Machine Learning Systems Design

Lecture 4: Model development & training

Zoom poll:

How do you feel about breakout exercises?

CS 329 | Chip Huyen

Logistics

- If you don’t have a team yet, search for a teammate now!
- Project mentor sign-up sheet: sign-up now. Most slots are this Wed/Thu!
- Assignment 1 due this Wednesday
- Late policy:
  - Each student has 2 24-hour grace periods
- Lecture notes 3 & 4 will be posted end of week

Agenda

1. Breakout exercise
2. Sampling
3. Class imbalance
4. Data augmentation
5. Feature engineering & data leakage
6. Model selection & training

Breakout exercise

Sampling exercise (group of 5, 7 minutes)

You want to build a model to classify whether a tweet spreads misinformation.

- **10M tweets** from **10K users** over the last **24 months**
- # tweets/user follows a long-tail distribution
- You estimate 1% of tweets are misinformation

Questions:

1. How would you sample 100K tweets to label?
2. You get 100K labels from 20 annotators and want to look at some labels to estimate the quality.
    a. How many labels would you look at?
    b. How would you sample?
3. Imagine you have a stream of unknown number of tweets coming in, can't fit all in memory. How to sample 10K tweets such that each tweet has an equal chance of being selected?

Sampling

Types of sampling

- Non-probability sampling
  - Convenience sampling: selection based on availability
    - Soliciting response
    - Choosing existing datasets
    - Looking at available reviews on Amazon
  - Snowball sampling: future samples are selected based on existing samples
    - E.g. to scape legit Twitter accounts, start with seed accounts then scrape their following
  - Judgment sampling: experts decide what to include
  - Quota sampling: quotas for certain slices of data (no randomization)
  - ....

Data used in ML is mostly driven by convenience

- Language models: BookCorpus, CommonCrawl, Wikipedia, Reddit links
- Sentiment analysis: IMDB, Amazon
  - Only users who have access to the Internet and are willing to put reviews online
- Self-driving cars: most data is from the Bay Area (CA) and Phoenix (AZ)
  - Very little data on raining & snowing weather

⚠️ Lots of biases in data! ⚠️

Types of sampling

- Non-probability sampling
- Random sampling
  - Simple random sampling
  - Stratified sampling
  - Weighted sampling
  - Importance sampling
  - Reservoir sampling
  - ...

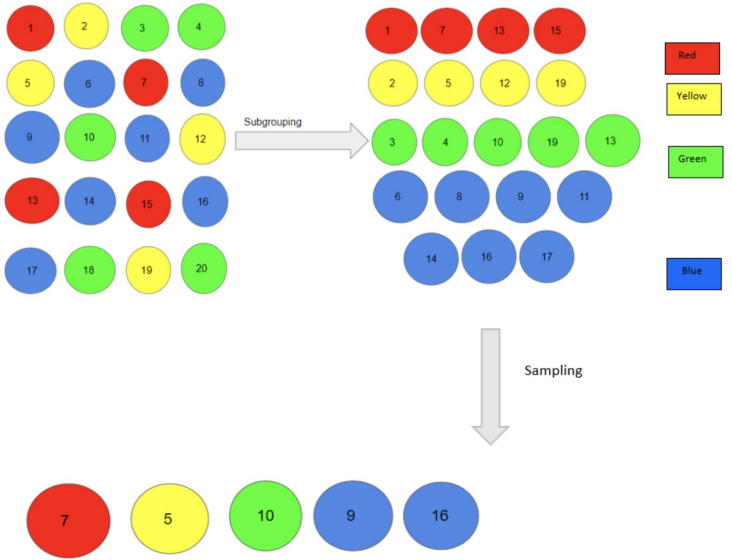
Simple random sampling

- Each sample in population has an equal chance of being selected
  - E.g. select 10% of all samples in population

Pros	Cons
• Simple (easiest type of random sampling)	• No representation guarantee: might exclude rare classes (black swan!)

Stratified sampling

- Divide population by subgroups
  - Slices of data
    - 20% of each age group:
      18-24, 25-34, 35+, etc.
  - Classes
    - 2% of each class



Pros	Cons
• Minor groups are represented	Can't be used when: • samples can't be put into subgroups • samples can belong in multiple subgroups (multilabel)

Image from https://www.analyticsvidhya.com/blog/2019/09/data-scientists-guide-8-types-of-sampling-techniques/

Weighted sampling

- Each element is given a weight, which determines the probability of being selected.
  - If you want to select a sample 30% of the time, give it 3/10 weight
- Might embed domain knowledge
  - E.g. know distribution of your target population or want to prioritize recent samples

```
random.choices(population=[1, 2, 3, 4, 100, 1000],
               weights=[0.2, 0.2, 0.2, 0.2, 0.1, 0.1],
               k=2)
```

```
random.choices(population=[1, 1, 2, 2, 3, 3, 4, 4, 100, 1000],
               k=2)
```

Weighted sampling -> sample weights

- Each element is given a weight, which determines how much it affects the loss function.
  - E.g. give critical samples higher weight to punish wrong predictions on them more
- Can modify decision boundaries

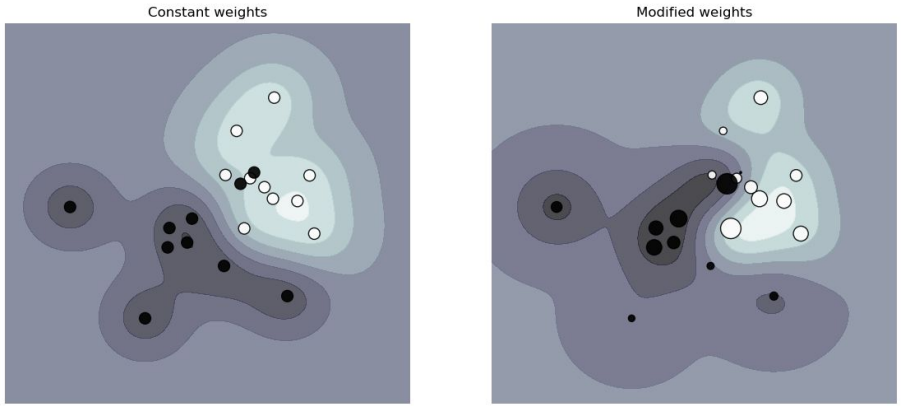


Image from https://scikit-learn.org/stable/auto_examples/svm/plot_weighted_samples.html

Importance sampling

- Useful when sampling from P(x) is expensive, slow, or infeasible
  - Sample x ~ Q(x) ← easier to sample from
  - Weight by P(x)/Q(x)
- ⚠️ Calculating P(x)/Q(x) might be expensive ⚠️

$$E_{P(x)}[x] = \sum_x P(x)\, x = \sum_x Q(x)\, x\, \frac{P(x)}{Q(x)} = E_{Q(x)}[x\frac{P(x)}{Q(x)}]$$

Importance sampling

- Useful when sampling from P(x) is expensive, slow, or infeasible
  - Sample x ∼ Q(x) ← easier to sample from
  - Weight by P(x)/Q(x)
- ⚠️ Calculating P(x)/Q(x) might be expensive ⚠️
- E.g. essential for RL
  - Too expensive to estimate reward under new policy, so use old policy

$$E_{P(x)}[x] = \sum_x P(x)\, x = \sum_x Q(x)\, x\, \frac{P(x)}{Q(x)} = E_{Q(x)}[x\frac{P(x)}{Q(x)}]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\bar{\pi}_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\,(\mathbf{a}_t|\mathbf{s}_t)\left(\prod_{t'=1}^{t}\frac{\pi_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{\bar{\pi}_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})}\right)\left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'},\mathbf{a}_{t'})\right)\right]$$

use old policy to sample data

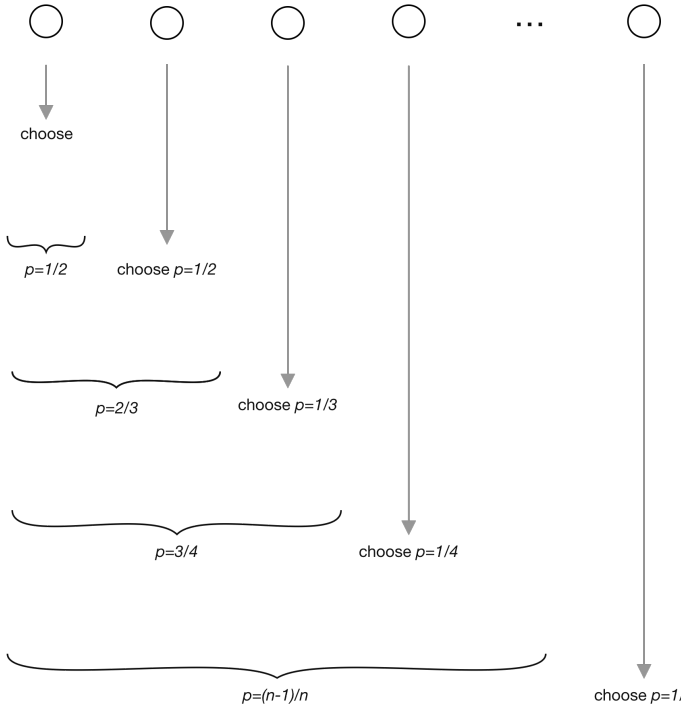
old policy

Reservoir sampling: problem

- Need select k samples from a stream of n samples with equal probability
  - n is unknown
  - impossible/inefficient to fit all in memory
- Can stop the stream any moment and get the required samples

Reservoir sampling: solution

1. First k elements are put in reservoir
2. For each incoming i^{th} element, generate a random number j between 1 and i
   a. If $1 \leq j \leq k$: replace j^{th} in reservoir with i^{th}
3. Each incoming element has k/i chance of being in reservoir!



