Final Exams Sol

2018

1) Define the ML Recipe:

i) Train the network and evaluate first on the training data
→ if bias is high (underfitting) (bad Perf. on train)
  → use bigger Network [more hidden layers/nodes]
  → Train longer
→ Check the bias again and keep changing until U reach good Bias

ii) Check for variance (validation set Performance)
→ if variance is high (overfitting) (bad Perf in valid.)
  → use more data (if Possible)
  → use Regularization
→ Check for Bias, then variance, keep changing until → good Bias, good Var

iii) search for better NN architecture, that better suits the Problem!

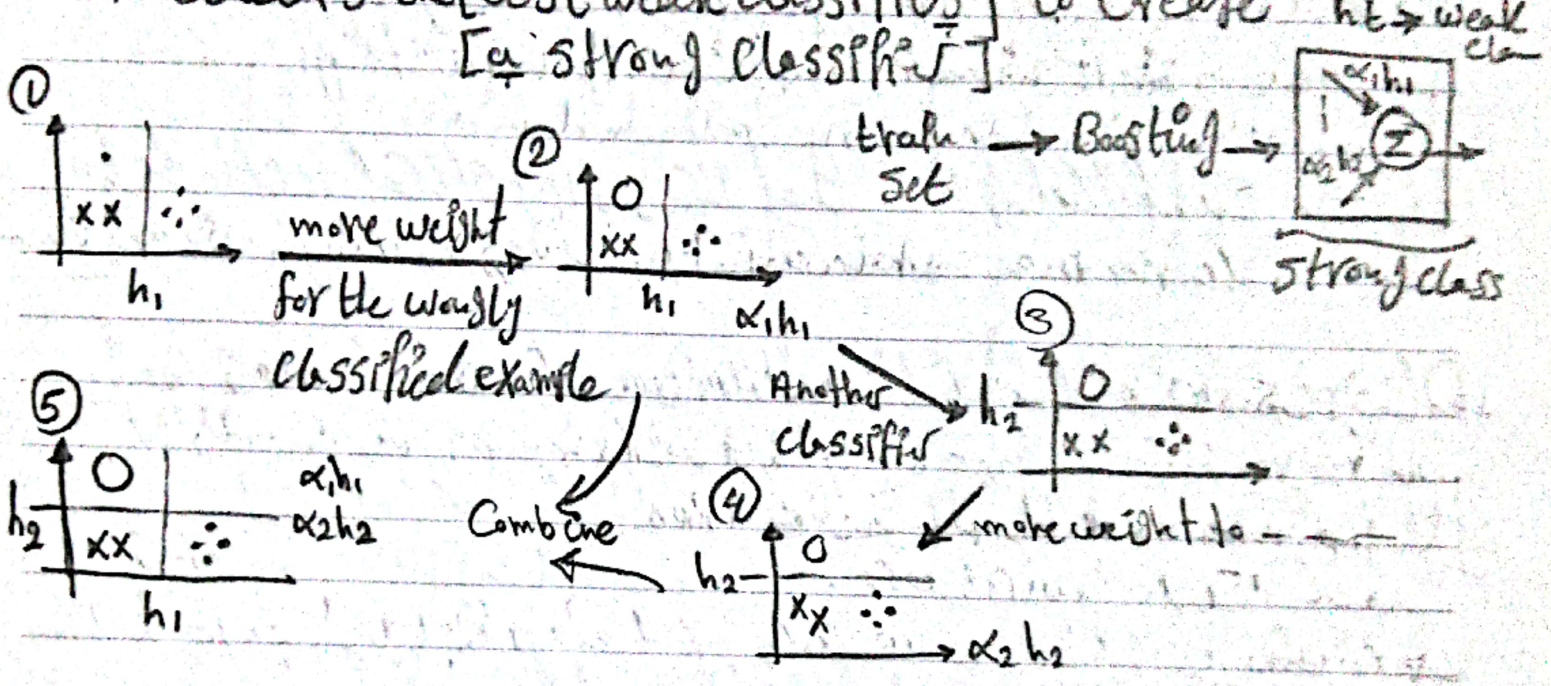
4) Diff between wrapper & filter method for feature extraction?

wrapper → Select features by taking the classifier into consideration
  → e.g SFS, SBS
  → Adv: High Accuracy
  → Dis: Slow Execution, lack of Generality

Filter → Select features regardless of the classifier, based on [Corr. bet features, Distance bet. class means]
  → e.g PCA
  → Adv: Fast exe, General
  → Dis: Tendency to select large subsets

2) Key Idea of Adaboost, using diagrams?

