2

- Measures**straight-linedistancefromorigin**
- Usedin**weightdecay/ L2regularization**
- MostcommoninDeepLearning

Example:

x=[3,4] ⟹||x||2=32+42=5\mathbf{x}=[3,4] \implies||\mathbf{x}||_2=\sqrt{3^2+4^2}= 5x=[3,4]⟹||x||2=32+42=5

c) InfinityNorm

||x||∞=max|xi|||\mathbf{x}||_\infty=\max|x_i|||x||∞=max|xi|

- Measures**largestmagnitudecomponent**
- Usefulinrobustoptimization

GeometricInterpretation

- L1 norm→ "cityblockdistance"
- L2norm→ "straightline"
- Infinitynorm→ "maximumcomponentmagnitude"

PythonExample

```
importnumpyasnp

x=np.array([3,-4,1])


print("L1norm:",np.linalg.norm(x,1))
print("L2 norm:", np.linalg.norm(x))

print("Infinitynorm:",np.linalg.norm(x,np.inf))
```

3. MatrixNorms

Definition

Matrixnorms measurethe**sizeofmatrices**.Somecommonnorms:

a) FrobeniusNorm

||A||F=∑i,jaij2||A||_F=\sqrt{\sum_{i,j}a_{ij}^2}||A||F=i,j∑aij2

- GeneralizesL2normformatrices
- Usedto**measureweightmagnitudeinneuralnetworks**

b) L1Norm(Maximumcolumnsum)

||A||1=max⁡j∑i|aij|||A||_1=\max_j\sum_i|a_{ij}|||A||1=jmaxi∑|aij|

c) InfinityNorm(Maximumrow sum)

||A||∞=max⁡i∑j|aij|||A||_\infty=\max_i\sum_j|a_{ij}|||A||∞=imaxj∑|aij|

PythonExample(MatrixNorms)

```
A=np.array([[1,2],[3, 4]])


print("Frobeniusnorm:",np.linalg.norm(A,'fro'))
print("L1 norm:", np.linalg.norm(A, 1))
print("Infinitynorm:",np.linalg.norm(A,np.inf))
```

4. PropertiesofNorms

- Non-negativity:||x||≥0||x||\ge0||x||≥0
- Scalability:||αx||=|α|||x||||\alphax||=|\alpha|||x||||αx||=|α|||x||
- Triangleinequality:||x+y||≤||x||+||y||||x+y||\le||x||+||y||||x+y||≤||x||+||y||
- Sub-multiplicative(formatrices):||AB||≤||A||·||B||||AB||\le||A||\cdot ||B||||AB||≤||A||·||B||

5. ImportanceinDeepLearning

- **Regularization**:Preventoverfitting(L1,L2)
- **Gradientnormalization**:Ensuresstableupdates
- **Distancecomputation**:Clustering,nearestneighbors
- **Weightscaling**:Controlsmagnitude inneuralnetworks

Example:
WeightdecayinL2 regularization:

Loss=MSE+λ||W||22\text{Loss}=\text{MSE}+\lambda||W||_2^2Loss=MSE+λ||W||22

6. CommonMistakes

- Confusingvectornormwithmatrix norm
- Ignoringdimensionality
- Usingwrongnorminoptimization

Eigendecompositionisa **fundamentalconceptinlinearalgebra** whereasquare matrixis expressed in terms of its **eigenvalues and eigenvectors**. It is extensively used in:

- Dimensionalityreduction(PCA)
- Covarianceanalysis
- Stabilityanalysis ofneuralnetworks

Eigendecompositionsimplifiesmatrixoperations,especiallyin **diagonalization**,whichhelpsin faster computation and better understanding of linear transformations.

2. Definitions

- **Eigenvector(vvv)**:Anon-zero vectorthatonly**changesinmagnitude**whena linear transformation is applied.

Av=λvAv= \lambdavAv=λv

- **Eigenvalue(λ\lambdaλ)**:Scalarrepresentingthe**factorbywhichtheeigenvectoris scaled**.

EigenDecomposition

ForasquarematrixAAA:

A=VΛV−1A=V \LambdaV^{-1}A=VΛV−1

Where:

- VVV = matrixofeigenvectors
- Λ\LambdaΛ= diagonalmatrixofeigenvalues

Note:Eigendecompositionexistsfor**diagonalizablematrices**.

3. ComputingEigenvaluesandEigenvectors

1. Solve **characteristic equation**:

det(A−λI)=0\det(A-\lambdaI)=0det(A−λI)=0

2. Solve(A−λI)v=0(A- \lambdaI)v=0(A−λI)v=0for eigenvectors.

Example

A=[4123]A=\begin{bmatrix}4 &1\\ 2&3\end{bmatrix}A=[4213]

1. Characteristicequation:det(A−λI)=0\det(A-\lambdaI) = 0det(A−λI)=0

det⁡[4−λ123−λ]=(4−λ)(3−λ)−2=λ2−7λ+10=0\det\begin{bmatrix}4-\lambda&1\\2&3-\lambda\end{bmatrix}=(4-\lambda)(3-\lambda)-2=\lambda^2-7\lambda+10=0det[4−λ2 13−λ]=(4−λ)(3−λ)−2=λ2−7λ+10=0

2. Eigenvalues:λ1=5,λ2=2\lambda_1=5,\lambda_2=2λ1=5,λ2=2
3. Eigenvectors:Solve(A−λI)v=0(A-\lambdaI)v=0(A−λI)v=0for eachλ\lambdaλ

4. ImportanceinDeepLearning

- **PrincipalComponentAnalysis(PCA)** useseigendecompositionofcovariance matrix to reduce dimensionality
- Analyze**stabilityofrecurrentneuralnetworks**
- Usedin**diagonalizationofweightmatrices**forefficientcomputation

PythonExample

```
importnumpyasnp

A=np.array([[4,1],[2,3]])

eigenvalues,eigenvectors=np.linalg.eig(A)
print("Eigenvalues:", eigenvalues)
print("Eigenvectors:\n", eigenvectors)
```

5. CommonMistakes

- Forgettingthateigenvectorsmustbenon-zero
- Confusingeigenvalueswithsingularvalues
- Usingnon-square matrices

SVDand PCAare **powerfullinearalgebra tools**for:

- Dimensionalityreduction
- Featureextraction
- Datacompression

Theytransformdataintoa**simpler, moreinformativespace**whileretainingessentialfeatures.

2. SingularValueDecomposition(SVD)

Definition

ForanymatrixA∈Rm×nA\in\mathbb{R}^{m\timesn}A∈Rm×n:

A=UΣVTA = U \Sigma V^TA=UΣVT

Where:

- UUU=m×mm\times mm×morthogonalmatrix(leftsingular vectors)
- Σ\SigmaΣ=m×nm\timesnm×ndiagonalmatrixofsingularvalues
- VTV^TVT=n×nn\times nn×northogonalmatrix(rightsingularvectors)

StepsofSVD

1. ComputeATAA^TAATAandAATAA^TAAT
2. FindeigenvectorsofATAA^TAATA(columnsofVVV)
3. Computesingularvalues(σi=λi\sigma_i=\sqrt{\lambda_i}σi=λi)
4. FindcolumnsofUUUusing ui=Avi/σiu_i=Av_i/\sigma_iui=Avi/σi

ImportanceinDeepLearning

- Dimensionalityreductionforlargedatasets
- Imagecompressionand reconstruction
- Datadenoising
- Featureextractionbeforetraining

PythonExample

```
importnumpyasnp

A=np.array([[1,2],[3,4],[5,6]])

U,S,Vt=np.linalg.svd(A)

print("U:\n", U)

print("S:",S)

print("Vt:\n",Vt)
```

3. PrincipalComponentAnalysis(PCA)

Definition

PCAisa**statisticaltechnique**thattransformshigh-dimensionaldatainto a**lower-dimensional space**

while preserving maximum variance.

StepsofPCA

1. Centerdata:X←X−mean(X)X\leftarrowX-\text{mean}(X)X←X−mean(X)
2. Computecovariancematrix:C=XTXC=X^T XC=XTX
3. Computeeigenvaluesand eigenvectorsofCCC
4. Selecttop-keigenvectors→principalcomponents
5. Projectdataontonew subspace

ImportanceinDeepLearning

- Reducesdimensionality→fastertraining
- Removesredundantfeatures
- Helpsvisualizehigh-dimensionaldata
- Preprocessingforneuralnetworks,clustering,andMLmodels

PythonExample

```
fromsklearn.decompositionimportPCA import
numpy as np


X=np.array([[2.5,2.4],[0.5,0.7],[2.2,2.9]])

pca = PCA(n_components=1)
X_pca=pca.fit_transform(X)
print(X_pca)
```

4. CommonMistakes

- Forgettingtocenter databeforePCA
- Confusingsingularvaluesandeigenvalues
- Selectingtoofewortoomanycomponents

Probabilitytheoryprovidesa **frameworktoquantifyuncertainty**, whichiscrucialinDeep Learning, ML, and AI for:

- Modelingpredictions
- Handlingstochastic data
- Optimizingprobabilistic models

Randomvariables, expectations, variance,andBayes'rulearecore concepts.

2. RandomVariables

- **RandomVariable(RV)**: Avariablewhosevaluedependsonchance.
- Types:
  1. **DiscreteRV**–finiteorcountableoutcomes(e.g.,dice roll)
  2. **ContinuousRV**–infiniteoutcomes(e.g.,height,temperature)

3. ProbabilityDistributions

- **DiscreteDistribution**:Probabilitymassfunction(PMF)

P(X=x)=pP(X=x) = pP(X=x)=p

- **Continuous Distribution**: Probability density function (PDF)

P(a≤X≤b)=∫abf(x)dxP(a\leX\leb)=\int_a^bf(x)dxP(a≤X≤b)=∫abf(x)dx

Examples:

- Binomial,Poisson(Discrete)
- Gaussian/ Normal(Continuous)

4. MarginalandConditionalProbability

- **MarginalProbability**:Probabilityofsingleeventignoringothers

P(X)=∑yP(X,Y)P(X) = \sum_y P(X,Y)P(X)=y∑P(X,Y)

- **Conditional Probability**: Probability of X given Y

P(X∣Y)=P(X,Y)P(Y)P(X|Y)=\frac{P(X,Y)}{P(Y)}P(X∣Y)=P(Y)P(X,Y)

5. Expectation,Variance,Covariance

- **Expectation**:Average value

E[X]=∑xP(X=x)or∫xf(x)dxE[X]= \sumxP(X=x)\quad\text{or}\quad\int xf(x) dxE[X]=∑xP(X=x)or∫xf(x)dx

- **Variance**: Spread of

distributionVar(X)=E[(X−E[X])2]Var(X)=E[(X-

E[X])^2]Var(X)=E[(X−E[X])2]

- **Covariance**:MeasureofrelationshipbetweenXand Y

Cov(X,Y)=E[(X−E[X])(Y−E[Y])]Cov(X,Y)=E[(X-E[X])(Y-E[Y])]Cov(X,Y)=E[(X−E[X])(Y−E[Y])]

6. Bayes'Rule

P(A∣B)=P(B∣A)P(A)P(B)P(A|B)=\frac{P(B|A)P(A)}{P(B)}P(A∣B)=P(B)P(B∣A)P(A)

- Usedin**NaiveBayesclassifiers**, probabilisticinference

7. InformationTheoryBasics

- **Entropy**:Measureofuncertainty

H(X)=−∑P(x)log⁡P(x)H(X)=-\sumP(x)\logP(x)H(X)=−∑P(x)logP(x)

- **KL-Divergence**:Measuredifferencebetweendistributions

DKL(P||Q)=∑P(x)log⁡P(x)Q(x)D_{KL}(P||Q)=\sumP(x)\log\frac{P(x)}{Q(x)}DKL (P||Q)=∑P(x)logQ(x)P(x)

8. ApplicationsinDeepLearning

- Probabilisticmodeling(BayesianNeuralNetworks)
- Loss functions(Cross-EntropyLoss)
- Uncertaintyquantification
- Reinforcementlearning

PythonExample(Probability&Expectation)

```
importnumpyasnp


X=np.array([1,2,3,4,5])
p=np.array([0.1,0.2,0.3,0.2,0.2])

expectation = np.sum(X*p)

print("Expectation:",expectation)
```

9. CommonMistakes

- MisunderstandingdiscretevscontinuousRV
- Forgettingtonormalizeprobabilities
- Confusingmarginalandconditionalprobability

Numerical computation and optimization are **core to training deep learning models**. Understandingoverflow,underflow,gradient-basedoptimization,andconstrainedoptimization ensures **stability and efficiency** of learning algorithms.