Also checkout algorithm L (based on geometric distribution)

Image from https://www.analyticsvidhya.com/blog/2019/09/data-scientists-guide-8-types-of-sampling-techniques/

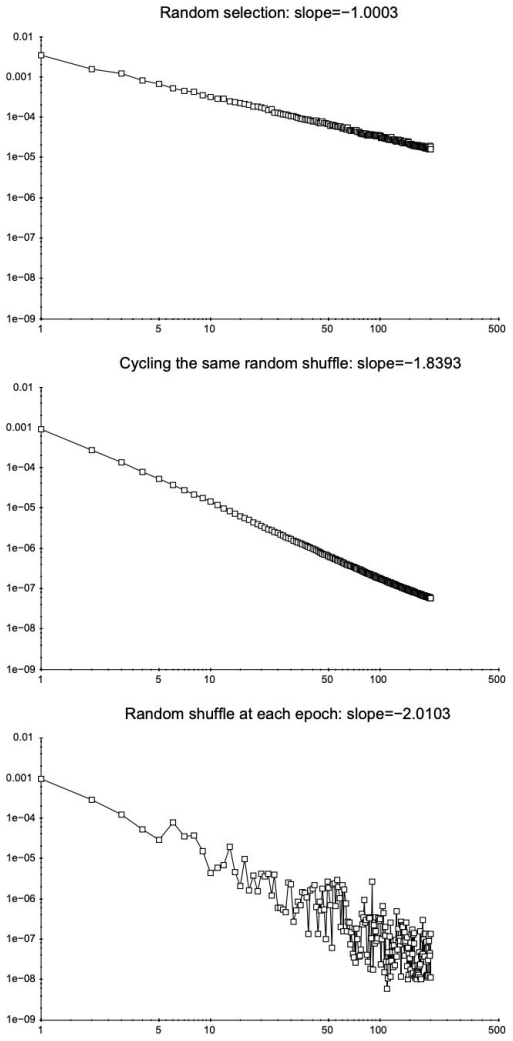
With vs. without replacement

With replacement	**Without replacement**
Same item can be chosen more than once	Same item can't be chosen more than once
• No covariance between two chosen samples • Approximate true population distribution	• Covariance between two chosen samples • Covariance reduced as dataset size becomes large
Bagging	Mini-batch gradient descent

Why do we use epochs instead of just sampling with replacement from the entire dataset?

With vs. without replacements

Because empirically it's converged faster
(proven for strongly convex loss functions)



Curiously Fast Convergence of some Stochastic Gradient Descent Algorithms (Leon Bottou, 2009)

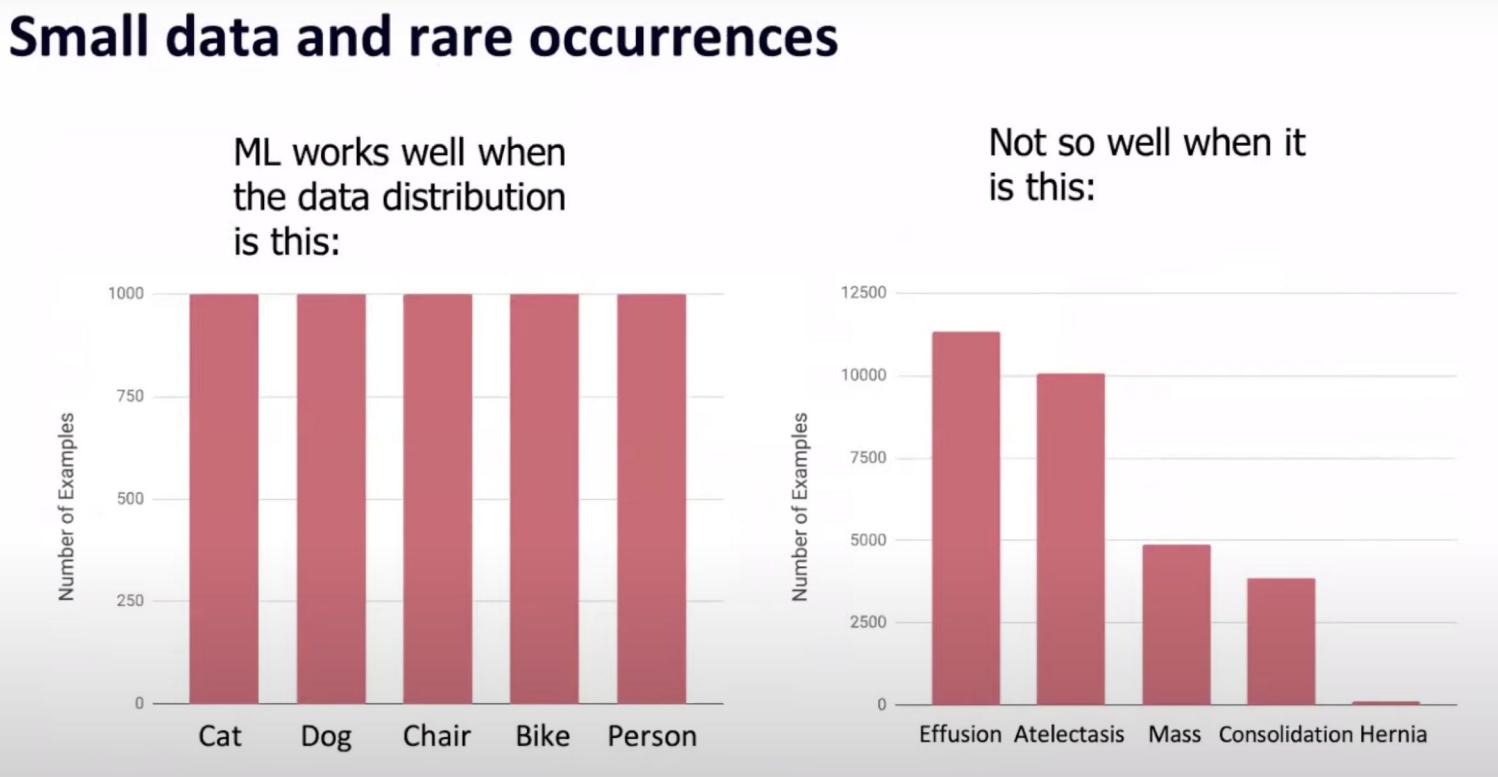
Why Random Reshuffling Beats Stochastic Gradient Descent (Gurbuzbalaba et al., 2015)

Selection bias

- Samples are chosen in a way that is not reflective of their real-world distribution.
- Example:
  - Collecting IMDB reviews to estimate how good a movie is excludes all people who watch movies but don’t leave online reviews.

Class imbalance

Class imbalance



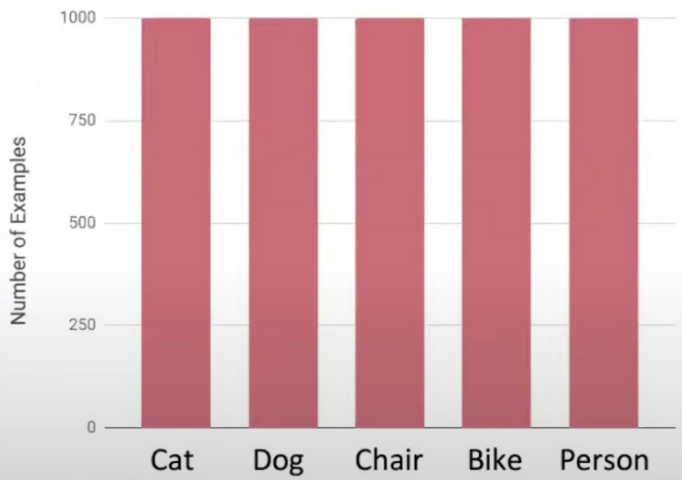
https://www.youtube.com/watch?v=tsPuVAMaADY&ab_channel=StanfordHAI

Class imbalance

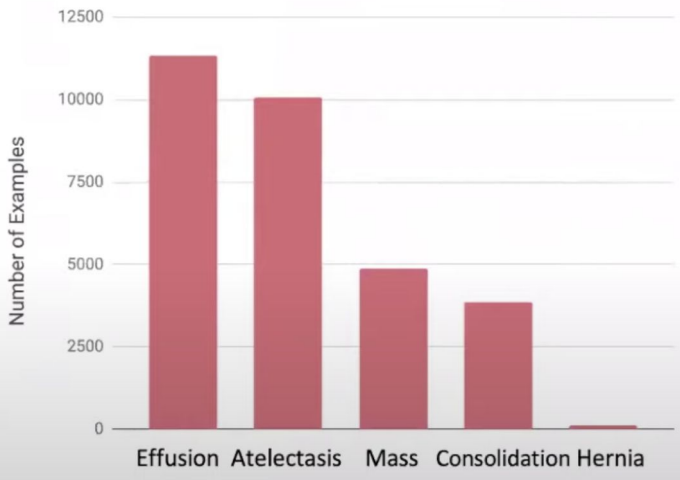
Why?