* it tries to approx the Bayes classifier by Combing many weak classifiers.
* it selects the [best weak classifiers] to create [a strong classifier]



3) why can't adaboost work for more than binary classific?

* Bec it assumes that the error of each weak classifier is less than (0.5). However, in case of multi-class, the error is $\frac{K-1}{K} = 1 - 1/K$. It may be more than 0.5 in some cases, resulting in -ve weights (α_t)

* Modification? - we added the [log(K-1)] term, to make it +ve in case of $K > 2$.

- $\alpha_t = \log\left(\frac{1-err_t}{err_t}\right) + \log(K-1)$
- α_t is +ve, iff: $(1-err_t) > 1/K$

5] Compare bet. AI, ML, DL!

AI ⇒ It's Categorized into 4 Categories
↳ Thinking humanly, Thinking rationally
↳ Acting humanly, Acting rationally
*most scientis consider it as acting rationally

ML ⇒ Subfield of AI, that uses statistical methods to determine some outputs, corresponding to some in

DL ⇒ Subfield of ML, where we use multi-layered NN

6] U trained a NN, U got 54% train accuracy and 51% valid accuracy, what will U do next?

* Try using BCG's Network and/or train longer
* Check for the accuracy again and keep changing until reaching an acceptable accuracy.

Hence,
* if the validation accuracy, wasn't satisfactory try using more data and/or Regularization
* Check again for both accuracy and keep changing until reaching acceptable accuracies

Hence,
* If nothing improves, we may search for better NN architectures

—

[ML Recipe] 31

7] Explain with examples the linear perceptron update rule [Widrow-Hoff] ← Also the main Algo

i] Initialize the weights and Bias randomly

ii] Present the feature vector of the m^{th} train example $\underline{x}(m)$ and its disired output $d(m) = \begin{cases} 1 , & x(m) \in C_1 \\ 0 ; & x(m) \in C_2 \end{cases}$

iii] Calculate the actual output for $m \rightarrow y(m) = f(\underline{w}^T \underline{x}(m))$

iv] weights are adapted to the [Widrow-Hoff] Rule

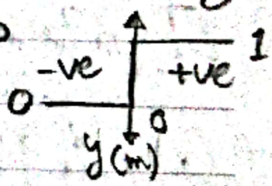
$$\underline{w}(new) = \underline{w}(old) + \alpha [d(m) - y(m)] \underline{x}(m)$$

(α ← learning rate)

v] Return to [ii] until, all patterns are classified correctly

---

example: let $d(m) = 1$, and $y(m) = f(\underline{w}^T \underline{x}(m)) = 0$



$\therefore \underline{w}^T \underline{x}(m) < 0$ (-ve) → but, we want $y(m) = 1$

So, we need to make $\underline{w}^T \underline{x}(m)$ more +ve.

Widrow-Hoff → $\underline{w}(new) = \underline{w}(old) + \alpha [d(m) - y(m)] \underline{x}(m)$

$$y(new) = f(\underline{w}^T(new) \cdot \underline{x}(m)) = f(\underbrace{\underline{w}^T_{old} \underline{x}(m)}_{-ve} + \alpha [d(m) - y_{old}(m)] * \underline{x}^T(m) \underline{x}(m))$$

$$= f(-ve + \underbrace{\alpha [1 - 0] \|\underline{x}(m)\|^2}_{+ve}) = f(+ve) = 1$$

and vice versa!

* كفاية تكتب ال example بس

انا كتبت ال Algo فوق زيادة

⑧ Explain the Naïve Estimator, compare it to Histogram Analysis!

* it's the same as histogram analysis, but instead of partitioning the feature space into pre-specified ranges, we take a range around each x. So the Naïve estimator is smoother than the histogram analysis.

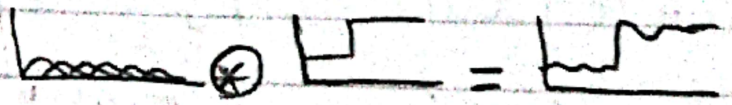
* Density estimation $\hat{P}(x) = \frac{\text{no of points in } [x-h/2, x+h/2]}{\text{total no of points} \leftarrow Mh}$

* DIS → $\hat{P}(x)$ is discontinuous
  ADV → All data points are treated equally, regardless of their distance to x (estimation point).