2. OverflowandUnderflow

- **Overflow**:Valuesexceedmaximumrepresentablenumber→infinity
- **Underflow**:Valuesapproachzerotooclosely→lossofprecision

Example:
Sigmoidfunctionσ(x)=11+e−x\sigma(x)=\frac{1}{1+e^{-x}}σ(x)=1+e−x1maycause overflow for large |x||x||x|.

- Solution:**Numericalstabilitytechniques**,e.g.,log-sum-exptrick.

3. Gradient-BasedOptimization

- **Objective**:MinimizelossL(θ)L(\theta)L(θ)
- **Gradient Descent**:

θ:=θ−η∇θL(θ)\theta:=\theta-\eta\nabla_\theta L(\theta)θ:=θ−η∇θL(θ)

- Variants:
  - StochasticGD
  - Mini-batchGD
  - Momentum,RMSProp,Adam

4. ConstrainedOptimization

- Solveoptimizationwith**constraints**:

minf(x)s.t.g(x)=0\minf(x)\quads.t.\quadg(x) =0minf(x)s.t.g(x)=0

- Lagrangemultipliers:

L(x,λ)=f(x)+λg(x)\mathcal{L}(x,\lambda)=f(x)+\lambdag(x)L(x,λ)=f(x)+λg(x)

- Usedin**regularization**and**resource-constrainedlearning**

5. LinearLeastSquares

- Objective: Fit linear model

y=Xβ+ϵy=X\beta+\epsilony=Xβ+ϵ

- Minimizesumofsquarederrors:

minβ||Xβ−y||22\min_\beta||X\beta- y||_2^2βmin||Xβ−y||22

- Solution:

β=(XTX)−1XTy\beta=(X^TX)^{-1} X^Tyβ=(XTX)−1XTy

- Usedinlinearregressionandas initializationinneuralnetworks

PythonExample

```
importnumpyasnp

X=np.array([[1,1],[1,2],[2,2]])

y=np.array([6,8,9])
beta=np.linalg.inv(X.T@X)@X.T@y
print("Least squares solution:", beta)
```

6. ImportanceinDeepLearning

- Ensures**numericalstability**
- Efficient**trainingofneuralnetworks**
- Handles**largedatasets**withouterrors
- Optimizesperformancewithconstraints

7. CommonMistakes

- Ignoring learningratescaling
- Notnormalizing data→overflow/underflow
- Usingwronggradientformulas

UNIT-2

Machine Learning:Basics and Under fitting,Hyper parameters and Validation Sets,Estimators, Bias and Variance, Maximum Likelihood, Bayesian Statistics, Supervised and Unsupervised Learning, Stochastic Gradient Descent, Challenges Motivating Deep Learning. Deep Feed forward Networks: Learning XOR, Gradient-BasedLearning,HiddenUnits,ArchitectureDesign,Back-PropagationandotherDifferentiation Algorithms.

1. IntroductiontoMachineLearning:

MachineLearning(ML)isafieldofArtificialIntelligencethatenablescomputer systemstolearn patterns from data and improve their performance on a specific task without being explicitly programmed. Instead of following rigid rules, ML models adapt based on experience.

ImportanceofMachineLearning

* Handleslargevolumesofdata

* Automatesdecisionmaking

* Learnscomplexpatterns

* Improvesaccuracyovertime

Machine Learning Process

1. DataCollection
2. DataPreprocessing
3. FeatureExtraction
4. ModelSelection
5. Training
6. Evaluation
7. Deployment

Data→Preprocessing→Model→Training→Evaluation→Prediction

2. BasicsofMachineLearning

MachineLearningsystemslearnamappingbetweeninputvariables(features)andoutputvariables (targets). The goal is to generalize well to unseen data.

Key Components

* Dataset
* Features
* Labels
* Model
* Lossfunction
* Optimizationalgorithm

Applications

* Spamdetection
* Facerecognition
* Medicaldiagnosis
* Recommendationsystems

---

3. Underfitting

Definition

Underfittingoccurswhenamodel istoosimpletocapturetheunderlyingstructureof thedata.Itfailsto learn meaningful relationships.

Characteristics

* Hightrainingerror

* Hightestingerror

* Poorpredictionaccuracy

Causes

* Simplemodel

* Insufficienttraining time

* Lackofimportantfeatures

Example
Usingastraightlinetofitcurved data.

Diagram

Datapoints:**

**

Modelline:--------
-

4. HyperparametersandValidationSets

Hyperparameters
Hyperparametersareconfigurationvaluessetbeforetrainingbegins.Theycontrolthelearningprocess.

Examples

* Learningrate

* Numberofepochs

* Numberofhiddenlayers

* Regularizationparameter

Validation Set

Avalidationsetisusedtotunehyperparametersandavoidoverfitting. Data

Split

* Trainingset:70%

* Validationset:15%

* Testset:15%

5. Estimators

Definition

Anestimatorisaruleoralgorithmusedtoestimateunknownparametersfromdata.

Types

* PointEstimator

* IntervalEstimator

PropertiesofGood Estimators

* Unbiasedness

* Consistency

* Efficienc

y Example

Samplemeanasanestimatorofpopulationmean

6. BiasandVariance

Bias

Biasistheerrorduetooverlysimplisticassumptionsinthemodel.

Variance

Varianceistheerrorcausedbysensitivitytofluctuationsinthe trainingdata. Bias–

Variance Tradeoff

* HighBias→Underfitting

* HighVariance→Overfitting

Diagram

```
Error

^

|\Variance

|\

|\______Bias
|__________>ModelComplexity
```

7. MaximumLikelihoodEstimation(MLE)

Definition

MLEisamethodusedtoestimatemodelparametersbymaximizingthelikelihoodofobservingthe given data.

Likelihood Function

P(Data|Parameters)

Key Points

* WidelyusedinstatisticsandML

* Assumesdatadistribution

* Sensitivetooutliers

Example

EstimatingmeanofGaussiandistribution

8. BayesianStatistics

Definition

Bayesianstatisticstreatsmodelparametersasrandomvariablesandupdatesbeliefsusingprobability.

Bayes Theorem

$P(\theta|X)=[P(X|\theta)\times P(\theta)]/P(X)$
Components

* Priorprobability

* Likelihood

* Posteriorprobability

Advantages

* Incorporatespriorknowledge

* Handlesuncertainty

Diagram

Prior+Data→ Posterior

9. SupervisedLearning

Definition

Supervisedlearninguseslabeleddatatolearnamappingfrominputstooutputs. Types

* Classification

* Regression

Algorithms

* LinearRegression

* LogisticRegression

* SupportVectorMachines

* K-NearestNeighbors

Diagram

Input→Model→ Output

10. UnsupervisedLearning

Definition

Unsupervisedlearningworkswithunlabeleddatatodiscoverhiddenpatterns.

Tasks

* Clustering

* DimensionalityReduction

Algorithms

* K-Means

* HierarchicalClustering

* PCA

Diagram

Data→PatternDiscovery

11. StochasticGradientDescent(SGD)

GradientDescent

Anoptimizationalgorithmusedtominimizelossfunction.

SGD

SGDupdatesmodelparametersusingonedatapointatatime.

Update

Rule$\theta=\theta-\eta\nabla$

$L(\theta)$

Advantages

* Fast

* Scalable

Disadvantages

* Noisyconvergence

Diagram

Loss

^*

|**

|*

|____> Parameters`

12. ChallengesMotivatingDeepLearning

Limitations of Traditional ML

* Manualfeatureextraction

* Poorperformanceonunstructureddata

Challenges

* High dimensionality

* Non-linearrelationships

* Largedatasets

MotivationforDeep Learning

* Automaticfeaturelearning

* Handlescomplexpatterns

Diagram

Input→HiddenLayers→ Output

DeepFeed-ForwardNetworks:

IntroductiontoDeepFeed-ForwardNetworks:
Deep Feed-Forward Networks (DFFNs), also known as Multilayer Perceptrons (MLPs), are one of the
mostfundamentalarchitecturesindeeplearning.Thesenetworksformthebackboneofmanymodern artificial intelligence systems. A feed-forward network is characterized by the unidirectional flow of information,meaning thatdata moves strictly from the inputlayer throughone or more hiddenlayers to the output layer without forming cycles or feedback loops.

The main motivation behind deep feed-forward networks is their ability to approximate complex
nonlinearfunctions.Unliketraditionalmachinelearningalgorithmsthatrelyheavilyonhandcrafted features,deepnetworkscan automaticallylearnhierarchical representationsfrom rawdata. Lower layers typically learn simple patterns, while higher layers capture increasingly abstract features.

BasicStructure

Adeepfeed-forwardnetworkconsistsofthefollowingcomponents:

* **InputLayer**:Receivesrawinputfeatures

* **HiddenLayers**:Performnonlineartransformations

* **OutputLayer**:Producesfinalprediction

Input Layer Hidden Layer 1 Hidden Layer 2

OutputLayer x1x2x3→ ooo→

oo→ o

MathematicalModelofFeed-ForwardNetworks

A feed-forward network computes a function f(x; θ), where x is the input and θ represents the parameters(weightsandbiases).Eachneuron computesaweightedsumofitsinputsfollowedbya nonlinear activation function.

Forasingle neuron:
$z=w_1x_1+w_2x_2+\ldots+w_nx_n+b$ a =

$\varphi(z)$

where:

* $w_i$are weights

* bisbias

* φisanactivationfunction

Fordeepnetworks, thiscomputationisrepeatedlayerbylayer,formingacompositionoffunctions.

Learning the XOR Problem

DefinitionofXOR

The XOR (Exclusive OR) problem is a classical example used to demonstrate the limitations of single-layerperceptronsandthepowerof multi-layernetworks.XORoutputs trueonlywhenthe two inputs are different.

InputA	InputB	XOROutput
0	0	0
0	1	1
1	0	1
1	1	0

Linear Separability

Aproblemislinearlyseparableifastraightline(orhyperplane)canseparatetheclasses.XORisnot linearly separable.

(0,1)● (1,1)○

(0,0)○ (1,0)●

Nosinglestraightlinecanseparatethetwoclasses.