Small data and rare occurrences

ML works well when the data distribution is this:



Not so well when it is this:



https://www.youtube.com/watch?v=tsPuVAMaADY&ab_channel=StanfordHAI

Why is class imbalance hard?

- Not enough signal to learn about rare classes

Why is class imbalance hard?

- Not enough signal to learn about rare classes
- Statistically speaking, predicting majority label has higher chance of being right
  - If a majority class accounts 99% of data, always predicting it gives 99% accuracy

Why is class imbalance hard?

- Not enough signal to learn about rare classes
- Statistically speaking, predicting majority label has higher chance of being right
- Imbalance often comes with differences in cost of wrong predictions

Class imbalance is the norm

- Fraud detection
- Spam detection
- Disease screening
- Churn prediction
- Resume screening
  - E.g. 2% of resumes pass screening
- Object detection
  - Most bounding boxes don't contain any object

People are more interested in unusual/potentially catastrophic events

Image from PyImageSearch

Sources of class imbalance

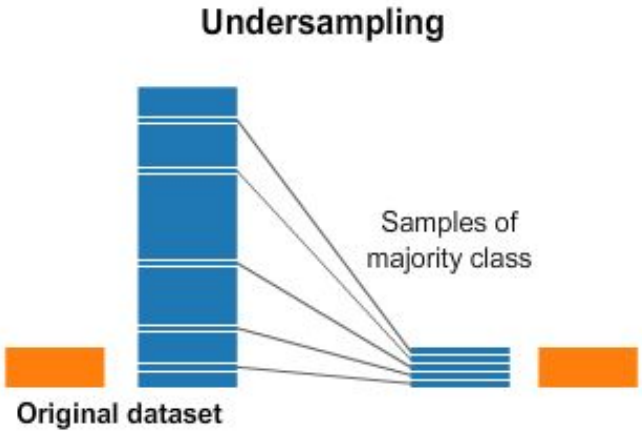
- Sampling biases
  - Narrow geographical areas (self-driving cars)
  - Selection biases
- Domain specific
  - Costly, slow, or infeasible to collect data of certain classes
- Labeling errors

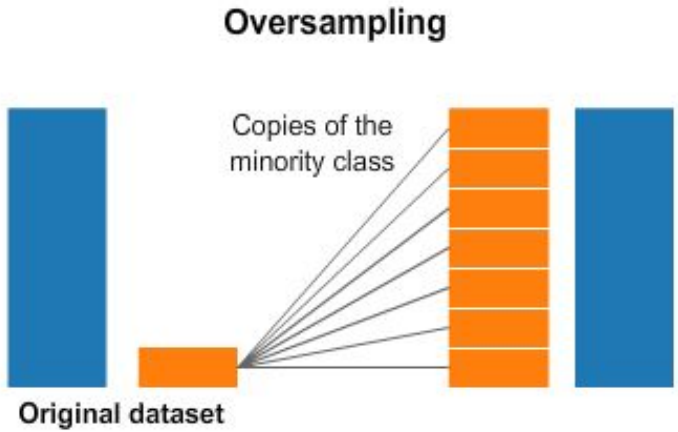
How to deal with class imbalance

- Add more minority samples or remove majority samples (resampling)
- Adjust losses (weight balancing)
- Choose algorithms robust to class imbalance

Class imbalance solution: Resampling

Undersampling	Oversampling
Remove samples from the majority class	Add more examples to the minority class
Can cause overfitting	Can cause loss of information





https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets#t1

Undersampling: Tomek Links

- Find pairs of close samples of opposite classes
- Remove the sample of majority class in each pair
  - Pros: Make decision boundary more clear
  - Cons: Make model less robust

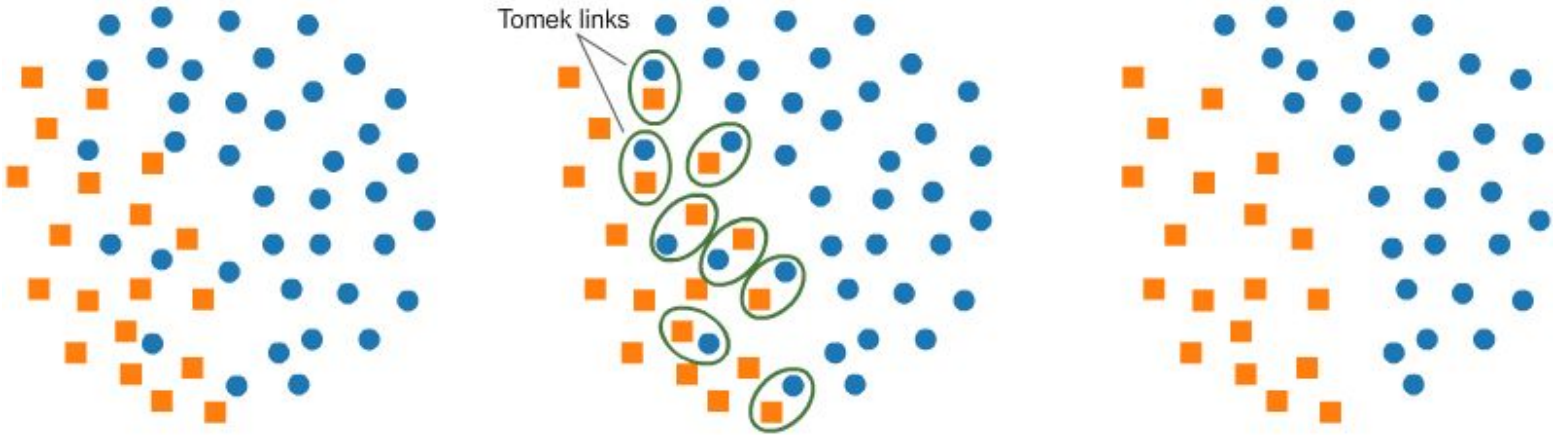


Image from https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

Oversampling: SMOTE

- Synthesize samples of minority class as convex (~linear) combinations of existing points and their nearest neighbors of same class.

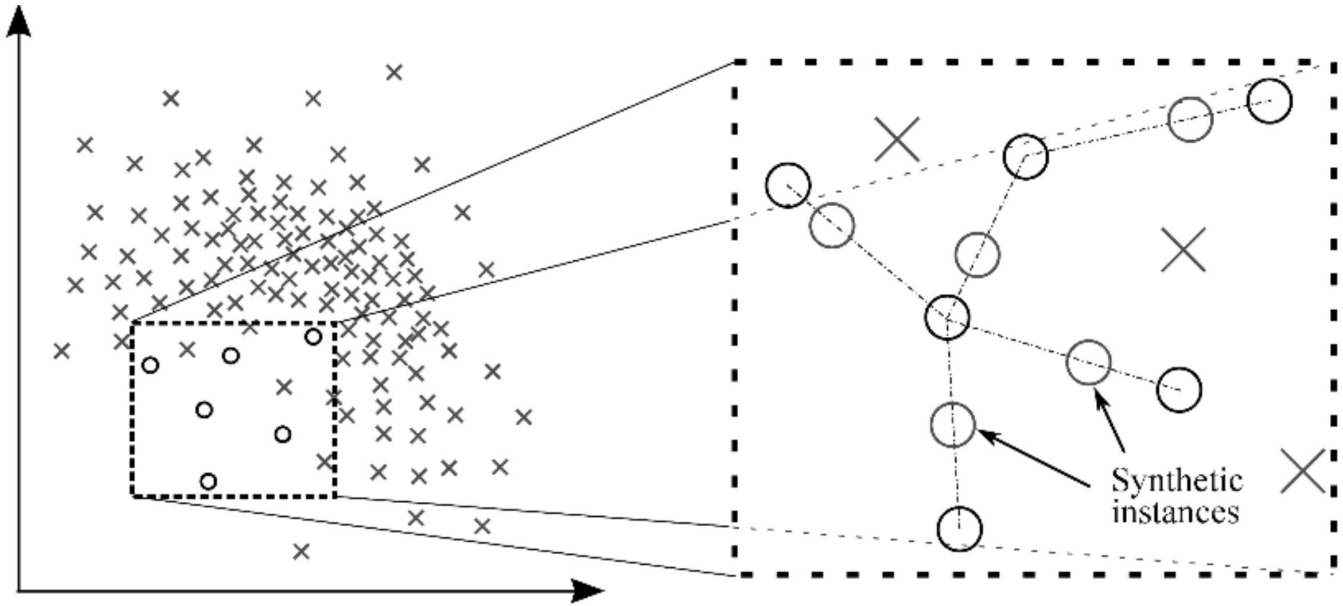


Image from Analytics Vidhya

Oversampling: SMOTE

Both SMOTE and Tomek links only work on low-dimensional data!