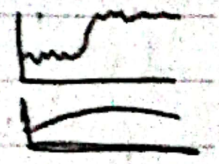
⑨ Kernel Density [Parzen Window] → used to get the density of data by using Bump func for each point, then it gets the estimation density through the average of their sum.
→ it's continues and gives weights for all the points, Hence, better than Naïve estimator.

$\phi_n(x) = \frac{1}{h} g\left(\frac{x}{h}\right)$ ← PDF

$\hat{P}(x) = \frac{1}{M} \sum_{1}^{M} \phi_n (x - x(m))$



* if Kernel width (h) → too small → too Bumpy
  → too large → lost Details



$h_{opt} = \frac{1}{N} \Sigma H_i$

$H_i = \sigma_i \left[\frac{4}{M(N+2)}\right]^{1/N+4}$

10] Why ReLU is used instead of Sigmoid?

* Because, ReLU results in a faster learning.
* Sigmoid has fast saturation
* ReLU → used with hidden layers / Sigmoid → output layer in binary classification

11] Explain the least sq. classifier [No Proof Needed]

* In some cases, if $\underbrace{M}_{\text{examples}} > \underbrace{N+1}_{\text{weights}}$ → we can't get exact equality to calc the weights!
  (M eq, $N+1$ unknowns)
* we will define an error func → $E = \sum_{m=1}^{M} (\underbrace{w^T u(m)}_{\text{actual}} - \underbrace{b_m}_{\text{desired}})^2$
* we want to find the weights that will lead to the least error i.e. → solve $\frac{\partial E}{\partial w} = 0$. Here, we can converge if the problem is not linearly separable

* $E = [Yw - b]^T [Yw - b]$
  $= w^T Y^T Y w - 2 b^T Y w + b^T b$ $\xrightarrow{\frac{\partial E}{\partial w}}$ $2 Y^T Y w - 2 Y^T b = 0$
  $[w = (Y^T Y)^{-1} Y^T b]$ ←

12] Main issues in GMM?

* Initialization
  - → GMM is an iterative algo, that's very sensitive to initial conditions
  - → we usually, use K-means to get a good initialization
* No. of Gaussian components
  - → try diff no. of comp., choose the best based on a validation set.

13] What are the types of feature removed in feature selection method?

→ Irrelevant features, that don't help in discriminating bet. classes eg

→ Correlated features, that are very close to each other, so, can be removed without much loss in information

14] Explain Regularization and why it's used?

* it's a technique used to prevent/reduce overfitting
it simplifies the network and makes it smaller.

L1 Regularization ⇒ $J = \frac{1}{M}\sum E_m + \frac{\lambda}{2M}\|\underline{W}\|_1$ → $\sum_j |W_j|$

L2 Regularization ⇒ $J = \frac{1}{M}\sum E_m + \frac{\lambda}{2M}\|\underline{W}\|_2^2$ → $\underline{W}^T\underline{W}$

Drobut Reg ——→ Eliminate some Nodes based on a Probability

16] How Bayes Rule can be used in classification?

→ we calculate $[P(x|C_i)P(C_i)]$ for each class and classify x for the class that gives the highest value

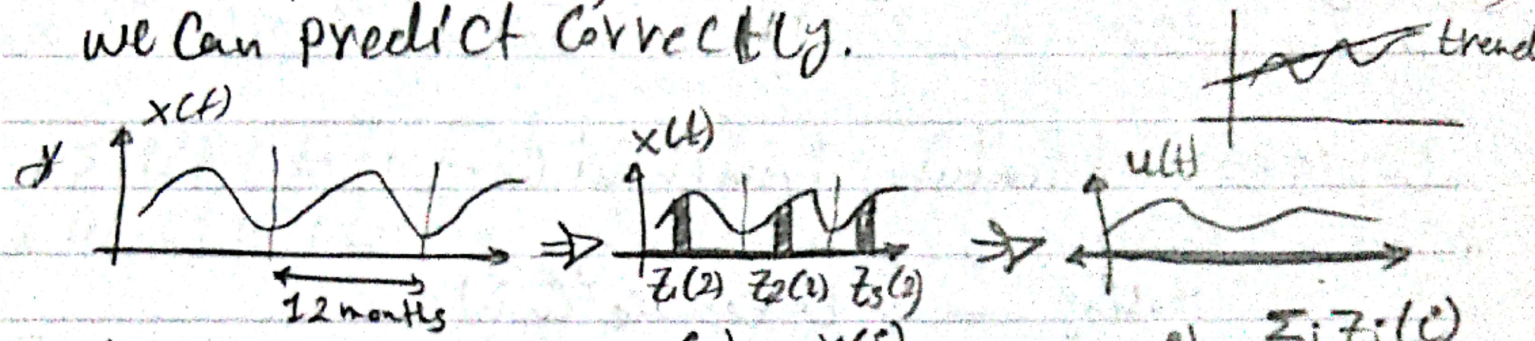
18] Disadv of Back Propagation?