XOR with a Hidden Layer

Afeed-forwardnetworkwithonehiddenlayercanlearnXORbytransformingtheinputspace.

$$\begin{tikzcd} x1 \arrow[r] & h1 \arrow[r] & {} \\ {} \arrow[r] & \text{Output} & \\ x2 \arrow[r] & h2 \arrow[r] & {} \end{tikzcd}$$

Hiddenneuronslearnintermediatefeaturessuchas ANDandOR,whicharecombinedtocompute XOR. This demonstrates that **hidden layers enable nonlinear decision boundaries

Gradient-BasedLearni

ng Learning Objective

Theobjectiveof trainingadeepfeed-forwardnetworkistominimizealossfunctionthatmeasuresthe discrepancy between predicted outputs and true targets.

Commonlossfunctionsinclude:

* MeanSquaredError (MSE)

* Cross-Entropy Loss

GradientDescentAlgorithm

Gradientdescentisanoptimizationtechniqueusedto updatenetworkparametersiteratively.

$w \leftarrow w - \eta\ (\partial L/\partial w)$

where:

* ηisthelearningrate

* $\partial L/\partial w$isthegradientoftheloss

Types of Gradient Descent

* **BatchGradientDescent**:Usesentiredataset

* **StochasticGradientDescent(SGD)**:Usesonesampleata time

* **Mini-BatchGradientDescent**:Usessmallbatches

Mini-batchgradientdescentismostcommonlyusedinpracticedue toefficiencyandstability

Hidden Units and Activation Functions

RoleofHidden Units

Hiddenunitsallowthenetworktolearninternal representationsofdata. Eachhiddenlayerextracts features at a different level of abstraction.

Withouthiddenlayers,neuralnetworksreducetolinearmodelsandfailtocapture complexpatterns.

Activation Functions

Activationfunctionsintroducenonlinearityintothenetwork.
SigmoidFunctio

n $\sigma(x)=1/(1$

$+e^{-x})$

* Outputsvaluesbetween0and1

* Suffersfromvanishinggradientproblem

Tanh Function

* Outputsvaluesbetween−1and1

* Zero-centered

ReLU(RectifiedLinearUnit)

$f(x) = \max(0, x)$

* Computationallyefficient

* Reducesvanishinggradients

Leaky ReLU

$f(x)=\max(0.01x,x)$

* SolvesdyingReLUproblem

ArchitectureDesignofDeepFeed-ForwardNetworks

Design Considerations

Keyarchitecturaldecisionsinclude:

* Numberoflayers(depth)

* Numberofneuronsperlayer(width)

* Choiceofactivationfunctions

* Regularizationtechniques

Depth vs Width

Deeparchitecturesarepreferredbecausethey:

* Learnhierarchicalfeatures

* Requirefewerparametersthanverywideshallownetworks

* Generalizebetterwhenproperlyregularized

Universal Approximation Theorem

Thetheoremstates thata feed-forwardnetworkwith asinglehiddenlayercontainingafinitenumberof neurons can approximate any continuous function. However, deep networks achieve this more efficiently.

Back-PropagationAlgorit

hm Overview:

Back-propagationisthecore algorithmusedtotraindeepfeed-forwardnetworks.Itefficiently computes gradients of the loss function with respect to all parameters.

PhasesofBack-Propagation

Forward Pass

* Inputspropagatedthroughthenetwork

* Outputandlosscomputed

Backward Pass

* Errorspropagatedbackward

* Gradientscomputedusingchainrule

* Weights updated

Forward:$x \rightarrow h_1 \rightarrow h_2 \rightarrow y$

Backward:$\leftarrow \delta_1 \leftarrow \delta_2 \leftarrow \delta y$

Mathematical Explanation

For a weight w in layer l:

$\partial L/\partial w=(\partial L/\partial a)(\partial a/\partial z)(\partial z/\partial w)$

Thislayereddecompositionmakestrainingdeepnetworkscomputationallyfeasible.

Other Differentiation Algorithms

Numerical Differentiation
Usesfinitedifferencestoapproximategradients.

* Simplebutinaccurate

* Computationallyexpensive

Symbolic Differentiation

* Computesexactderivatives

* Suffersfromexpressionexplosion

Automatic Differentiation

Automaticdifferentiationisusedinmoderndeeplearningframeworks suchasTensorFlowandPyTorch.

Advantages:

* Exactgradients

* Efficientcomputation

* Scalestodeeparchitecture

ChallengesinTrainingDeepFeed-ForwardNetworks

* Vanishingandexplodinggradients

* Overfitting

* Choiceofhyperparameters

* Computationalcomplexit

Techniquessuchas normalization,betterinitialization,andadvancedoptimizershelpovercomethese challenges.

Conclusion

Deep feed-forward networks form the foundation of deep learning. Their ability to learn nonlinear, hierarchical representations makes them powerful tools for solving complex real-world problems. Understanding XOR learning, gradient-based optimization, hidden units, architecture design, back-propagation,anddifferentiationalgorithmsisessentialformasteringmodernneuralnetworks.

UNIT-III:RegularizationandOptimizationforDeepLearning

Regularization for Deep Learning: Parameter Norm Penalties, Norm Penalties as Constrained Optimization, Regularization and Under-Constrained Problems, Dataset Augmentation, Noise Robustness, Semi-Supervised Learning, Multi-Task Learning, Early Stopping, Parameter Tying and Parameter Sharing, Sparse Representations, Bagging and Other Ensemble Methods, Dropout, Adversarial Training, Tangent Distance, Tangent Prop and Manifold Tangent Classifier. Optimization for Training DeepModels:Pure Optimization,Challenges inNeural Network Optimization,BasicAlgorithms, ParameterInitializationStrategies,AlgorithmswithAdaptiveLearningRates,ApproximateSecond-Order Methods, Optimization Strategies and Meta-Algorithms

ParameterNormPenalties

Definition:

Parameternormpenaltiesaretechniquesused indeep learningto **preventoverfitting** byadding a penaltyto the loss function based onthe **size (norm) of the weights**. Large weights are discouraged, making the model simpler and more generalizable.

KeyPoints:

1. **L1Regularization (Lasso):**
   - Adds thesumofabsolutevalues ofweights totheloss.
   - Encourages**sparsity**→someweightsbecomeexactlyzero.
2. **L2Regularization (Ridge):**
   - Adds thesumofsquaredweightstotheloss.
   - Discourages**large weights** butrarelysetsthemexactlytozero.

Example(L2Regularization):

Originallossfunction(MeanSquaredError,MSE):

MSE=1n∑i=1n(yi−yi^)2\text{MSE}=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y_i})^2MSE=n1i=1∑n(yi−yi^)2

WithL2 penalty:

J(θ)=MSE+λ∑iθi2J(\theta)=\text{MSE}+\lambda\sum_{i}\theta_i^2J(θ)=MSE+λi∑θi2

Where:

- θi\theta_iθi→weights ofthemodel
- λ\lambdaλ → regularization parameter (controls penalty strength)

Largerλ\lambdaλ→strongershrinkageofweights→preventsoverfitting.

DiagramExplanation:

- **L1Regularization:**Weightvaluesshrinktowardzero,somebecomeexactly**0**(sparse).
- **L2Regularization:**Weightvaluesshrink **smoothly**,butrarelybecomeexactlyzero.

Conceptual Illustration:

```
Weightmagnitude
|
|          L2
|        /
|       /
|      /
|     /
|    /
|/
|/
|/
+---------------------------------->
        L1
```

- L1→diamond-likepenalty→ sparsity
- L2 → circle-likepenalty→ smoothshrinkage

NormPenaltiesasConstrainedOptimization

Definition:

Regularizationcanbe interpretedas**minimizingthelossfunctionwhilekeepingtheweights within a certain norm constraint**. This means we only allow weight values that satisfy a specific size limit, which helps prevent overfitting.

Mathematically:

min⁡θL(θ)subjectto∥θ∥p≤c\min_\thetaL(\theta)\quad\text{subjectto}\|\theta\|_p\lecθmin L(θ)subject to ∥θ∥p≤c

Wherep=1p =1p=1 for L1 andp=2p =2p=2 forL2.

KeyPoints:

1. **L2Regularization (Ridge):**

- Constraint:‖θ‖2≤c\|\theta\|_2\lec‖θ‖2≤c
- Geometrically→ **circular(spherical)region**inweightspace.
- Encouragessmallbutnon-zeroweights.

2. **L1Regularization (Lasso):**
   - Constraint:‖θ‖1≤c\|\theta\|_1\lec‖θ‖1≤c
   - Geometrically→**diamond-shapedfeasibleregion**.

- Encourages**sparseweights**, someexactlyzero.

3. **Purpose:**
   - Helpsvisualizehowregularizationlimitsweightgrowth.
   - ShowswhyL1leadstosparsityandL2to smoothshrinkage.

Example:

SupposewewanttominimizethelosssubjecttoanL2constraint:

∑iθi2≤1\sum_i\theta_i^2\le1i∑θi2≤1

- Onlyweight combinationsthat satisfy∑iθi2≤1\sum_i\theta_i^2\le1∑iθi2≤1are allowed.
- Theoptimizerfindsthebestweights**insidethe circle**.

DiagramExplanation:

- **L2Constraint:**Circlein2Dweightspace→allpoints insidecirclesatisfy ∥θ∥2≤c\|\theta\|_2\lec∥θ∥2≤c.
- **L1Constraint:**Diamondin2Dweightspace→allpoints insidediamond satisfy ∥θ∥1≤c\|\theta\|_1\lec∥θ∥1≤c.

ConceptualIllustration:

```
Weightθ1
  |
  |        *(L2 circle)
  |       /\
  |      /    \
  |     *      *
  |    /        \
  +------------------------------------------  θ2
       \        /
        *      *
         \    /
          \/
           *(L1 diamond)
```

- The**circle**→smoothweightshrinkage(L2
- The**diamond**→sparsity(L1)

3. RegularizationandUnder-ConstrainedProblems

Definition:

Whenthe**numberofmodelparametersexceeds thenumberoftrainingsamples**, the model is

under-constrained—there are infinitely many solutions that fit the training data. Regularizationhelps**selectsimplersolutions**thatgeneralizebetter.

KeyPoints:

1. **Reducesvariance:**Preventsthemodelfromfittingnoiseintraining data.
2. **Encouragessmoothnessandstability:** Solutionsaresimplerandlesssensitivetosmall input changes.

Example:

- Polynomialregressionwith**degreehigherthan the numberof samples**.
- Withoutregularization→modelfitsallpointsexactly→ overfitting.
- WithL2 or L1regularization→ smoother curve,bettergeneralization.

DiagramExplanation:

- Overfittingcurve(high-degreepolynomial) →passesthroughallpoints.
- Regularizedcurve→smoother,approximates overalltrend.

ConceptualIllustration:

```
y
|
|      *        *      *
|   *     *        **
| *         *          *
|------------------------------------------------>x
OverfittingvsRegularized
```

4. DatasetAugmentation

Definition:

Datasetaugmentationinvolves **artificiallyexpandingthetrainingdataset**byapplying **transformations**toexistingsamples.Thisimprovesmodelgeneralizationandrobustness.

KeyPoints:

1. Commonimagetransformations:**rotation,flipping,cropping,scaling**.
2. Improvesmodelrobustnesstovariationsin input.
3. Reducesoverfittingonsmalldatasets.

Example:

- CIFAR-10 images:Originalimage→rotated,flipped,orcroppedversionsaddedto training set.

DiagramExplanation:

- Originalimage→rotated+flipped→multipleaugmentedsamples.

ConceptualIllustration:

```
[OriginalImage]→rotate15°→[RotatedImage]
                    →horizontalflip→[FlippedImage]
```

5. NoiseRobustness

Definition:

Noiserobustnessinvolves**addingnoisetoinputsorlayersduringtraining**to improve model generalization. The model learns to be less sensitive to small variations.