- Synthesize samples of minority class as convex (~linear) combinations of existing points and their nearest neighbors of same class.

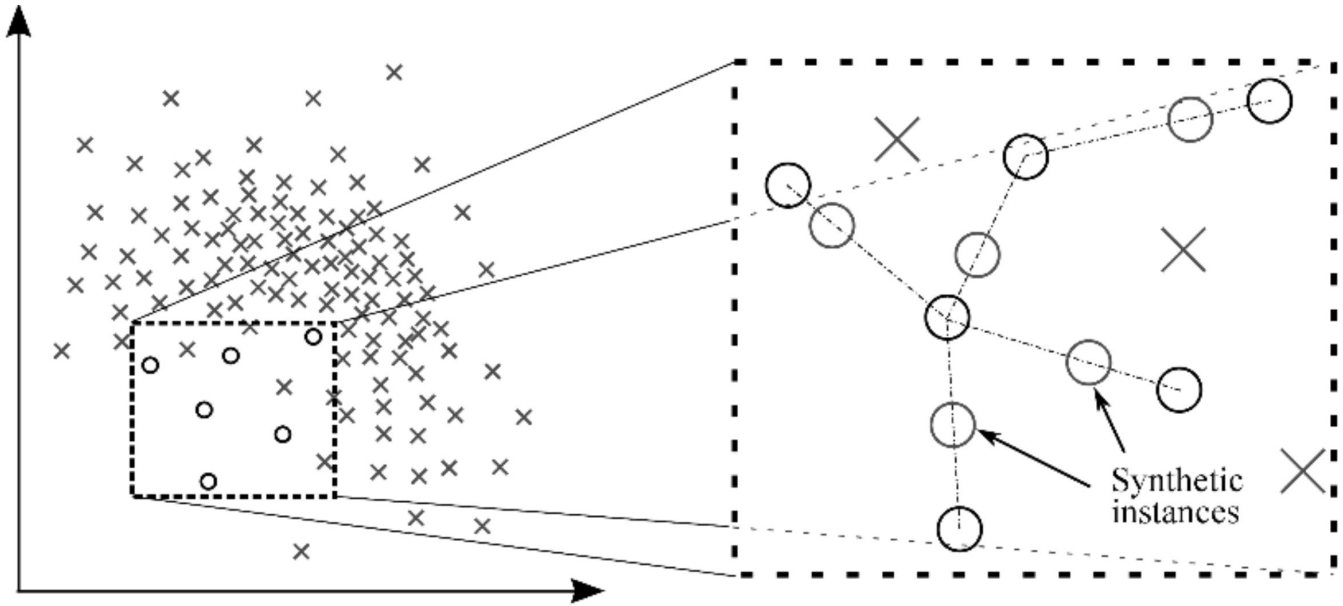


Image from Analytics Vidhya

Class imbalance solution: weight balancing

- Give more weight to rare classes

Non-weighted loss

$$L(X;\ \theta)\ =\ \sum_i L(x_i;\theta)$$

Weighted loss

$$L(X;\ \theta)\ =\ \sum_i W_{y_i} L(x_i;\theta)$$

$$W_c\ =\ \frac{N}{number\ of\ samples\ of\ class\ C}$$

```
model.fit(features, labels, epochs=10, batch_size=32, class_weight={“fraud”: 0.9, “normal”: 0.1})
```

Class imbalance solution: weight balancing

- Give more weight to rare classes
- Give more weight to difficult samples:
  - downweighs well-classified samples

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases}$$

$$\mathrm{CE}(p_t) = -\log(p_t)$$

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

Focal Loss for Dense Object Detection (Lin et al., 2017)

Class imbalance solution: algorithms

- Sample **with** replacement to create different datasets
- Train a classifier with each dataset
- Aggregate predictions from classifiers

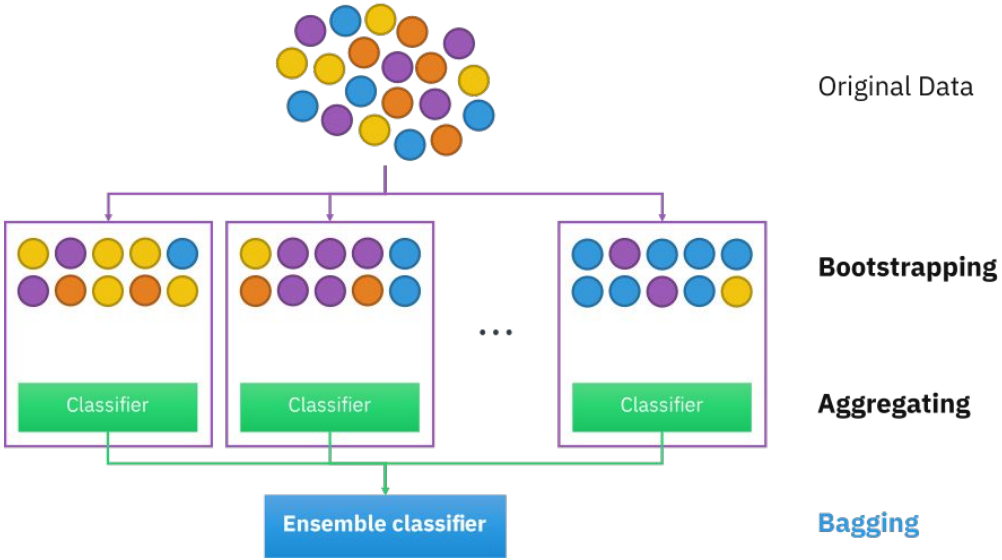


Illustration by Sirakorn

Class imbalance solution: algorithms

1. Train a weak classifier
2. Give samples misclassified by the weak classifier more weight
3. Repeat (1) on this reweighted data

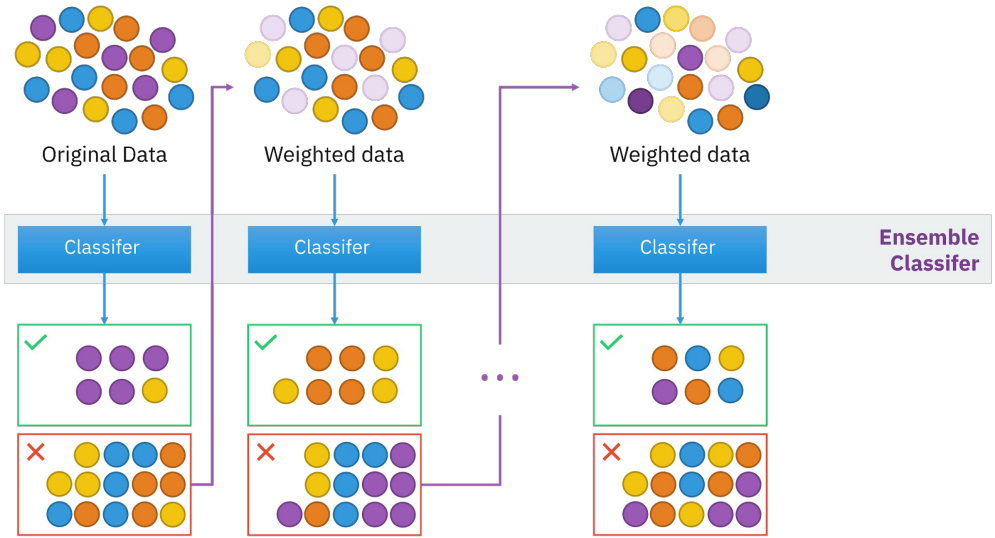


Illustration by Sirakorn

Data augmentation

Data augmentation: goals

- Improve model’s performance overall or on certain classes
- Generalize better
- Enforce certain behaviors

