Designing ML Systems

Class imbalance

→ Have weights in confusion matrix

$$L(x; \theta) = \sum_j C_{ij} P(j|x_i, \theta)$$

Class-balanced loss

Normalize weight by no. of examples in each class

$$w_i = \frac{N}{\text{no of samples in } i}$$

Inversely proportional to let rarer classes have more weight.

$$L(x; \theta) = w_i \sum_j P(j|x, \theta) \, Loss(x, j)$$

↓ Loss of x when it is classified as j

Loss can be cross entropy or any other loss function

More effective loss can use effective sample size (ESS) to account for overlap between samples.

Focal loss

Help the model focus on classifying difficult examples correctly more often.

$$FL(p_t) = -(1-p_t)^{\gamma} \log(p_t)$$

Feature Engineering

Missing values

MNAR - Missing not at Random

Data not available because missing value is the true value itself
Eg:- People not disclose income
→ Rich people tend to not talk about income

MAR - Missing at Random

Data missing through another observed variable
Eg: Gender values missing because people don't want to disclose

MCAR - Missing completely at Random

No pattern is the missing values
Eg: Missing value in 'Job' since no job for person

Scaling

Normalization → Between a range

If arbitrary range [a, b] then

$$x_{scaled} = a + \left(\frac{x_{original} - x_{min}}{x_{max} - x_{min}}\right) \times (b-a)$$

Standardization → Mean 0 & unit variance

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

Mitigate skewness with log transformation

For unseen categories, use hashing trick

→ Hash all values of category (such as brand name in Amazon, which can be > 2 Million)

→ Hash will cause collisions
→ Effect of collisions is not very bad. Even for 50% collision rate, perf loss is less than 0.5%.

Discrete & Continous Positional Embeddings

Explicit position of words (0, 1, 2 ..) don't work because, generally, NN don't work well with features that are not unit variance (that's why features are scaled)

Rescaling positions between 0 & 1 would make the values too small to be meaningful.

Treat like embedding. Columns of embedding are maximum no. of possible positions (i.e length)

If seq len = 8, the no. of positions = 0..7

Embedding size for positions = embedding size for word

Embedding at 0 for 'food' = Sum of emb of 'food' stored at idx 0

Fixed position embedding?

Combination of sine & cosine funcs

If i % 2 == 0:
  use sine
else
  use cosine

Also called fourier features
↓
Help learning when positions are continous features

Feature coverage

Percentage of samples that has values for this feature in the dataset
⇒ ↓ values missing → ↑ coverage

If coverage of feature is different between train & test set ⇒ train & test dataset don't come from the same distribution.
→ Check if split is correct or if there is data leakage.

Model Training & Dev

Prediction assumption
→ Possible to predict Y from X

IID
→ NN assume samples are IID

Smoothness
→ set of functions that allow similar inputs to be grouped (same for outputs). Supervised learning makes this assumption

Tractability
→ For X & latent Z, it should be possible to compute P(Z|X).

Boundaries
→ Linear boundaries for linear classifier

Conditional independence
→ Naive bayes assumes sample independence

Normally distributed
→ Many models assume normally distributed data.

Bagging (Bootstrap aggregation)

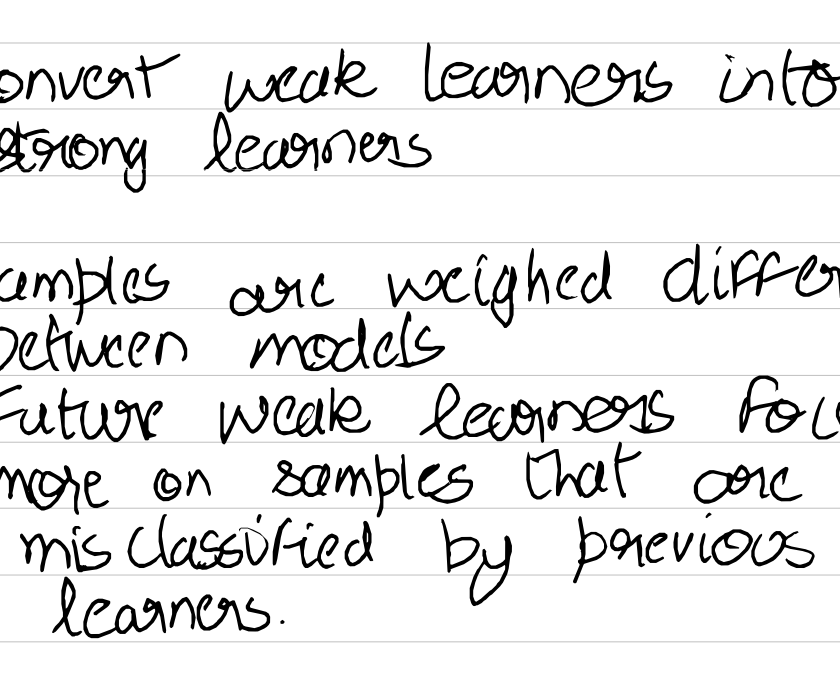
→ Sample with replacement to create multiple datasets (bootstrap)
→ Train classifier
→ Use aggregation on predictions (Majority voting, etc.)

Boosting

→ Convert weak learners into strong learners

→ Samples are weighed differently between models
→ Future weak learners focus more on samples that are misclassified by previous weak learners.

Stacking



Model Evaluation

→ Perturbation tests

→ Ensure model inputs mimic inputs in real life
Eg:- Audio inputs can have BG noise, music, etc.

→ Make changes to your dataset to mimic prod settings.

→ Invariance tests

→ Certain changes to inputs shouldn't lead to changes in outputs

→ Avoid using sensitive info in models

→ Directional Expectation tests

→ Certain changes to I/P should cause expected changes in output
Eg:- ↑ bedrooms should ↑ price of house

→ Model calibration

→ Say model predicts A > B with 70% prob
→ Out of 1000 tests, model predicts A > B only 600 times (i.e 60% prob)
→ Model is NOT calibrated.

→ Calibrated model should predict 60% probability

Measure calibration using counting

Count no. of times model predicts prob x & frequency (y) of that prediction coming true.

Plot x against y using sklearn

Confidence measurement

→ Which measurements to show to user?
→ Don't show low confidence metrics since that will ↓ user satisfaction

Slice-based evaluation

→ Seperate data into subsets & evaluate
→ ALWAYS Report slice based metrics along with coarse-grained metrics (like F1 score, accuracy)

→ How to know critical slices?
→ Heuristics
Use domain knowledge
→ Error Analysis
Find patterns in misclassified examples
→ Slice Finder
→ Generate slices using algos like beam search, clustering, then prune out bad candidates & then rank.

Chapter 8 - Data Distribution Shifts

Causes of ML failures

1. Software system failures
   - Dependency → Package, libs, etc
   - Deployment → wrong version, permissions
   - Hardware → CPU overhead, broken memory
   - Downtime → Server shutdown comp

2. ML specific failures
   → Mismatched data distribution
   → Edge cases: low failure rate but ↑ consequences

   → Degenerate feedback loop:
   → System outputs give feedback to inputs, making a loop of biased influence.

Data Distribution Shifts