KeyPoints:

1. Typesofnoise:**inputnoise**(e.g.,Gaussiannoise), **dropoutnoise**(randomlyturningoff neurons).
2. Helpsmodel**generalize**tounseenornoisydata.

Example:

- MNISThandwrittendigits: AddGaussiannoisetoimagesduringtraining→model learns to classify even noisy digits.

DiagramExplanation:

- Originalimagevs noisy image

ConceptualIllustration:

```
[OriginalImage]→+GaussianNoise→[NoisyImage]
```

- The modelseesbothversionsduringtraining→learnsrobustness.

6. Semi-SupervisedLearning

Definition:

Semi-supervised learning isa learningapproachthat usesa **smallamountoflabeleddata together with a large amount of unlabeled data** to improve model performance.

KeyPoints:

1. Combines**supervised** (labeleddata)and **unsupervised** (unlabeleddata)learning.

2. Assumesunlabeled datahelpsrevealthe**underlying datastructure**.
3. Reduceslabelingcostandimprovesgeneralization.
4. Commontechniques:
   - **Pseudo-labeling:**Modelassigns labelstounlabeled data.
   - **Consistencyregularization:**Modelpredictionsshouldremainstableunderinput perturbations.

Example:

- CIFAR-10datasetwithonlyasmallportionoflabeledimages.
- The modellearnsfrombothlabeledandunlabeled imagestoimproveclassification accuracy.

DiagramExplanation:

- Labeleddataandunlabeleddataarefedintothe model.
- Themodellearnsjointlyandproducespredictions.

```
LabeledData+UnlabeledData→Model→Predictions
```

7. Multi-TaskLearning

Definition:

Multi-tasklearningisatechniquewherea modellearns**multiplerelatedtaskssimultaneously** usingsharedinternalrepresentations.

KeyPoints:

1. Improvesgeneralizationbysharingknowledgeacrosstasks.
2. Actsasa**regularizer**,reducingoverfitting.
3. Shared featurescapturecommonpatternsamong tasks.

Example:

- Predicting**ageandgender**fromfaceimagesusingasingleneuralnetwork.

DiagramExplanation:

- Hiddenlayersaresharedacrosstasks.
- Eachtaskhasa separateoutputlayer.

```
Input→SharedHiddenLayers→AgeOutput
                          →GenderOutput
```

8. EarlyStopping

Definition:

Earlystopping isaregularizationtechniquewhere training is **stoppedwhenvalidationloss stops improving**, even if training loss continues to decrease.

KeyPoints:

1. Prevents overfitting.
2. Actsasan**implicitregularizer**.
3. Usesvalidationsetperformancetodecidestoppingpoint.

Example:

- Traininglossdecreasescontinuously.
- Validationlossstartsincreasing→trainingisstopped.

DiagramExplanation:

- Graphshowstraininglossdecreasing.
- Validationlossdecreasesinitially,thenincreases→earlystoppingpoint.

```
Loss
|
|\Validation
|\StopHere
|     \
|Training
+------------------------------------>Epochs
```

9. ParameterTying/ParameterSharing

Definition:

Parametertying(orsharing) forces**multipleparameterstohavethesamevalue**,reducing model complexity.

KeyPoints:

1. Reducesnumberoffree parameters.
2. Improvesgeneralization.

3. Commonlyusedin**CNNsandRNNs**.

Example:

- InCNNs,the**sameconvolutionfilter**isappliedacrossdifferent spatiallocationsofan image.

DiagramExplanation:

- Asinglefilter slidesacrosstheimagetoextractfeatures.

```
[Filter]→→→acrossimage
```

10. SparseRepresentations

Definition:

Sparserepresentationsencourage **mostneuronactivationstobezeroornearzero**,allowing only a few neurons to be active at a time.

KeyPoints:

1. Improvescomputationalefficiency.
2. Enhancesgeneralization.
3. Oftenenforcedusing**L1regularization**.
4. Commonin**sparseautoencoders**.

Example:

- Sparsecodingofimagepatcheswhereonlyafewfeaturesrepresenteachpatch.

DiagramExplanation:

- Denseactivation:manyneuronsactive.
- Sparseactivation:fewneuronsactive.

```
Dense:[11111]
Sparse:[00100]
```

11. BaggingandEnsembleMethods

Definition:

Bagging(BootstrapAggregating)andensemblemethodscombinethepredictionsof**multiple models** to produce a more accurate and stable final prediction.

KeyPoints:

1. Reduces**variance**ofpredictions.
2. Improvesrobustnessand generalization.
3. Eachmodelistrainedonadifferent**bootstrapsample**ofdata.
4. Finaloutputisobtained by**averagingorvoting**.

Example:

- **RandomForest:** Anensembleofdecisiontreestrainedondifferent subsetsofdataand features.

DiagramExplanation:

- Multiplemodelsmakepredictions.
- Predictionsareaggregatedto producethefinaloutput.

```
Model1→
Model2→Aggregation→FinalOutput Model 3

→
```

12. Dropout

Definition:

Dropoutisaregularizationtechniquewhere **randomneuronsaretemporarilyremoved during training**, forcing the network to learn robust features.

KeyPoints:

1. Prevents **co-adaptation** ofneurons.
2. Actsliketraining**manysubnetworks**.
3. Controlledbya**dropoutprobability(p)**.
4. Usedonlyduringtraining,nottesting.

Example:

- Applyingdropouttofullyconnectedlayers inaneuralnetwork.

DiagramExplanation:

- Someneuronsarerandomlyremovedduringtraining.

```
OOO
\X/
OOO
```

```
(X=droppedneuron)
```

13. AdversarialTraining

Definition:

Adversarialtraining improvesmodelrobustnessby **trainingonadversarialexamples**,which are inputs slightly modified to fool the model.

KeyPoints:

1. Smallperturbationsaddedtoinputsmaximizeloss.
2. Improvesresistancetoadversarialattacks.
3. Enhancesmodelrobustnessandsecurity.

Example:

- **FGSM (FastGradientSignMethod):**Addsperturbationsinthedirectionofthe gradient.

DiagramExplanation:

- Originalimageandadversarialimageappearsimilarbutcausedifferentpredictions.

```
Original Image → Correct Prediction
AdversarialImage→WrongPrediction
```

14. TangentDistance&ManifoldMethods

Definition:

These methodscapture**invariancesalongthedatamanifold**,allowingthe modeltoignore small transformations such as rotation or translation.

KeyPoints:

1. **TangentDistance:** Measuressimilarityinvarianttosmalltransformations.
2. **TangentPropagation:**Penalizessensitivityalongmanifolddirections.
3. Improvesinvarianceandgeneralization.

Example:

- Handwrittendigitsthatarerotatedorshiftedbutrepresentthesamenumber.

DiagramExplanation:

- Datapointslieonasmoothcurvedmanifold.

```
.   .   .
  .   .
     .
(Manifold)
```

PARTB:OptimizationforDeepLearning

15. PureOptimization

Definition:

Pureoptimizationfocusessolelyon**minimizingthelossfunction**,without considering statistical or generalization properties.

KeyPoints:

1. Uses**gradientdescent**anditsvariants.
2. Includes**second-ordermethods**likeNewton's method.
3. Concernedonlywithfindingminima.

Example:

- Minimizingmeansquarederrorin**linear regression**.

DiagramExplanation:

- Optimizationmovesdownhillonthelosssurfacetowardminimum.

```
Loss
|
|   \
|    \
|        \
+------------------------>Parameters
```

16. OptimizationChallenges

Definition:

Optimizationchallengesarise because **neuralnetworklosssurfacesarehighlynon-convex**, making training difficult and unstable.

KeyPoints:

1. **Vanishinggradients:**Gradientsbecomeverysmall,slowing learningindeepnetworks.
2. **Explodinggradients:**Gradientsgrowtoolarge, causingunstableupdates.
3. **Saddlepoints:**Flatregionswheregradientsarenearzerobutnotminima.
4. **Localminima:** Optimizationmayconvergetosub-optimalsolutions.

DiagramExplanation:

- Losssurfaceshowsvalleys,peaks, andsaddlepointswhereoptimizationslows.

```
Loss
|
     __
|   /\
|/         \
+----------------------------------------->Parameters
```

17. BasicOptimizationAlgorithms

Definition:

Basicoptimizationalgorithmsare**gradient-basedmethods**usedto minimizethelossfunction during training.

KeyPoints:

1. **BatchGradientDescent:**Usesentiredataset perupdate(slowbutstable).
2. **StochasticGradientDescent(SGD):**Updatesusingonesampleat atime(fast but noisy).
3. **Mini-batchGradientDescent:**Usessmallbatches;mostcommonlyused.

DiagramExplanation:

- Optimizationstepsmovedownhilltoward minimum.

```
Loss
|
|   •
|    •
|      •
+----------------------------------------->Parameters
```

18. ParameterInitialization

Definition:

Parameterinitializationistheprocessofchoosing**initialweightvalues**beforetrainingbegins.

KeyPoints:

1. **Zeroinitializationfails:**Causesneuronstolearnidentical features.
2. **XavierInitialization:**Suitablefortanhorsigmoidactivations.
3. **HeInitialization:** Designed forReLUactivations.
4. Helpsprevent vanishingorexplodinggradients.

DiagramExplanation:

- Properinitializationmaintainsbalancedsignalflowacrosslayers.

```
Input→Layer1→Layer2→Output (Stable
weight flow)
```

19. AdaptiveLearningRateAlgorithms

Definition:

Adaptivelearningratealgorithms**automaticallyadjustlearningrates**duringtrainingforfaster convergence.

KeyPoints:

1. **AdaGrad:** Adaptslearningrateperparameter.
2. **RMSProp:**FixesAdaGrad'saggressivedecay.
3. **Adam:**Combines momentumand RMSProp.
4. Providesfasterandmorestableconvergence.

DiagramExplanation:

- Learningratedecreasessmoothlyoveriterations.

```
LearningRate

|

|\

|\

|\

+----------------------------------------->Iterations
```

20. ApproximateSecond-OrderMethods

Definition:

Approximatesecond-ordermethodsuse**curvature(secondderivative)information** toimprove optimization efficiency.

KeyPoints:

1. **Newton'sMethod:**UsesHessianmatrix.
2. **Quasi-NewtonMethods:**ApproximateHessian(e.g.,BFGS).
3. Fasterconvergencebut **computationallyexpensive**.
4. Rarelyusedfor verydeepnetworks.

DiagramExplanation:

- Curvatureinformationhelpstakesmarter stepstowardminimum.

```
LossCurve

|

|/\

+------------------------------------------>Parameters
```

21. OptimizationStrategies&Meta-Algorithms

Definition:

Optimizationstrategiesandmeta-algorithmsenhance**convergencespeedandtrainingstability**.

KeyPoints:

1. **Momentum:**Accelerateslearningincorrect direction.
2. **NesterovAcceleratedGradient:**Looksaheadbeforeupdating.
3. **Gradientclipping:**Preventsexplodinggradients.
4. **Learningrateschedules:**Reducelearningrateovertime.

DiagramExplanation:

- Momentumsmoothsoptimizationpathcompared to standardGD.

```
Standard GD:zig-zag path
Momentum:     smoothcurvedpath
```

UNIT-IV

Convolution Networks: TheConvolution Operation,Pooling,Convolution,Basic Convolution Functions, Structured Outputs, Data Types, Efficient Convolution Algorithms,RandomorUnsupervisedFeatures,BasisforConvolutionNetworks.