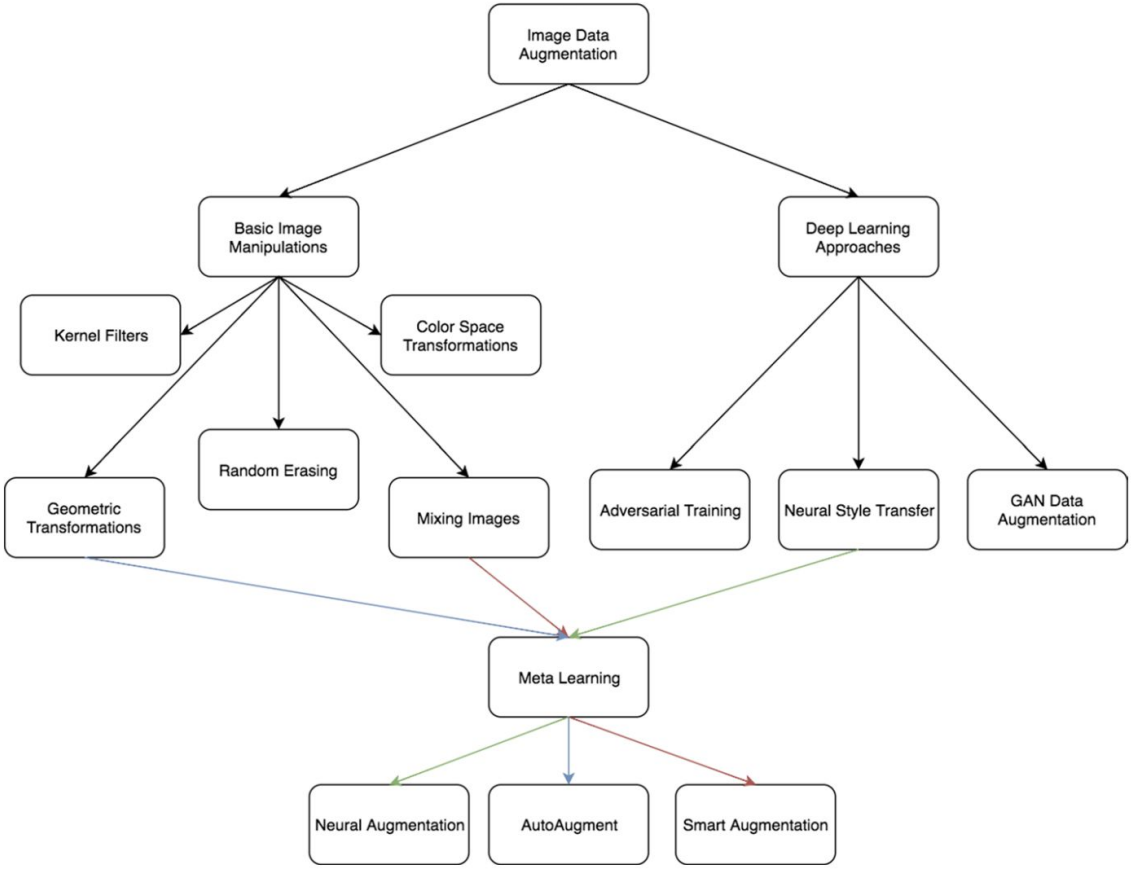
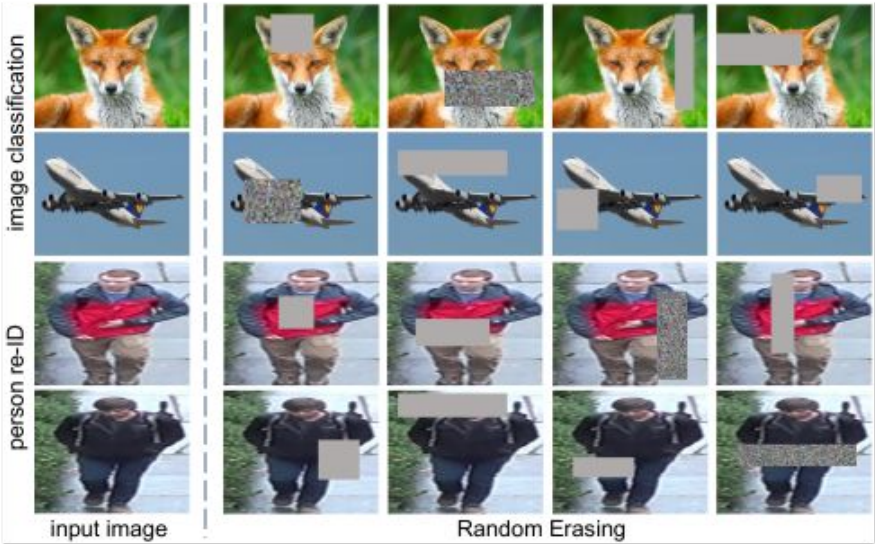


Fig. 2 A taxonomy of image data augmentations covered; the colored lines in the figure depict which data augmentation method the corresponding meta-learning scheme uses, for example, meta-learning using Neural Style Transfer is covered in neural augmentation [36]

A survey on Image Data Augmentation for Deep Learning (Connor Shorten & Taghi M. Khoshgoftaar, 2019)

Data augmentation: image

- Artificially enlarge the dataset using label-preserving image manipulations
  - random cropping, flipping, erasing



Random Erasing Data Augmentation (Zhong et al., 2017)

mixup

- Create convex combination of samples of different classes
  - Original: image 1 - label 3 (cat), image 2 - label 4 (dog). All labels are discrete
  - Mixup: 70% dog, 30% cat - label 3.3
- Incentivize models to learn linear relationships
  - Assumption: linear behaviour reduces the amount of undesirable oscillations when predicting outside the training examples

mixup: Beyond Empirical Risk Minimization (Zhang et al., 2017)

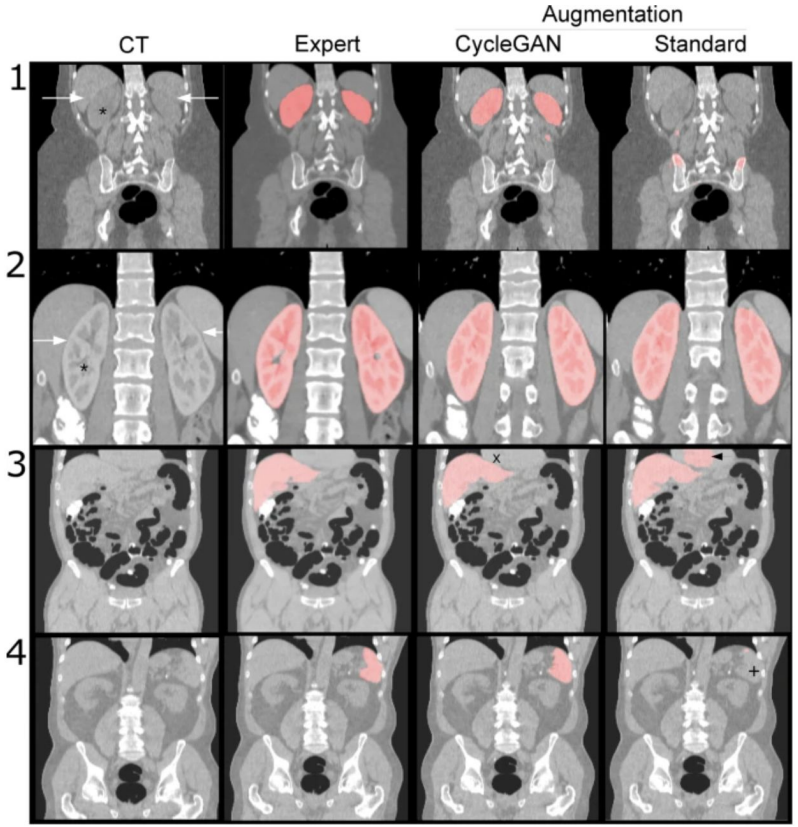
mixup

- Improves generalization on speech and tabular data
- Can be used to stabilize the training of GANs

https://forums.fast.ai/t/mixup-data-augmentation/22764

Data augmentation: GAN

Example: kidney segmentation with data augmentation by CycleGAN



Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks (Sandfort et al., 2019)

Data augmentation: text

- Random replacement
  - e.g. BERT (10% * 15% = 1.5%)

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., 2018)

Feature engineering

Engineered features vs. learned features

- Feature engineering: extract features to use in your model

Isn’t the promise of deep learning is that you won’t have to engineer features?

Engineered features: text

Stopword removal I have a dog. He's sleeping.

