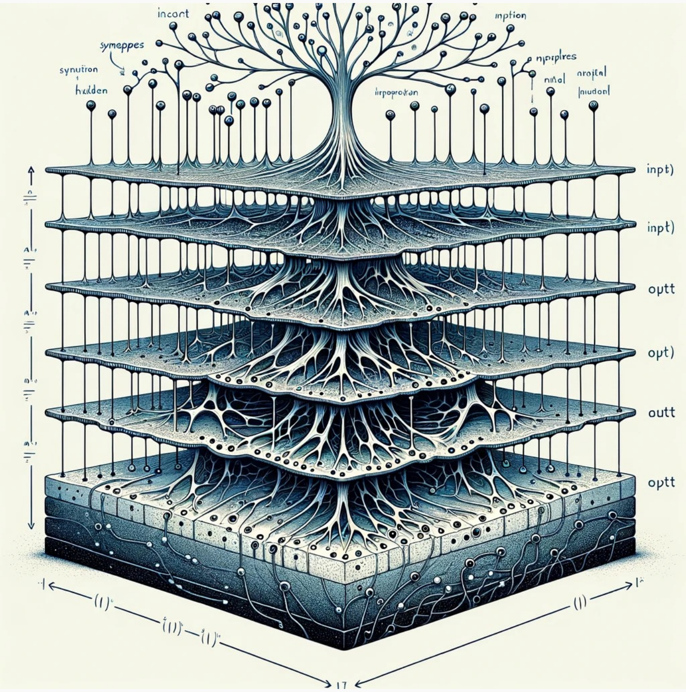
kokchun giang

multilayered perceptron (MLP) for image classification



strategy for building a an MLP

Occam's razor

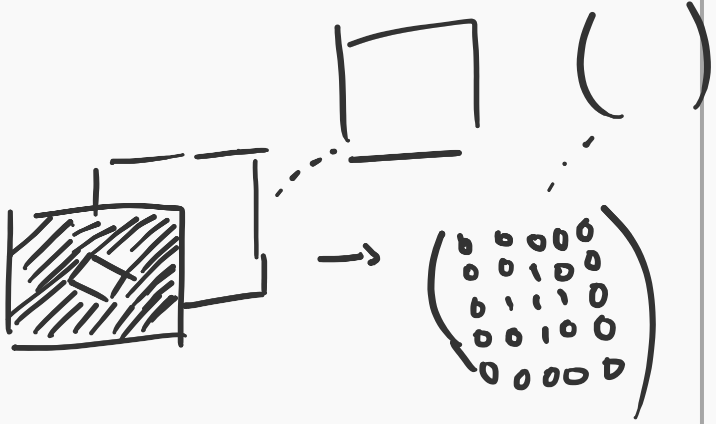
For MLP: find least amount of neurons & hidden layers to generalize well

Strategies

1. Growing - build from scratch & testing
2. Pruning - large network then discard neurons
3. Global search e.g. genetic algorithm
4. Regularization - punish weights
5. Early stopping
6. Use proven architectures for similar tasks
7. Fine tune a pretrained model (transfer learning)