1. BasisforConvolutionalNetworks

Definition:

CNNs are specialized deep neural networks for **grid-structured data** like images, video, and audio.Theyexploit spatial,temporal,orvolumetricstructuretoreduceparametersandimprove learning efficiency.

KeyPrinciples:

1. **SparseConnectivity(LocalReceptive Fields)**
   - Eachneuronconnectsonlytoa**small localregion**intheinput.
   - **Example:** Ina32×32 image,a5×5 kernelconnectstoonly25 pixelsatatime.
   - Captures**localfeatures**like edgesor corners.
   - Reducesparameters→lowerscomputationalcostandmemoryusage.
2. **ParameterSharing (WeightReuse)**
   - Thesamekernel/filterisappliedacrossdifferentinputlocations.
   - Ensures**translation invariance**:patternsarerecognizedregardlessoflocation.
   - Reducesparametersdrastically(e.g.,5×5×3kernelvs32×32×3 fullyconnected layer).
3. **TranslationEquivariance**
   - Shiftinginput→proportionalshiftinoutputfeaturemap.
   - Criticalforimageandvideotaskswhereobjectsmaymoveslightly.
4. **HierarchicalFeatureLearning**
   - Earlylayersdetect**low-levelfeatures**(edges,textures).
   - Middle layersdetect**mid-levelfeatures**(eyes,noses).
   - Deeplayersdetect**high-levelconcepts**(faces, objects).

Applications:

- Imageclassification,objectdetection,facialrecognition,videounderstanding.

Example:

- Edgedetectionusingasmallfilterremainseffectiveregardlessofedge position.

Diagram(ASCII)

```
InputImage            ConvolutionFilter

[xxxxx][www]    [xxxxx]--->[www]
[xxxxx][www]
```

Aspect	FullyConnectedNN	CNN
Connectivity	Dense	Sparse
Parameters	VeryHigh	Low
SpatialInfo	Ignored	Preserved
Translation	Poor	Excellent

2. TheConvolutionOperation(Expanded)

Definition:

Convolution isa **mathematicaloperation** that combinesan input signal(or image)withakernel(filter)toproduceafeaturemap.InCNNs,convolution is used to **extract local features** from data.

MathematicalFormulation:

- **1DDiscreteConvolution:**

(f∗g)(n)=∑mf(m)g(n−m)(f*g)(n)=\sum_mf(m) g(n-m)(f∗g)(n)=m∑ f(m)g(n−m)

- **2DConvolution(forimages):**

S(i,j)=∑m∑nI(i+m,j+n)⋅K(m,n)S(i,j)=\sum_m\sum_nI(i+m, j+n)\cdot K(m,n)S(i,j)=m∑n∑I(i+m,j+n)⋅K(m,n)

Where:

- III=inputimage
- KKK=convolutionkernel
- SSS=outputfeaturemap

Cross-correlationinCNNs:

- CNNstypicallyuse**cross-correlation**,nottrue convolution:

S(i,j)=∑m∑nI(i+m,j+n)⋅K(m,n)S(i,j)=\sum_m\sum_nI(i+m, j+n)\cdot K(m,n)S(i,j)=m∑n∑I(i+m,j+n)⋅K(m,n)

Kernelisnotflipped,simplifiescomputation,same learningeffect

KeyConcepts:

1. **Receptive Field**
   - Areaofinput that a neuron“sees”
   - Largerkernels→ largerreceptive field
   - Deepernetworks→largereffectivereceptivefield
2. **Stride**
   - Determinesstep sizeofkernel
   - Largerstride→smallerfeaturemap,lesscomputation,but may miss fine details
3. **Padding**
   - Zero-paddingmaintainsoutputdimensions
   - Types:**Validpadding**(nopadding),**Samepadding**(output size = input size)

Examples:

- EdgeDetection:Verticalorhorizontaledgedetectionfiltershighlight patterns in images.
- Smoothing:Averagefilterreducesnoise.
- Sharpening:Kernelemphasizestransitions inpixelintensity.

Diagram(ASCII):

```
InputPatch(3x3)          Filter(3x3)
[123]                   [10-1]
[456]         *         [10-1]
[789]                   [10-1]

Output=sumofelement-wiseproducts
```

Applications:

- Imageprocessing:Edgedetection,blur,sharpening
- FeatureextractioninCNNs
- Audioprocessing(1Dconvolution)

Table:Convolution Parameters

Parameter	Description	Effect
KernelSize	Dimensionsofthe filter	Determinesreceptive field
Stride	Stepsizeforkernelmovement	Controlsoutputsize
Padding	Zero-padding ofinput	Maintainsoutputsize
Number of Filters	Number of kernels	Determinesdepthoffeature map

3. ConvolutioninCNNs(Expanded)

Definition:

InCNNs,convolutionlayersapplymultiple **learnedfilters**toinputstoproduce**featuremaps**, capturing hierarchical patterns in the data.

KeyDetails:

1. **Multi-channelInputs**
   - RGBimages:kernelspansall3 channels
   - Eachfilterproduces**singlefeaturemap**
   - Multiple filters→ multi-channelfeaturemaps
2. **HierarchicalFeatureLearning**
   - **Shallowlayers:**Detectedges,corners,textures
   - **Intermediatelayers:**Detectmotifs,objectparts
   - **Deeplayers:**Detectcomplexobjects(faces,vehicles)
3. **Strideand PaddinginCNNs**
   - Stridereducescomputationalloadand featuremapsize
   - Paddingpreventsshrinkingoffeaturemapsafter convolution
4. **ParameterEfficiency**
   - Convolutionuses **weight sharing**→fewerparametersthanfullyconnected layers
   - Reducesoverfitting

Example:

- Facedetection:
  - Earlylayers→horizontal/verticaledges
  - Middle layers→eyes,nose,mouth
  - Finallayers→completefacerepresentation

Diagram(ASCII):

```
InputImage(RGB)
```

```
   ↓
ConvolutionFilters(3x3)applied

   ↓
FeatureMaps(MultipleChannels)
```

Applications:

- Imageclassification
- Object detection
- Sceneunderstanding

Table:Convolution LayerParameters

Parameter	Description	Example
Filters	Numberoflearnedkernels	32filtersforfirstlayer
KernelSize	Size ofeachkernel	3x3,5x5
Stride	Stepsizeofconvolution	1,2
Padding	Zero-padding method	Same,Valid

4. BasicConvolutionFunctions(Expanded)

Definition:

- CNNsuse**1D, 2D, or3Dconvolutions** dependingoninput type.
- ThesefunctionsallowCNNstohandlesequential,spatial,andspatiotemporaldata.

Typesof Convolution:

1. **1DConvolution**
   - Input:Sequentialdata
   - Example:Audiowaveform
   - Captures**temporalcorrelations**
2. **2DConvolution**
   - Input:Images(height×width)
   - Example:MNISTdigit classification
   - Captures**spatialfeatures**likeedgesandtextures
3. **3DConvolution**
   - Input:Videoframesorvolumetricdata
   - Example:Actionrecognitioninvideosequences
   - Captures**spatiotemporaldependen**

cies Extended Concepts:

- **DepthwiseConvolution:**
  - Separateconvolutionper channel→ reducescomputation
  - UsedinMobileNet
- **SeparableConvolution:**
  - Factorizesconvolutioninto**depthwise+pointwise**
  - Reducesparameterswhilemaintaining performance

Examples:

- 1D:Heartbeatanomalydetection
- 2D:Dogvscatimageclassification
- 3D:Predictinghumanmotionfromvideoframes

Diagram(ASCII):

```
1D:[x1x2x3x4x5] 2D: [
x11 x12 x13 ]

    [x21x22x23]

3D:[x111x112x113](time/depth)
```

Table:Convolution Types

Type	Input	CNN Function	Application
1D	Audio	1DConv	Speechrecognition
2D	Image	2DConv	Classification
3D	Video	3DConv	Actionrecognition

---

5. Pooling(Expanded)

Definition:

- Poolingreduces**spatialsize**offeaturemapswhileretaining**important features**.
- Provides**translationinvariance** andreducescomputation.

PoolingTypes:

1. **Max Pooling**
   - Selects maximumvalueineachregion
   - Preservesstrongestfeature
   - Commoninimagerecognition
2. **AveragePooling**
   - Computesaveragevalueineachregion
   - Smoothsfeaturemaps

3. **GlobalPooling**
    - Reducesentirefeaturemaptoasinglevalueperchannel
    - Oftenused before fullyconnectedlayers

KeyConcepts:

- **Strideand PoolSize:**
    - Poolingwindowsize(2x2,3x3)determinesthedownsamplingfactor
    - Stridecontrolsmovementofpooling window
- **EffectonFeature Maps:**
    - Reducesoverfitting
    - Reducescomputationalload
    - Slightlossofspatialresolution

Example:

- InCIFAR-10classification,2x2 maxpoolingreduces featuremap from32x32→16x16, retaining key features.

Diagram(ASCII):

```
Input Feature Map (4x4)       2x2MaxPooling
[1  3  2  1]                 [3  4]

[4652]      ---------->      [89]

[7283]

[1490]
```

Table:PoolingComparison

Type	Operation	Effect	Use Case
Max	Maxvalue	Retainsstrongestactivation	Classification,objectdetection
Average	Mean	Smoothsfeatures	Lesscommon,regression
Global	Mean/Maxofall	Singlevector	Fullyconnectedlayersinput

Applications:

- Imageclassification(MNIST, CIFAR-10)
- Objectdetection(YOLO,SSD)
- ReducingmemoryandcomputationindeepCNNs

6. StructuredOutputs

Definition:

StructuredoutputsinCNNsare**outputsthatpreserve the spatialortemporalstructure**ofthe input,insteadofcollapsing it into asinglescalarorclasslabel.Thisiscrucialfortaskswhere every element (pixel, time-step, voxel) requires a prediction.

KeyConcepts:

1. **FullyConvolutionalNetworks(FCNs)**
   - Replace fullyconnectedlayerswithconvolutionallayers.
   - Maintains**spatialcorrespondence**betweeninputandoutput.
   - Canhandle**arbitrary-sized inputs**.
2. **ApplicationsofStructuredOutputs:**
   - **SemanticSegmentation:**Pixel-levelclassification(e.g.,labelingroads,cars, pedestrians in self-driving datasets).
   - **DepthEstimation:**Predictdepthforeachpixelinanimage.
   - **HeatmapPrediction:**Locateobjectsor keypoints(e.g.,faciallandmarks).
   - **ObjectDetection/Localization:**Predictboundingboxesorprobability maps.
3. **TechniquesforMaintainingResolution:**
   - **Upsampling/TransposedConvolution:**Increasesfeaturemapsizetomatch input.
   - **SkipConnections:**Combineshigh-levelsemanticinformationwithlow-level spatial details.
   - **DilatedConvolutions:**Expandsreceptive fieldwithoutlosingresolution.

Example:

- Inmedicalimaging,aCNNwithstructuredoutputidentifiestumorregions **pixel-by-pixel**, instead of just classifying the image as “tumor”or “no tumor.”

Diagram(ASCII):

```
InputImage(HxW)

        ↓
Conv+PoolLayers→FeatureMaps

        ↓
Upsampling/TransposedConv

        ↓

Pixel-wiseOutputMap(HxW)
```

Table:StructuredvsScalarOutput

OutputType	**Description**	**Example**
Scalar	Single label	Imageclassification
Vector	Multiplelabels	Multi-labelclassification
Structured	Pixel-wiseorsequence-wise	Semanticsegmentation,videoprediction

Importance:

- Maintains**spatialdependencies**.
- Enables**densepredictiontasks**that areessentialforrobotics,autonomousdriving,and medical imaging.