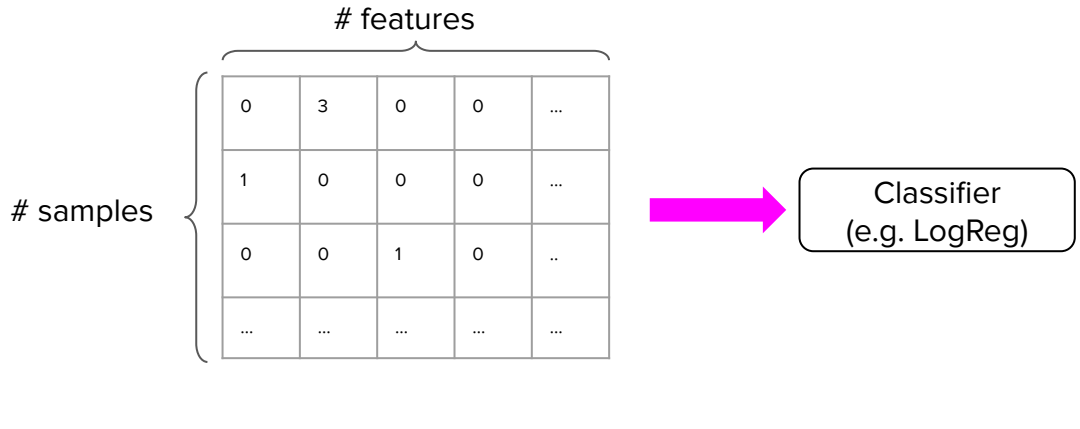
Lemmatization I have dog. He's sleeping.

Contraction I have dog. He's sleep.

Punctuation I have dog. He is sleep.

Lowercase I have dog He is sleep

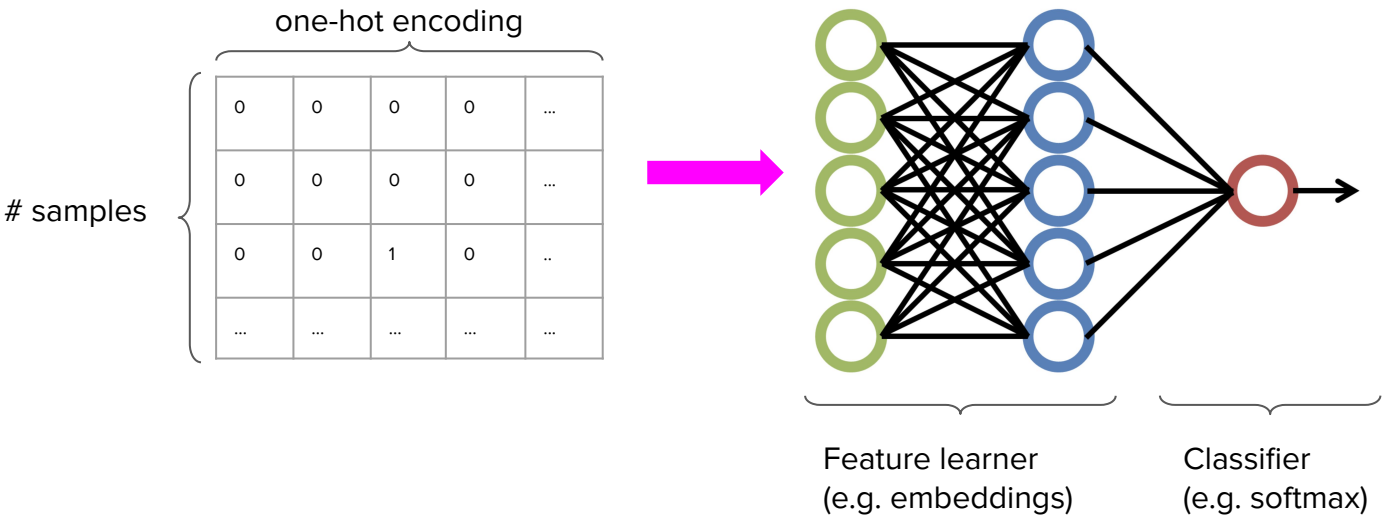
N-gram i have dog he is sleep



Features

I	you	have	dog	cat	he	she	is	they	sleep	I, have	have, dog	good, dog	...
1	0	1	1	0	1	0	1	0	1	1	1	0	...

Learned features: text



Text I have a dog. He's sleeping.

One-hot

I	you	have	dog	cat	he	she	is	they	sleep	mom	food	yes	...
1	0	1	1	0	1	0	1	0	0	0	0	0	...

Learned features: spam classification

Comment ID	Time	User	Text	# ↑	# ↓	Link	# images	Thread ID	Reply to	# replies	...
93880839	2020-10-30 T 10:45 UTC	gitrekt	Your mom is a nice lady.	1	0	0	0	2332332	n0tab0t	1	...

User ID	Created	User	Subs	# ↑	# ↓	# Threads	# replies	Karma	Verified email	Sex	...
4402903	2015-01-57 T 3:09 PST	gitrekt	[r/ml, r/memes, r/socialist]	15	90	28	776	304	No		...

Thread ID	Time	User	Text	# ↑	# ↓	Link	# images	# replies	# visits	Award	...
9388320839	2020-10-30 T 2:45 PST	doge	Human is temporary, AGI is forever	120	50	1	0	32	2405	1	...

Feature engineering: spam classification

Even more features:

- Post frequency, max posts per day
- Post repetitiveness
- Language detection, typos, abnormal punctuations, ratio uppercase/lowercase
- IP, other users from the same IP
- NSFW words, blacklisted links
- Targeted users
- ...

Feature engineering

- For complex tasks, number of features can go up to millions or billions!
- Lots of ML production work involves coming up with new features
  - Fraudsters come up with new techniques very fast, so need to come up with new features very fast to counter
- Often require subject matter expertise

Data leakage

- Some form of the label “leaks” into the features
- This same information is not available during inference

Data leakage: example 1

- Problem: detect lung cancer from CT scans
- Data: collected from hospital A
- Performs well on test data from hospital A
- Performs poorly on test data from hospital B

Patient ID	Date	Doctor note	Medical record	Scanner type	CT scan

Data leakage: example 1

- Problem: detect lung cancer from CT scans
- Data: collected from hospital A
- Performs well on test data from hospital A
- Performs poorly on test data from hospital B

Patient ID	Date	Doctor note	Medical record	Scanner type	CT scan

At hospital A, when doctors suspect that a patient has lung cancer, they send that patient to a higher-quality scanner

Data leakage: example 2

- Problem: predicting how many views an article will get
- Data: historical data on the site
- Where might data leakage come from?

Article ID	Date	Title	Article	Author	Language	Translations

Data leakage: example 2

- Problem: predicting how popular an article will become
- Data: historical data on the site

Not leakage because author popularity also available during inference

Article ID	Date	Title	Article	Author	Language	Translations

The site only translate articles that are already gaining attention

Data leakage

- What if leakage is baked into training samples?
  - Remove all leaked information
  - Normalization

Types of data leakage

- Feature leakage
- Training data leakage
  - Premature featurization: creating feature on the entire data instead of just training data
    - E.g. create n-gram counts/vocabulary from train + test sets
  - Oversampling before splits
    - Train splits might contain test samples
  - Time leakage
    - Randomly splitting data instead of temporal split can cause training data to be able to see the future
  - Group leakage
    - A patient has 3 CT scans: 2 in train, 1 in test.

How to avoid leakage?

Lots of data exploration and testing!

- Check for duplication between train and valid/test splits
- Temporal split data (if possible)
- Use only train splits for feature engineering: no peak!
- Measure correlation between features and labels
- Train model on subset of features
  - If performance very high on a subset, either very good set of features or leakage!
- Monitor model performance as more features are added
  - Sudden increase: either a very good feature or leakage!
- Involve subject matter experts in the process

Feature engineering: the more the better?

- Adding more features tends to improve model performance

Feature engineering: the more the better?

- Adding more features tends to improve model performance
- Overfitting
- Too many features makes your model different to maintain
- The more feature engineering you do, the more chance for labels to leak
    - Feature A doesn't cause leakage, but feature A + feature B can cause leakage

Feature engineering: the more the better?

- Adding more features tends to improve model performance
- Overfitting
- Too many features makes your model different to maintain
- The more feature engineering you do, the more chance for labels to leak
    - Feature A doesn't cause leakage, but feature A + feature B can cause leakage

- Clean up features that are no longer relevant
- Store them in case you want to reuse them

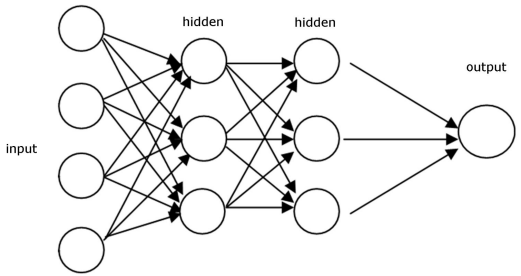
Model selection & training

Classical ML algorithms

- Logistic regression
  - Very hard to beat baseline
- Tree-based models: random forest, bagging, boosting
  - XGBoost still one of the most popular winning algorithms on Kaggle
- K-nearest neighbor
  - Great for anomaly detection
- SVM
- ...

