26/05/23

DAY 5:

I. Machine Learning

Applications

(sound/speech recognition)

↳ t vs voltage wave.

(emotions, accent, speed detected)

(FEATURE EXTRACTION)

of an AI system:

written problem vs expected outcome

↓

this accuracy is very high.

naturally → no idea how its done.

$y = f(x)$

learn f

code f

Classification → int y

clustering → unsupervised.

Regression → real no. y

Time series prediction → e.g. predict future weather based on past 5 days

* (formulate real world problem) *

training data → feature extraction → feature space representation

↓ select relevant info

↓ classifier design

(line)

Class label ← classifier model ← classifier design

test data → feature set → fsr → classifier model

(line eqn)

not correctness, the accuracy

IMP.

we decide an eqn that divides | fit a model for different classes.

generative | discriminative.

1) Classifier — k-NN (find k nearest neighbours
↳ most common class is the answer) (majority) most frequent label.

image → feature vector

for multiple classes → voronoi tessellation. (between nearest neighbours)

piecewise linear boundary → final product.
(cause ⊥ bisector)

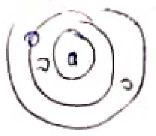
sklearn.neighbours.

Training is trivial. (no learning effectively).

★ Bayesian classifier known to be most ideal (theoritical proof)

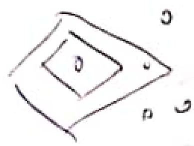
what is near →

L2 1) Euclidean Distance → $\sum(|x_1 - x_2|)^2$
Iso surfaces →



} draw circles.

L1 2) Manhattan distance (no sq.).
$\sum |x_1 - x_2|$
iso surface



} draw rhombus.

L0 distance?

3) Minkowski Dist
$\sum |x_1 - x_2|^r$

r = 2 = ○ r = ∞ = □
r = 1 = ◇
r = 1.25 =

If features have different variances

↳ some feature will dominate

↳ so req. to nominate.

normalization.

4) Mahalnobis Dist.

$d(P,Q) = \underbrace{(P-Q)^T S^{-1} (P-Q)}_{\text{weighted distance}}$ → for accounting for features that change less|more.

5) Hamming distance.

(binary vectors)

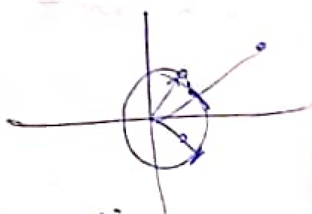
$d(P,Q) = \underbrace{\Sigma I}_{\text{Indicator funct.}} (p_i \neq q_i)$

↳ Indicator funct. (T = 1, F = 0)

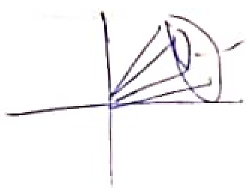
6) cosine distance

$d(P,Q) = \cos\theta = \dfrac{\vec{P}\cdot\vec{Q}}{|\vec{P}|\,|\vec{Q}|} = \dfrac{P\cdot Q}{\|P\|\cdot\|Q\|}$

(hypersphere – 4D+)



normalize.



feature vectors need not be of same length.

non metric, edit, jaccard dist.

(2) linear classifier

linear decision boundaries.

pt, line, plane, hyperplane.

$w_0 + w_1 x_1 + w_2 x_2 \; - \; - \; w_d x_d : 0$

(d dimension-space line).

$f(x) = 0$.

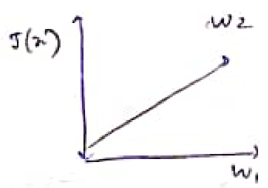
goal → learn parameters w_i so training samples are separated.

linear boundry → better to compute.

understanding is simplifying.

Minimizing loss func.

$J(w)$



point in parameter space
⇒ line in feature space.

iteratively modify w_i to reduce $J(w)$.

what modifications reduce $J(w)$?

$\frac{\partial J}{\partial w_i}$ ↑ slope } ~~gradient descent~~ slope

$\frac{\partial J}{\partial w}$ } gradient. (ascent) (vector)
↓
vector

$-\frac{\partial J}{\partial w}$ } gradient descent.

$w = \frac{\partial J}{\partial w}$.

gd algorithm. $w^{t+1} = w^t - \eta \frac{\partial J}{\partial w}$

Loss function of (1)

↳ no. of misclassified samples.
(not differentiable).

↳ $y_i(w^T x_i) > 0$ for all samples)

$J(w) = -y_i(w^T x_i)$ (for misclassified samples)

$\nabla J = -y_i x_i$

↳ called perceptron update rule.

so $w^{t+1} = w^t + \eta \sum_{x_i \in X} y_i x_i$ ↱ already negative.

1	0
0	1

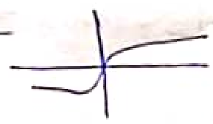
} linear classifier doesn't work, no way to separate.

x_1 w_1
x_2 w_2
⋮ ⋮

$\sum_{i=1} (w_i x_i) + b$ (bias) if $> T$ (threshold) } $\Phi(\cdot) = y$ (output)

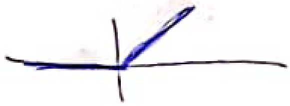
↳ activation function.

hyperbolic tangent tanh

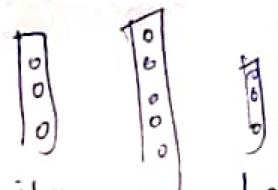


} most popular.

ReLU



determine $\underline{w, b}$



i/p layer

hidden (neuron) layer
hidden layer.

o/p layer

neural networks help in creating cloud boundaries (highly complex).

determining w/b is very very difficult.

Only 1 hidden layer suffices.

>2 hidden layers => vanishing gradient descent. (back propagation).

hyperparameter = no. neurons
↓
layers.

MLP