---

7. DataTypesinConvolutionalNetworks

Definition:

CNNsaredesignedto handle **grid-structureddata**, whichcanbe1D,2D,or3Ddependingon the application.

KeyConcepts:

1. **1DData:**
    - Examples:Audiosignals,time-seriesdata,sensorreadings.
    - Processedusing**1Dconvolution**alongthetemporalaxis.
    - Captureslocaltemporalcorrelations.
2. **2DData:**
    - Examples:Images(height×width).
    - Processedusing**2Dconvolution**(spatialfilters).
    - Capturesedges,corners,textures,andpatterns.
3. **3DData:**
    - Examples:Video(height×width×time),volumetricMRI/CT scans.
    - Processedusing**3Dconvolution**(spatiotemporalorvolumetricfilters).
    - Capturesmovementpatterns, depthfeatures, andspatial-temporalcorrelations.
4. **Mini-Batching:**
    - Inputsareprocessedin**batches**forcomputational efficiency.
        - Standardtensorformat:`(batch_size,channels,height,width)`for2D,

          `(batch_size,channels,depth,height,width)`for3D.

Example:

- Videoactionrecognitionuses3Dconvolutionto capturemotionacrossframes.
- ECGsignalclassificationuses1Dconvolutionalongthetime axis.

Diagram(ASCII):

```
1D:[x1x2x3x4x5] 2D: [
x11 x12 x13 ]

    [x21x22x23]

3D:[x111x112x113](time/depth)
```

Table: DataTypesand Applications

DataType	Dimensionality	CNNType	Application
Audio	1D	1DConvolution	Speechrecognition,ECGanalysis
Image	2D	2DConvolution	Objectdetection,imageclassification
Video	3D	3DConvolution	Actionrecognition,medicalimaging

Importance:

- Understandingthe **datatype**ensurescorrectconvolutiondesignandarchitecturechoice.
- EnablesCNNstoadaptto**variedreal-worldinputs**acrossdomains.

---

8. EfficientConvolutionAlgorithms

Definition:

Convolutionsarecomputationallyintensive,especiallyfor **deepnetworksandlargeinputs**. Efficient algorithms accelerate training and inference.

KeyTechniques:

1. **NaiveConvolution:**
   - Directcomputation→highcomputationalcost.
   - Suitableonly forsmalldatasetsor prototypes.
2. **im2colTransformation:**
   - Convertsconvolutioninto**matrixmultiplication**.
   - AllowshighlyoptimizedGPU linearalgebra libraries(BLAS).
   - WidelyusedinmodernCNNframeworks.
3. **FFT-basedConvolution:**
   - Applies**FastFourierTransform**toinputand kernel.
   - Reducescomputationalcomplexityfor**largekernels**.
   - Lessefficientforsmallkernels(3×3)commonlyusedinCNNs.
4. **WinogradAlgorithm:**
   - Reducesthenumber of**multiplications**required.
   - Veryeffectivefor smallkernels(3×3,5×5).
   - Usedextensivelyin**high-performanceCNN libraries**.

Example:

- AResNet50modelwith3×3convolutionlayersuses **Winogradalgorithm**toreduce GPU computation.

Table:EfficientConvolution Methods

Method	Description	Best UseCase
Naive	Direct convolution	Smallnetworks
im2col	Transformtomatrixmultiplication	GPUacceleration
FFT	Convolutioninfrequencydomain	Large kernels
Winograd	Reducemultiplications	Smallkernels(3x3)

Importance:

- Allows**real-timeinference**inimage/video applications.
- Enables**deeperCNNs**to betrained onlargedatasets.

---

9. RandomorUnsupervisedFeatures

Definition:

- Randomorunsupervisedfeaturesare**learnedwithoutexplicitlabels**.
- Usefulwhen labeled dataisscarceor unavailable.

KeyPoints:

1. **Random Filters:**
   - Initiallayerswith**randomweights**cancapturebasicpatterns(edges, textures).
   - Surprisinglyeffectiveduetonaturalstatisticsof images.
2. **UnsupervisedFeatureLearning:**
   - Techniques:Autoencoders,Sparsecoding,K-meansclustering.
   - Learn**filters**fromdata distribution instead of labels.
   - CanpretrainCNN layers,followed bysupervisedfine-tuning.
3. **ComparisonwithSupervisedLearning:**
   - Supervisedlearninggenerallyoutperformsunsupervisedmethodsonlargelabeled datasets.
   - Unsupervisedpretrainingisusefulfor**representationlearning**whenlabelsare limited.

Example:

- AutoencodertrainedonunlabeledimagesproducesGabor-likefilters inearlylayers.

Table:FeatureLearningMethods

Method	LabelsRequired	Performance	Example
Random	No	Limited	Randomedgefilters

Method	LabelsRequired	Performance	Example
Unsupervised	No	Moderate	Autoencoderforfeature extraction
Supervised	Yes	Best	CNNclassification

Importance:

- Supports**semi-supervisedlearning**,**pretraining**,andunderstanding**statistical properties** of natural images.
- Basisfor**transferlearning**indeepCNNs.

Unit-5

SequenceModelling:RecurrentandRecursiveNeural Networks

SequenceModelling:RecurrentandRecursiveNets:UnfoldingComputationalGraphs,RecurrentNeural Networks, Bidirectional RNNs, Encoder-Decoder Sequence-to-Sequence Architectures,Deep Recurrent Networks,RecursiveNeural Networks,EchoStateNetworks,LSTM,Gated RNNs,OptimizationforLong- Term Dependencies, Auto encoders, Deep Generative Model

Introduction

Sequence modelling refers to learning fromdata where the **order of inputs matters**. In many real-world problems, the current output depends not only on the current input but also on **previousinputs**.Examplesincludenaturallanguagesentences,speechsignals,andtime-series data. To handle such data, **Recurrent Neural Networks (RNNs)** and **Recursive Neural Networks** are used.

1. RecurrentNeuralNetworks(RNNs)

Definition

ARecurrent NeuralNetworkisa neuralnetworkdesignedto process **sequentialdata** by maintaining a **hidden state** that stores information from previous time steps.

WorkingPrinciple

Ateachtimestep:

- The networktakesthecurrentinput
- Combinesitwiththeprevioushiddenstate
- Produces a new hidden state and

outputht=f(Wxt+Uht−1+b)h_t=f(Wx_t+Uh_{t-

1}+b)ht=f(Wxt+Uht−1+b)

Here,thesameweightsare **sharedacrossalltime steps**,enablingthe modeltorememberpast information.

Characteristics

- Hasfeedbackconnections

- Maintainsmemorythroughhidden state
- Handlesvariable-lengthsequences

Advantages

- Suitableforsequentialdata
- Parametersharingreducescomplexity

Limitations

- Suffersfromvanishingandexplodinggradients
- Pooratlearninglong-termdependencies

Applications

- Language modeling
- Speechrecognition
- Time-seriesprediction

2. RecursiveNeuralNetworks

Definition

RecursiveNeuralNetworksareneuralnetworksthat operateon **hierarchicalortree-structured data** instead of linear sequences.

WorkingPrinciple

- Thesameneuralnetworkisapplied**recursively**to smallerpartsofastructure
- Parentnoderepresentationsarecomputedfromchildnode representations
- Weightsharingoccursacrossalltreenodes

Characteristics

- Worksontreestructures
- Captureshierarchicalrelationships
- Usesrecursive composition

Advantages

- Effective forstructureddata
- Modelssyntacticandsemanticrelationships

Limitations

- Requirespredefined treestructure
- Computationallycomplex

Applications

- Sentenceparsing
- Sentiment analysis
- Naturallanguage understanding

ComparisonBetweenRecurrentandRecursiveNeural Networks

Aspect	RecurrentNN	RecursiveNN
DataType	Sequential	Tree-structured
Processing	Time-based	Hierarchical
Memory	Hiddenstate	Nodecomposition
Applications	Speech,text	Parsing,sentiment

UnfoldingComputationalGraphs

Introduction

Unfoldingcomputationalgraphsisatechniqueusedin**RecurrentNeuralNetworks(RNNs)** to represent their behavior over time. Since RNNs process sequentialdata using the same network repeatedly, unfolding helps visualize and understand how the network operates across multiple time steps.

ConceptofUnfolding

In an RNN, the same set of weights is applied at everytime step. Unfolding converts this cyclic structureinto a**linearchain ofidenticalnetworkcopies**, onefor eachtimestepinthesequence.

Eachcopyrepresentsthenetworkataparticulartimestep,whilesharing thesameparameters.

WorkingPrinciple

- Theinputsequenceis fedstepbystepintotheRNN
- Eachtimestep producesahidden state
- Whenunfolded,theRNNbecomesadeep feed-forwardnetworkThedepthofthenetwork equals the length of the input sequence

Thisunfolded representationallowsthenetworktotrack howinformationflowsthroughtime.

MathematicalRepresentation

Foreachtimestep*t*:

ht=f(Wxt+Uht−1+b)h_t=f(Wx_t+Uh_{t-1}+b)ht=f(Wxt+Uht−1+b) Here:

- xtx_txt =inputattime*t*
- hth_tht=hiddenstate
- W,U,bW, U, bW,U,b=sharedparameters

RoleinTraining

UnfoldingisessentialfortrainingRNNsusing**BackpropagationThroughTime(BPTT)**.

BackpropagationThroughTime

- Erroriscomputedateachtimestep
- Gradientsarepropagatedbackwardthroughtheunfoldednetwork
- Weightsareupdatedbased onaccumulatedgradients

AdvantagesofUnfolding

- MakesRNNcomputationunderstandable
- Enablesgradientcalculation
- Helpsanalyzetemporaldependencies

Limitations

- Longsequencesincreasecomputationalcost
- Causesvanishingandexplodinggradientproblems
- Requireslargememoryforlongsequences

Application

- Language modeling
- Speechrecognition
- Time-seriesprediction

RecurrentNeuralNetworks(RNNs)

Introduction

Recurrent NeuralNetworks(RNNs)areaclassofneuralnetworksspeciallydesignedto process **sequential and time-dependent data**. Unlike feed-forward neural networks, RNNs have **recurrent connections**that allow informationtopersist acrosstimesteps,enablingthenetwork to capture temporal dependencies.

Definition

ARecurrent NeuralNetworkisa neuralnetworkinwhichtheoutputfromtheprevioustimestep is fed back as input to the network, allowing it to maintain a **hidden state (memory)**.

Architecture

AnRNNconsistsof:

- Inputlayer
- Hiddenlayerwithrecurrentconnections
- Outputlayer

Thehiddenlayerstoresinformationfrompastinputsandpassesitforwardintime.

WorkingPrinciple

Ateachtimestep*t*:

- Thecurrentinputxtx_txtandpreviousinhiddenstateht−1h_{t-1}ht−1are combined
- Anewhiddenstatehth_thtisproduced
- Outputisgenerated based onhth_tht

ht=f(Wxt+Uht−1+b)h_t=f(Wx_t+Uh_{t-1}+b)ht=f(Wxt+Uht−1+b) Here:

- W,UW,UW,Uareweightmatrices
- bbbisbias
- fffisanactivationfunction

Thesameweightsare **sharedacrossalltime steps**.