Neural architecture evolution

Feed-forward NNs

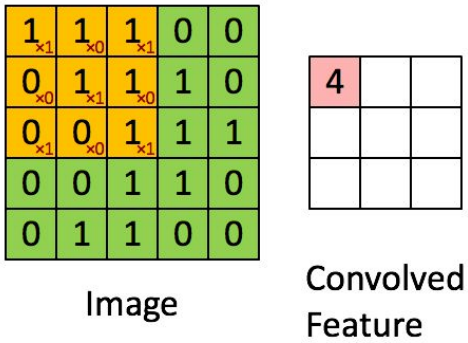


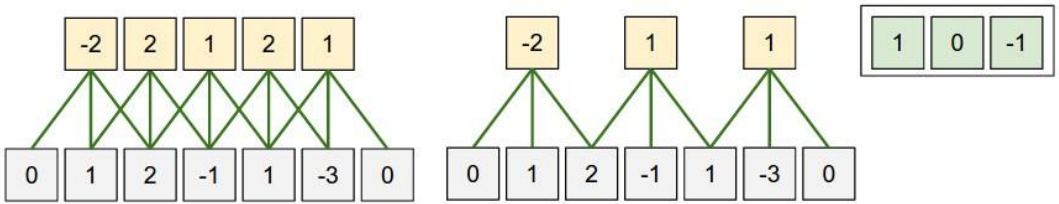
Neural architecture evolution

local information → **CNNs**

Feed-forward NNs

Instead of connecting each neuron to all input, connect each neuron to a subset of inputs





Left image from http://deeplearning.stanford.edu/, right image from S231N Lecture 7.

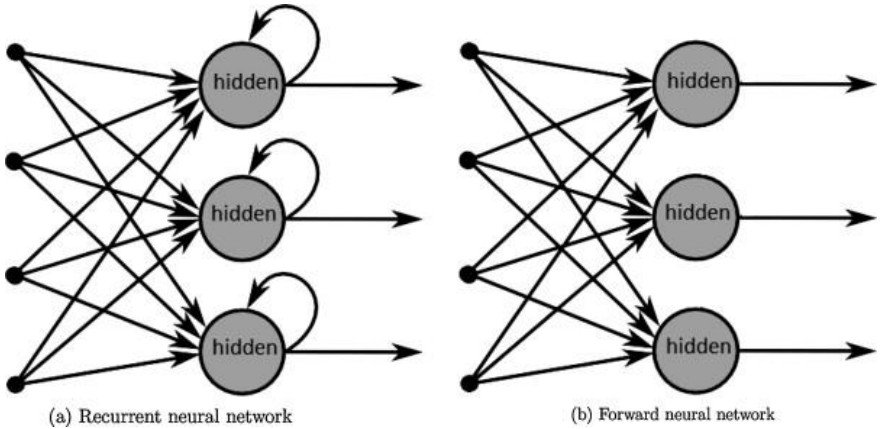
Neural architecture evolution

local information → **CNNs**

Feed-forward NNs

weights sharing across steps → **RNNs**

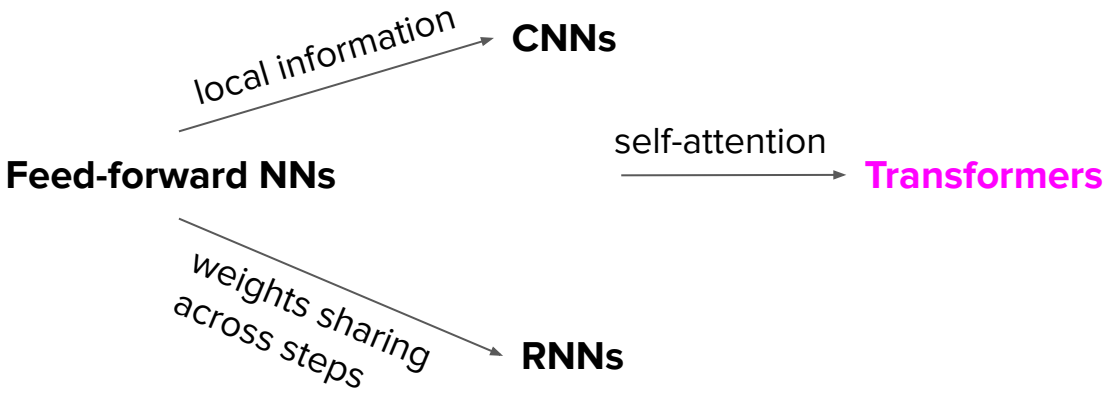
Instead of using different weights for different timesteps, use same weights!



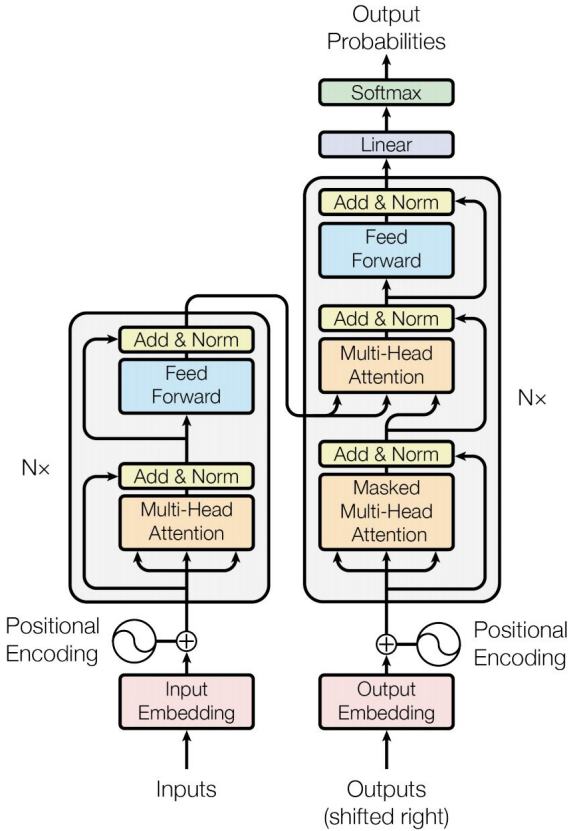
(a) Recurrent neural network (b) Forward neural network

"A survey on the application of recurrent neural networks to statistical language modeling." (De Mulder et al., 2015)

Neural architecture evolution

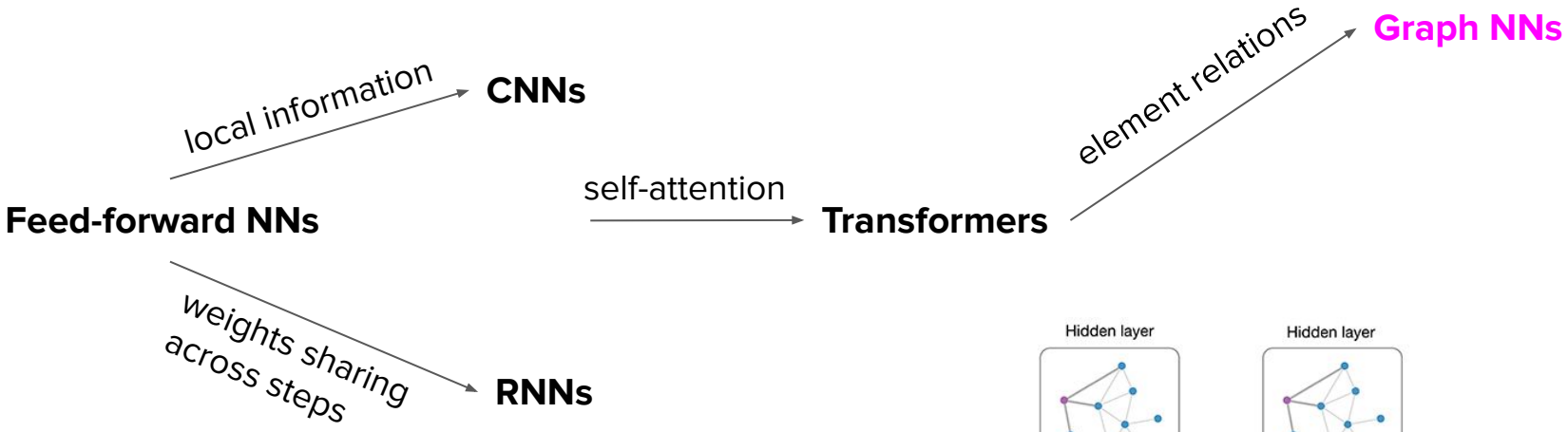


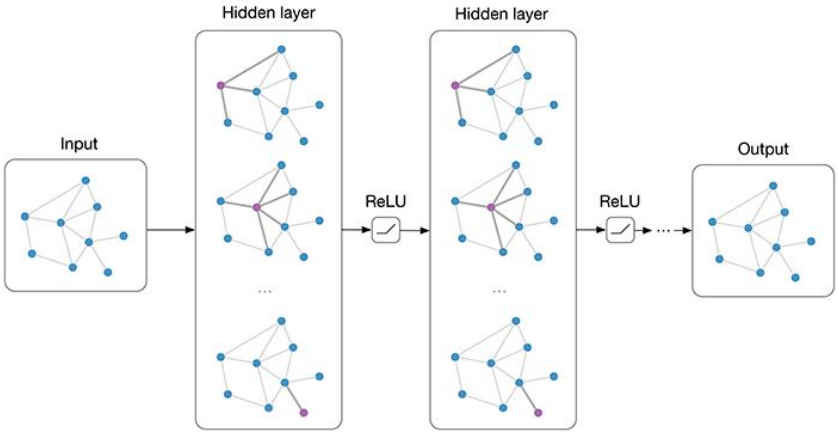
Allow classifier to peak at all previous hidden states