→ Can be slow in reaching the Min. [adjust α]

→ Prone to get stuck in local Minimen (local → not global)

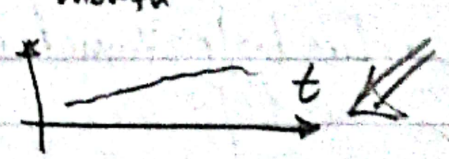
17] What's the Importance of deseasonalization?

* Used to Remove seasonal Periodicities. Here we can predict correctly.



$a(year) = \frac{1}{12} \sum_{\text{window}} x(t)$ $\quad z(i) = \frac{x(t)}{a(year)}$ (i: month) $\quad u(i) = \frac{\sum_j z_j(i)}{\text{no of years}}$

avg of z of diff years for the same month



$x_{deseason}(t) = \frac{x(t)}{u(month(t))}$

19] Three Methods to update weights?

* Gradient Descent → Use full set of training examples
  - Adv: Consistent optimization
  - Dis: Slow

* Mini-Batch → Subset of training examples
  - Adv: Fast

* Stochastic GD → one training example at a time.
  - Adv: Faster than GD
  - Dis: Hard to converge
    : loss speedup from vectorization

Final 2019

Same as 2018. but those were added

① Describe how CNN's work, why they have less Footprints?

* layers extract features from input images.
* it uses Conv layers → for filtering
  Pooling → to reduce inputs.
* it learns [filter parameters & weights of fully connected layers]

it has less memory Footprints → due to [Parameter sharing]
→ due to [Sparsity of Connections]

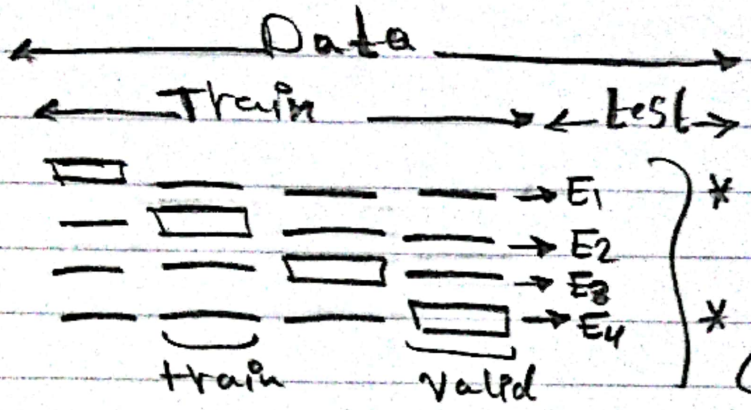
② 3-limitations of DL?

→ Needs large amount of labeled data
→ Datasets may be biased
→ Sensitive to std adversarial attacks
→ oversensetive to changes in context.
→ visual understanding is tricky
→ Unintended results from fitness functions
→ Not a magic tool [Not able to build general-intelligent machine]
→ cannot fit all real-world scenarios

3) Explain K-fold validation, in what context is it used?

* Used for Parameter tuning over the train data
* Better than Convention [train-valid-test split]

* Split into train & test (No need for validation)
* Apply K-fold to the train set (usually K=5)



* Train on (▭) then, validate on (▭).
* Eval = E1 + E2 + E3 + E4
  ↳ Repeat for every Param Value
  Minimize Eval

18) Steps of BackPropagation? [Algo]

i) Initialize ALL the weights → small rnd values [-1, 1]
ii) let u(m) & d(m) be the train input & output example
iii) For (m = 1 to M)
- → Present u(m) to the network → Compute outputs
- → using them → calc $\frac{\partial E}{\partial w}$ of each layer
- → update the weights → $w_{new}^{(l)} = w_{old}^{(l)} - \alpha \frac{\partial E}{\partial w}^{(l)}$

iv) Compute the total error → stop in case of convergence