UnfoldinginTime

Whenunfolded,anRNNbecomesa **deepfeed-forwardnetwork**withone layerpertimestep. This representation is used during training.

TrainingofRNNs

RNNsaretrainedusing**BackpropagationThroughTime(BPTT)**,where:

- Erroriscalculatedateachtimestep
- Gradientsarepropagatedbackwardthroughtheunfoldednetwork
- Weightsareupdatedaccordingly

Advantages

- Suitableforsequentialdata
- Handlesvariable-length inputs
- Parametersharingreducesmodelsize

Limitations

- Vanishingandexplodinggradientproblems
- Difficultyinlearninglong-termdependencies
- Trainingiscomputationallyexpensive

Applications

- Language modelingandtextgeneration
- Speechrecognition
- Time-seriesforecasting
- Sentiment analysis

BidirectionalRecurrentNeuralNetworks

Introduction

Bidirectional Recurrent Neural Networks (Bidirectional RNNs) are an extension of standard RNNs that improve sequence modelling byprocessing data in **both forward and backward directions**.Inmanysequenceproblems,thepredictionat aparticulartimestepdependsnotonly on past inputs but also on **future context**. BidirectionalRNNs are designed to capture this complete context.

Definition

A BidirectionalRNN is a recurrent neural networkthat consists of **two separate RNN layers**: onethatprocessesthesequence fromleft toright (forwarddirection)andanotherthatprocesses it from right to left (backward direction).

Architecture

ThearchitectureofaBidirectionalRNNincludes:

- **ForwardRNN**: processesinputsequencefromt=1t =1t=1tot=Tt =Tt=T
- **BackwardRNN**: processesinputsequencefromt=Tt =Tt=Ttot=1t =1t=1
- **Outputlayer**:combinesoutputs frombothdirections

Thehiddenstatesfrombothdirectionsareconcatenated orsummed to producethefinaloutput.

WorkingPrinciple

Ateachtimestep*t*:

- Forwardhiddenstatecapturesinformationfrompastinputs
- Backward hiddenstatecapturesinformation fromfutureinputs
- Combined hidden state provides richer contextual representation

ht=[ht→,ht←]h_t=[\overrightarrow{h_t},\overleftarrow{h_t}]ht=[ht,ht]

TrainingofBidirectionalRNNs

Bidirectional RNNs are trained using **Backpropagation Through Time (BPTT)** in both forwardandbackwarddirections.ThesameprinciplesofRNNtrainingapply,but withincreased computational cost.

Advantages

- Capturesbothpastandfuture context
- Improvespredictionaccuracy
- Effectiveforcomplexsequencetasks

Limitations

- Cannotbeusedforreal-timeoronlineprediction
- Requirescompleteinputsequence
- Highercomputationalandmemorycost

Applications

- Speechrecognition
- Handwritingrecognition
- Named entityrecognition
- Part-of-speechtagging

Encoder–DecoderSequence-to-SequenceArchitectures

Introduction

Encoder–DecoderSequence-to-Sequence(Seq2Seq)architecturesaredesignedtomapan**input sequence to an output sequence**, even when the input and output lengths are different. These models are widely used in tasks such as machine translation, speech recognition, and text summarization, where one sequence needs to be transformed into another.

Definition

ASeq2Seqmodelconsistsoftwomain components:

- **Encoder**:convertstheinputsequenceintoafixed-lengthrepresentation
- **Decoder**:generatestheoutput sequencefromthis representation

Bothencoder anddecoder areusuallyimplemented using**RNNs,LSTMs,orGRUs**.

Architecture

1. **Encoder**
   - Readstheinputsequenceoneelementatatime
   - Updatesitshiddenstateateachtimestep
   - Finalhiddenstaterepresentstheentireinputsequence(calledcontextvector)
2. **Decoder**
   - Takesthe context vectorasinput
   - Generatestheoutputsequencestep bystep
   - Uses itspreviousoutputasinputforthenextstep

WorkingPrinciple

- Theencoderprocessestheinputsequenceandproducesa**context vector**
- Thedecoderusesthiscontextvectortopredicttheoutput sequence
- At eachdecodingstep,thedecoderpredictsoneoutputtoken

Mathematically:

ht=f(Wxt+Uht−1)h_t =f(Wx_t+Uh_{t-1})ht=f(Wxt+Uht−1)

TrainingofSeq2SeqModels

- Trainedusing**BackpropagationThroughTime(BPTT)**
- Uses**teacherforcing**,wheretheactualoutputisgivenasinputduring training
- Loss iscomputedover theentireoutputsequence

Advantages

- Handlesvariable-lengthinputandoutputsequences
- Flexibleandpowerful
- Suitableformanysequencetransformationtasks

Limitations

- Fixed-lengthcontextvectorcancauseinformationloss
- Performancedegradesforlongsequences
- Computationallyexpensive

AttentionMechanism(Improvement)

Toovercomethe fixed-lengthcontext limitation,the**attentionmechanism**allowsthedecoderto focus on relevant parts of the input sequence at each time step, improving performance significantly.

Applications

- Machinetranslation
- Textsummarization
- Chatbots
- Speech-to-text system

Conclusion

Encoder–Decoder Seq2Seq architectures provide a powerful framework for sequence transformationproblems.Withimprovementslikeattentionmechanisms,theyformthe foundation of modern natural language processing systems.

DeepRecurrentNetworks

Introduction

Deep Recurrent Networks are anextensionofstandard Recurrent NeuralNetworks inwhich **multiplerecurrentlayersarestackedontopofeachother**. Whileasingle-layerRNNcan model simple temporal dependencies, deep RNNs are capable of learning **complex and hierarchical temporal patterns** present in sequential data.

Definition

ADeepRecurrent Networkisarecurrent neuralnetworkthat contains **morethanonehidden recurrent layer**, where the output ofone recurrent layer is fed as input to the next recurrent layer.

Architecture

- Consistsof**inputlayer**,multiple**recurrenthiddenlayers**, andan**outputlayer**
- Eachrecurrentlayermaintainsitsownhidden state
- Thehiddenstateofa lowerlayerattime *t* ispassedtothenext layeratthesametimestep This

stacking increases the depth of the network **in both time and space**.

WorkingPrinciple

Ateachtimestep:

1. Inputisprocessedbythe firstrecurrentlayer
2. Theoutputofthis layerispassedtothenextrecurrentlayer
3. Thisprocesscontinuesthroughalllayers
4. Finallayerproducestheoutput

Mathematically:

ht(l)=f(W(l)ht(l−1)+U(l)ht−1(l))h_t^{(l)}=f(W^{(l)}h_t^{(l-1)}+U^{(l)}h_{t-1}^{(l)})ht(l) =f(W(l)ht(l−1)+U(l)ht−1(l))

Wherellldenotesthelayer number.

TrainingofDeepRNNs

- Trainedusing**BackpropagationThroughTime(BPTT)**
- Gradientsarepropagatedthrough**multiplelayersand timesteps**
- OftencombinedwithLSTMorGRUunitstoimprovelearningstability

Advantages

- Learnshigh-leveland abstracttemporalfeatures

- Betterperformanceoncomplexsequencetasks
- Higherrepresentationalpower

Limitations

- Computationallyexpensive
- Difficulttotrainduetovanishinggradients
- Requireslargedatasets

Applications

- Speechrecognition
- Language modeling
- Machinetranslation
- Audiosignalprocessing

RecursiveNeuralNetworks

Introduction

Recursive Neural Networks are a class of neural networks designed to process **hierarchical or structureddata**ratherthanlinearsequences.UnlikeRecurrent NeuralNetworks,whichoperate over time steps, Recursive NeuralNetworks work on **tree-structured representations**, making themsuitable formodelingdatawithinherent hierarchicalrelationshipssuchas naturallanguage syntax trees.

Definition

A Recursive Neural Network is a neural network that applies the **same set of weights recursively**overastructuredinput,typicallyrepresentedasatree,tocomputerepresentations for parent nodes from their child nodes.

Architecture

- Inputdataisrepresentedasa**tree structure**
- Leafnodesrepresentbasicinputunits(wordsor features)
- Internalnodesrepresentcompositionsofchildnodes
- Thesameneuralnetworkisused ateverynodeinthetree

Thisweightsharingenablesthemodeltogeneralizeacrossdifferentstructures.

WorkingPrinciple

- Thenetwork startscomputationfromthe**leafnodes**
- Child noderepresentationsarecombinedusinganeuralfunction
- Thisprocesscontinuesrecursivelyuntiltherootnodeisformed
- Therootrepresentationcapturesthemeaningoftheentirestructure

Mathematically:

hparent=f(W[hchild1,hchild2]+b)h_{parent}=f(W[h_{child1},h_{child2}]+b)hparent =f(W[hchild1,hchild2]+b)

TrainingofRecursiveNeuralNetworks

- Trainedusing**BackpropagationThroughStructure(BPTS)**
- Errorsarepropagated fromtherootnodetoleafnodes
- Requiresknowntreestructuresduring training

Advantages

- Effectivelymodelshierarchicalrelationships
- Suitableforstructuredandcompositionaldata
- Capturessyntacticand semanticinformation

Limitations

- Requirespredefined treestructures
- Computationallyexpensive
- Difficult toscaletolargedatasets

Applications

- Sentenceparsing
- Sentiment analysis
- Naturallanguage understanding
- Semanticrelationship modeling

EchoStateNetworks

Introduction

Echo StateNetworks(ESNs)areaspecialtypeof**RecurrentNeuralNetwork(RNN)** designed to simplify training while still capturing temporal dependencies in sequential data. Unlike traditionalRNNs,ESNsusealarge, fixedrecurrentlayercalleda **reservoir**,whicheliminates the need to train recurrent connections.

Definition

AnEcho StateNetworkisarecurrent neuralnetworkinwhichthe **recurrentweightsare randomly initialized and kept fixed**, and **only the output layer weights are trained**.

Architecture

AnESNconsistsofthreemaincomponents:

1. **InputLayer**–feedsinputsignalsintothereservoir
2. **Reservoir**–alarge,sparselyconnectedrecurrentnetworkwithfixed weights
3. **OutputLayer**–trainedusingsupervisedlearning

Thereservoiractsasadynamic memorythat transformsthe input intoahigh-dimensional representation.

WorkingPrinciple

- Inputsignalsareprojectedintothe reservoir
- Thereservoircreatesrich,dynamic internalstates
- Thesestatesarecombined linearlyattheoutputlayerto generatepredictions

State update equation:

x(t)=f(Winu(t)+Wx(t−1))x(t)=f(W_{in}u(t)+Wx(t-1))x(t)=f(Winu(t)+Wx(t−1)) Where:

- u(t)u(t)u(t)isinput
- x(t)x(t)x(t)isreservoirstate
- WinW_{in}Win,WWWarefixedweightmatrices

EchoStateProperty

The**echostateproperty** ensuresthat:

- Theinfluenceofinitialstatesfadesovertime
- Thereservoirstatedependsmainlyonrecentinputs

Thispropertyisachieved bycontrolling the**spectralradius** ofthereservoir weightmatrix.

TrainingofESNs

- Onlyoutputweightsaretrained
- Trainingisfast andefficient
- Typicallyuses linearregressionorridge regression

Advantages

- Veryfast training
- Avoidsvanishingandexplodinggradients
- Simpleandstablelearning

Limitations

- Performancedependsonreservoir design
- Notoptimal foralltasks
- Limitedadaptabilitysincereservoirisfixed

Applications

- Time-seriesprediction
- Speechrecognition
- Signalprocessing
- Controlsystems

Long Short-TermMemory(LSTM)

Introduction

LongShort-TermMemory(LSTM)isaspecialtypeof**RecurrentNeuralNetwork(RNN)** developed to overcome the **long-term dependency problem** faced by traditional RNNs. StandardRNNsstruggleto learninformationoverlongsequencesduetothe **vanishinggradient problem**. LSTM addresses this issue by introducing a memorycell and gating mechanisms that control information flow.