Attention Is All You Need (Vaswani et al., 2017)

Neural architecture evolution



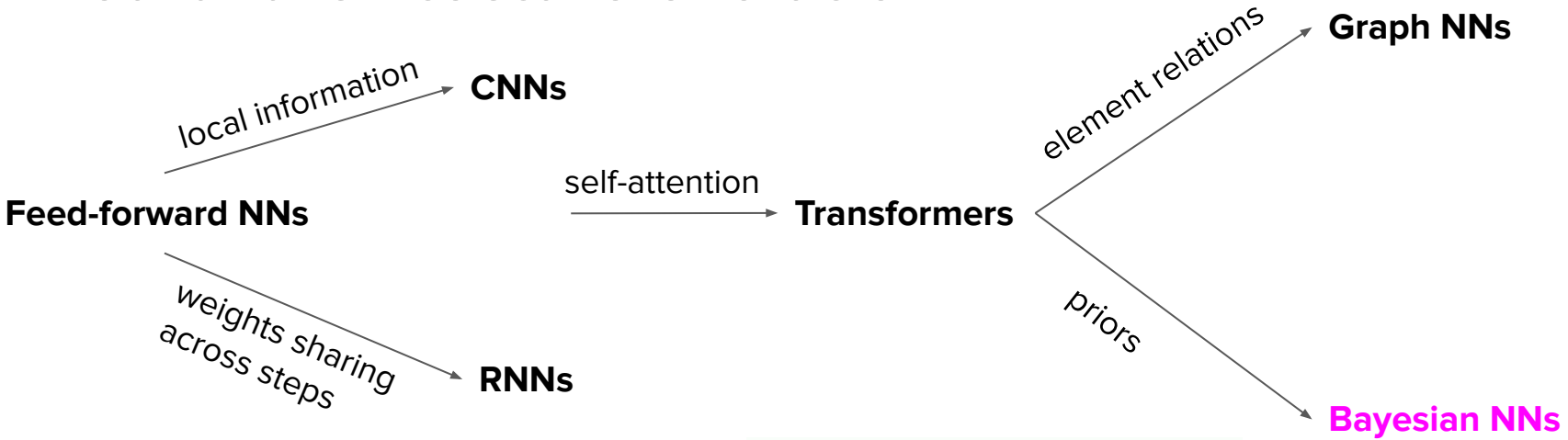


Graphs are natural representations for:

- inputs: social networks, knowledge bases, game states
- outputs: any joint distribution can be represented as a factor graph

Graph Convolutional Networks (Thomas Kipf, 2016)

Neural architecture evolution



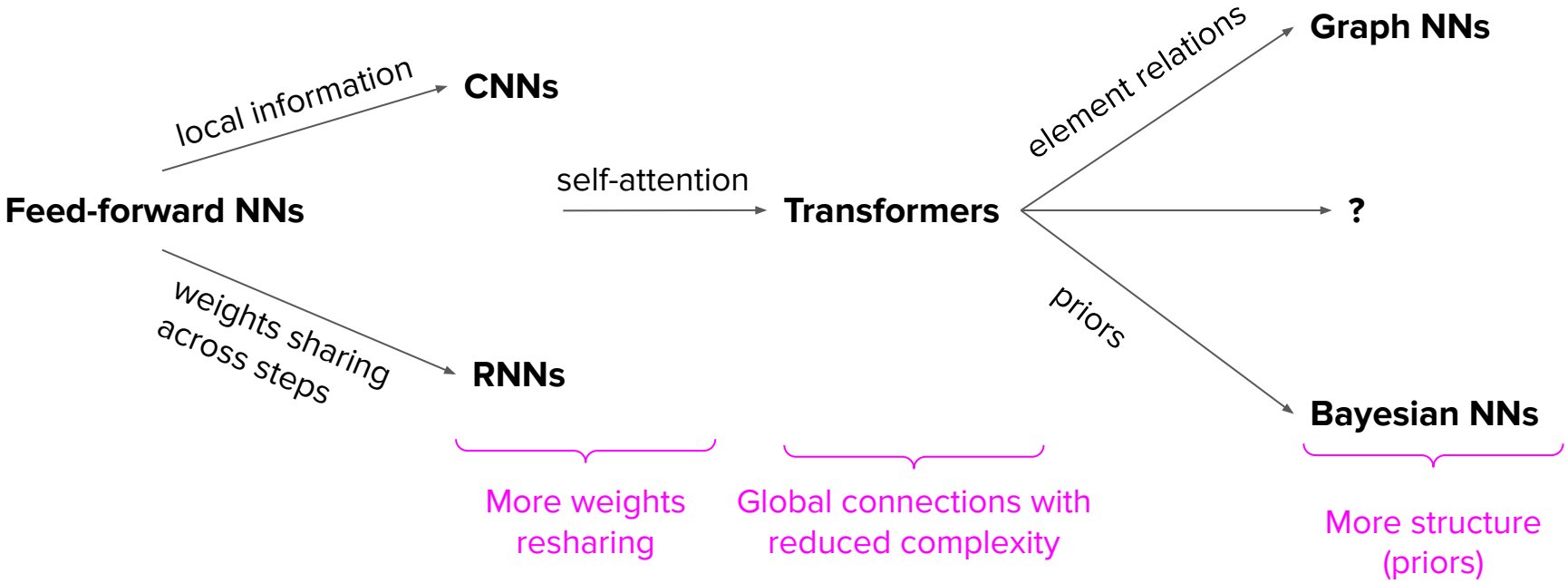
Use a probability distribution over the network weights and output an average prediction of all sets of weights in that distribution

Bayesian learning	Deep learning
Bayesian models (GPs, BayesNets, PGMs,)	Deep models (MLP, CNN, RNN etc.)
Bayesian inference (Bayes rule)	Stochastic training (SGD, RMSprop, Adam)

	Bayes	DL
Can handle large data and complex models?	✗	✓
Scalable training?	✗	✓
Can estimate uncertainty?	✓	✗
Can perform sequential / active /online / incremental learning?	✓	✗

Deep Learning with Bayesian Principles (Emtiyaz Khan, NeurIPS 2019)

Neural architecture evolution



Architecture evolution

- Architectures come and go
  - LSTM-RNNs: largely replaced by Transformers for text, still used for time series (frequency trading)
- Be solution-focused, not architecture/buzzword-focused

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following same distribution
- **Zero rule baseline**
  - Predict the most common class always
- **Human baseline**
  - Human expert?
- **Simple heuristic**:
  - E.g. autocompletion: predict the character most likely to appear based on a trigram lookup table
- **Existing solutions**:
  - Existing APIs

Don’t be afraid to generate fake data for baselines!

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following same distribution
- **Zero rule baseline**
  - Predict the most common class always
- **Human baseline**
  - Human expert?
- **Simple heuristic**:
  - E.g. autocompletion: predict the most character based on a trigram lookup table
- **Existing solutions**:
  - Existing APIs

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Uniform random	0.5	0.02
Distribution random	0.98	0.01
Most common (preds = [0] * n)	?	?

Model selection: baselines

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

- **Random baseline**
  - Predict at random:
    - uniform
    - following same distribution
- **Zero rule baseline**
  - Predict the most common class always
- **Human baseline**
  - Human expert?
- **Simple heuristic**:
  - E.g. autocompletion: predict the most character based on a trigram lookup table
- **Existing solutions**:
  - Existing APIs

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

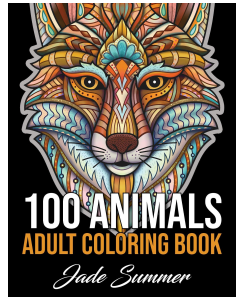
	Accuracy	**F1**
Uniform random	0.5	0.02
Distribution random	0.98	0.01
Most common (preds = [0] * n)	0.99	NaN

1

2

9

4



5

3

6

7

8

10

Machine Learning Systems Design

Next class: Distributed training & framework tutorials