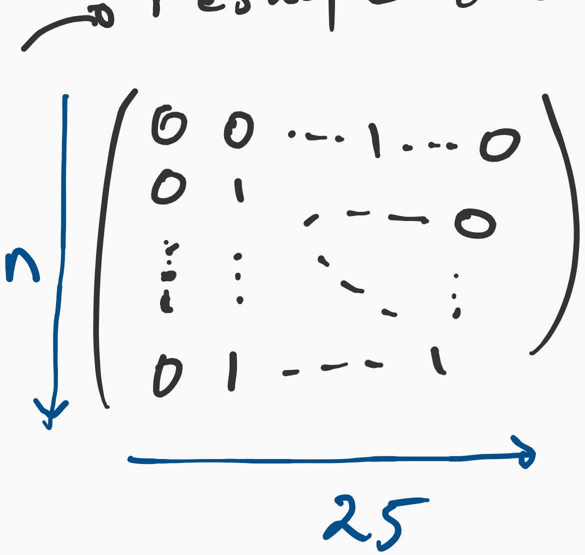
image classification with MLP

Ex classify rhombus, triangle, circle



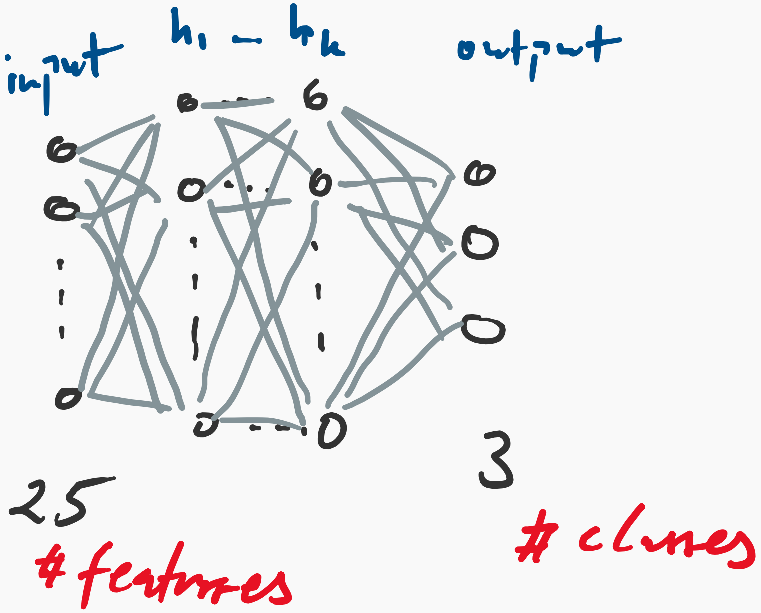
n gray scale images

reshape the matrices



1 column (feature) is an image

Flattened



questions to consider

1. How many hidden layers?
2. How many neurons?
3. Activation fcn ?
4. Loss fcn?

---

1 & 2 → trial & error
check research on similar problems

→ dropout, regularization
early stopping ...

3. Hidden layers

use ReLU + He initialization
or leaky ReLU + —"—
or swish + Glorot init..

Output layers

→ linear (regression)

↓ Softmax (multi-class classification)

↘ sigmoid (binary classification)

généralisation of logistic fcn to multidimensions

→ sum of outputs = 1
=> probability for each output neuron

questions to consider

softmax ex

0 - rhombus ◇

1 - circle ○

2 - triangle △

output

$$y = \begin{pmatrix} 0,1 \\ 0,2 \\ 0,7 \end{pmatrix}$$

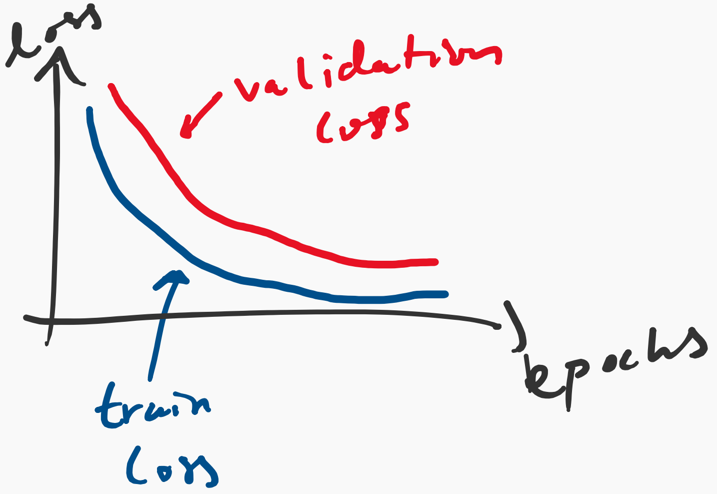
$$\underset{i}{\arg\max}\, y = 2$$

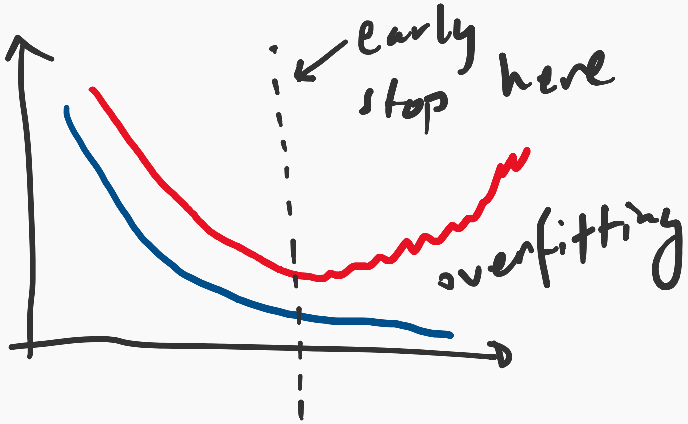
classify as triangle

Loss fn

- MSE (regression)
- Binary cross entropy (binary class)
- Cross entropy (multiclass one hot encoded)
- Sparse cross-entropy (multiclass)

Loss curves





decrease overfitting

→ regularization l_1, l_2 on loss fn

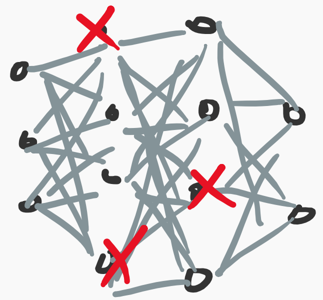
→ early stopping
stop before validat loss increases

→ data augmentation
rotate, scale, color changes ...

→ synthetic data
simulate artificial data e.g. 3D model on 2D backgrounds

→ reduce model complexity
simpler model

→ dropout
randomly disable neurons



doesn't fit as hard to train data