Definition

AnLSTM isarecurrent neuralnetworkarchitecturethatusesa **memorycellandthreegates** to selectively remember, update, and output information over long periods of time.

Architecture

AnLSTMunitconsistsof:

1. **CellState (CtC_tCt)**–storeslong-terminformation
2. **ForgetGate(ftf_tft)** –decideswhatinformationtodiscard
3. **InputGate(iti_tit)** –decideswhatnewinformationtostore
4. **OutputGate(oto_tot)**–decideswhat informationtooutput

These gates use **sigmoid** and **tanh** activation functions.

WorkingPrinciple

Ateachtimesteptttt:

ForgetGate

ft=σ(Wf[ht−1,xt]+bf)f_t=\sigma(W_f[h_{t-1},x_t]+ b_f)ft=σ(Wf[ht−1,xt]+bf)

InputGateandCandidateMemory

it=σ(Wi[ht−1,xt]+bi)i_t = \sigma(W_i [h_{t-1}, x_t] + b_i)it=σ(Wi[ht−1,xt]+bi)
C~t=tanh⁡(Wc[ht−1,xt]+bc)\tilde{C}_t=\tanh(W_c[h_{t-1},x_t]+b_c)C~t=tanh(Wc[ht−1,xt
]+bc)

CellStateUpdate

Ct=ft·Ct−1+it·C~tC_t =f_t\cdotC_{t-1}+i_t\cdot \tilde{C}_tCt=ft·Ct−1+it·C~t

OutputGate

ot=σ(Wo[ht−1,xt]+bo)o_t=\sigma(W_o[h_{t-1},x_t]+b_o)ot=σ(Wo[ht−1,xt]+bo)
ht=ot·tanh⁡(Ct)h_t = o_t \cdot \tanh(C_t)ht=ot·tanh(Ct)

TrainingofLSTM

- Trainedusing**BackpropagationThroughTime(BPTT)**
- Gateshelp maintainstablegradients
- Learns long-termdependencieseffectively

Advantages

- Solvesvanishinggradientproblem
- Captureslong-termdependencies
- Stableandpowerfulsequencelearning

Limitations

- Complexarchitecture
- Highcomputationalcost
- Requiresmorememory

Applications

- Speechrecognition
- Machinetranslation
- Textgeneration
- Time-seriesforecasting

GatedRecurrentNeuralNetworks(GRUs)

Introduction

Gated Recurrent Neural Networks are an improved form of **Recurrent Neural Networks (RNNs)**designedtoovercomethe**vanishinggradientproblem**andtobettercapture**long-term dependencies** in sequentialdata. Byusing gating mechanisms, Gated RNNs controlthe flow of information through the network.

Definition

AGatedRecurrent NeuralNetworkisarecurrentneuralnetworkthatuses **gates**to regulatehow much past information is retained and how much new information is added at each time step. ThemostcommongatedRNNisthe**GatedRecurrentUnit(GRU)**.

Architecture

AGRU consistsoftwomaingates:

1. **UpdateGate(ztz_tzt)**– decideshowmuchpastinformationto keep
2. **Reset Gate (rtr_trt)** – decides how much past information to forget

UnlikeLSTM,GRUdoes**not**haveaseparatememorycell, makingit simpler.

WorkingPrinciple

Ateachtimesteptt:

UpdateGate

zt=σ(Wz[ht−1,xt]+bz)z_t=\sigma(W_z[h_{t-1},x_t]+b_z)zt=σ(Wz[ht−1,xt]+bz)

ResetGate

rt=σ(Wr[ht−1,xt]+br)r_t=\sigma(W_r [h_{t-1},x_t]+ b_r)rt=σ(Wr[ht−1,xt]+br)

CandidateHiddenState

h~t=tanh(Wh[rt·ht−1,xt]+bh)\tilde{h}_t = \tanh(W_h[r_t\cdoth_{t-1},x_t]+b_h)h~t
=tanh(Wh[rt·ht−1,xt]+bh)

FinalHiddenState

ht=(1−zt)·ht−1+zt·h~th_t=(1-z_t)\cdoth_{t-1}+z_t\cdot\tilde{h}_tht=(1−zt)·ht−1+zt·h~t

TrainingofGatedRNNs

- Trainedusing**BackpropagationThroughTime(BPTT)**
- Gatesallowgradientstoflow smoothly
- FasterconvergencethantraditionalRNNs

Advantages

- Solvesvanishinggradientproblem
- Fewer parametersthan LSTM
- Fastertrainingand lessmemoryusage

Limitations

- SlightlylessexpressivethanLSTM
- Performancedependson task

Applications

- Textandspeechprocessing
- Machinetranslation
- Time-seriesprediction

OptimizationforLong-TermDependencies

Introduction

Insequence modellingtasks, **long-termdependencies**occurwheninformationfromearliertime steps ina sequence significantly influences outputs at much later time steps. Standard Recurrent Neural Networks (RNNs) face difficulty in learning such dependencies due to optimization problemsduringtraining.Efficientoptimizationtechniquesarethereforerequiredtoensure stable learning over long sequences.

Long-TermDependencyProblem

WhentrainingRNNsusinggradient-based methods,gradientsarepropagatedbackwardthrough time. As the sequence length increases:

- Gradientsmay**shrinkexponentially**(vanishinggradientproblem)
- Gradientsmay**growuncontrollably**(explodinggradientproblem)

These issuespreventthenetworkfromlearningrelationshipsbetweendistanttime steps.

VanishingGradientProblem

- Gradientsbecomeverysmall
- Earlylayersortimestepsreceivealmostnoupdates
- Networkfailsto learnlong-termpatterns

ExplodingGradientProblem

- Gradientsgrowexcessivelylarge
- Causesunstablelearning
- Leadstonumericaloverflow

OptimizationTechniquesforLong-TermDependencies

1. GatedArchitectures(LSTMandGRU)

- LSTMuses**gatesandmemorycells**topreserveinformation
- GRUuses**updateandresetgates**
- Thesearchitecturesallowgradientstoflowsmoothlyoverlongtimesteps

2. GradientClipping

- Gradientsareclippedtoafixedrange
- Preventsexplodinggradients
- Ensuresstabletraining

3. ProperWeightInitializatio

- Initializesweightscarefullytoavoidextremevalues
- Helpsmaintainstablegradients
- Improvesconvergence

4. TruncatedBackpropagationThroughTime(TBPTT)

- Backpropagationislimitedtoafixednumberoftime steps
- Reducescomputationalcost
- Controlsgradientinstability

5. UseofBetterActivationFunctions

- ReLUandgatedactivationsreducegradientdecay
- Sigmoidandtanhaloneworsenvanishing gradients

6. RegularizationandNormalization

- Techniqueslikedropoutand layernormalization
- Improvegeneralizationand stability

AdvantagesofOptimizationTechniques

- Enableslearningoflong-rangedependencies
- Improvesmodelstability
- Enhancesconvergence speed

Applications

- Speechrecognition
- Machinetranslation
- Language modeling
- Time-seriesforecasting

Autoencoders

Introduction

Autoencoders are atype of **artificial neural network** used for **unsupervised learning**, where theobjective isto learnanefficient representationofinput data.Theyworkbycompressingthe input into a lower-dimensional representation and then reconstructing it back to the original form. Autoencoders are widely used for feature learning and dimensionality reduction.

Definition

Anautoencoderisaneuralnetworkthatistrainedto**reproduceitsinputattheoutput layer** by passing it through a compressed hidden representation.

Architecture

Anautoencoderconsistsofthreemainparts:

1. **Encoder**–compressestheinputdataintoalatentrepresentation
2. **LatentSpace(BottleneckLayer)**–holdscompressedfeatures
3. **Decoder**–reconstructstheoriginalinput fromthelatent representation

The encoder and decoder are usually symmetric.

WorkingPrinciple

- Inputdataisfedintotheencoder
- Encoder reducesdimensionality
- Decoderattemptstoreconstructtheoriginalinput
- Thenetworkminimizesreconstructionerror

Mathematically:

z=f(Wx+b)z=f(Wx+b)z=f(Wx+b) x^=g(W′z+b′)\hat{x}=g(W'z+b')x^=g(W′z+b′)

Where:

- xxxisinput
- zzzislatentrepresentation
- x^\hat{x}x^ isreconstructedoutput

TrainingofAutoencoders

- Trainedusing**backpropagation**
- Useslossfunctionslike**MeanSquaredError(MSE)**
- Doesnotrequirelabeleddata

TypesofAutoencoders

1. UndercompleteAutoencoder

- Latentspacesmallerthaninput
- Forcesfeaturelearning

2. SparseAutoencoder

- Usessparsityconstraints
- Learnsmeaningfulrepresentations

3. DenoisingAutoencoder

- Trainedtoremovenoisefromdata

4. VariationalAutoencoder(VAE)

- Probabilisticgenerativemodel
- Usedindata generation

Advantages

- Learnsfeatures automatically
- Reducesdimensionality
- Workswithoutlabeled data

Limitations

- Maylearntrivialidentitymapping
- Requirescarefultuning
- Notidealforsupervisedtasks

Applications

- Dimensionalityreduction
- Noiseremoval
- Featureextraction
- Anomalydetection

DeepGenerativeModels

Introduction

Deep Generative Models are a class of deep learning models that learn the **underlying probability distribution of data**. Unlike discriminative models that only make predictions, generative modelscan**generatenewdatasamples**thataresimilartothetrainingdata.These models are widely used in image, audio, and text generation tasks.

Definition

ADeepGenerative Modelisa neuralnetworkthatlearnstomodelthe**data-generatingprocess** andcangeneratenewdatasamplesbysamplingfromthelearned distribution.

BasicIdea

- Learnhowdatais distributed
- Capturehiddenpatternsand structure
- Generaterealisticnewsamples

Thesemodelsareusuallytrainedusing**unsupervisedorsemi-supervisedlearning**.

TypesofDeepGenerativeModels

1. VariationalAutoencoders(VAE)

- Encodermapsinputtoaprobabilitydistribution
- Decodergeneratesdatafromsampledlatentvariables
- Usesprobabilisticlearning

Advantage: Stabletraining
Limitation: Slightlyblurryoutputs

2. GenerativeAdversarialNetworks(GANs)

- Consistsoftwonetworks:**Generator**and**Discriminator**
- Generatorcreatesfakedata
- Discriminatordistinguishesrealvs fakedata
- Trainedusingadversariallearning

Advantage: High-qualitydatageneration
Limitation: Traininginstability

3. AutoregressiveModels

- Generatedataoneelementatatime
- Eachoutputdependsonprevious outputs

Examples: PixelRNN,WaveNet

TrainingofDeepGenerativeModels

- Usesgradient-basedoptimization
- Minimizeslikelihoodoradversarialloss
- Requires largedatasetsandhighcomputation

Advantages

- Cangeneraterealisticdata
- Useful fordataaugmentation

- Learns complexdatadistributions

Limitation

- Trainingcanbe unstable
- Computationallyexpensive
- Difficulttoevaluateperformance

Applications

- Imageandvideogeneration
- Speechsynthesis
- Dataaugmentation
- Anomalydetection