Meta Learning III

CS771: Introduction to Machine Learning

Purushottam Kar

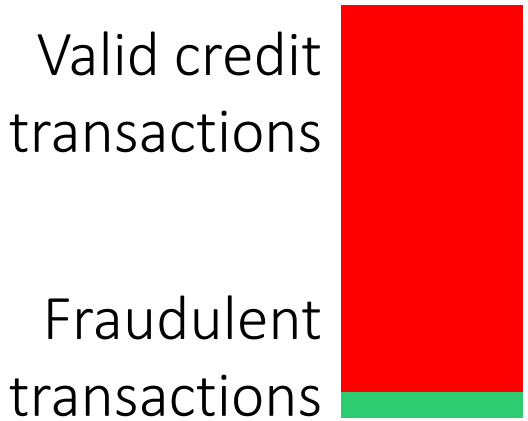
Precap

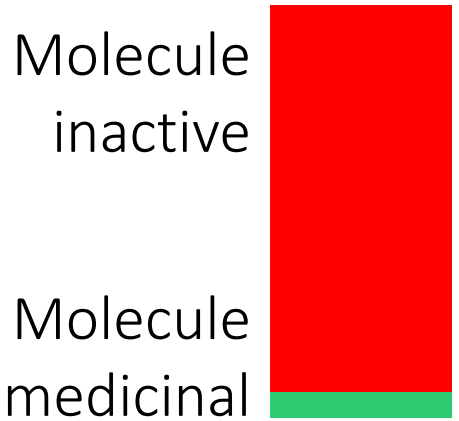
Learning with Imbalanced Data

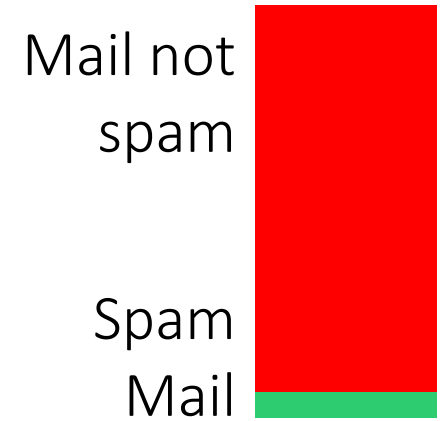
Examples of imbalanced data in ML problems

Techniques to handle imbalanced data

Imbalance is common – Binary Classfn



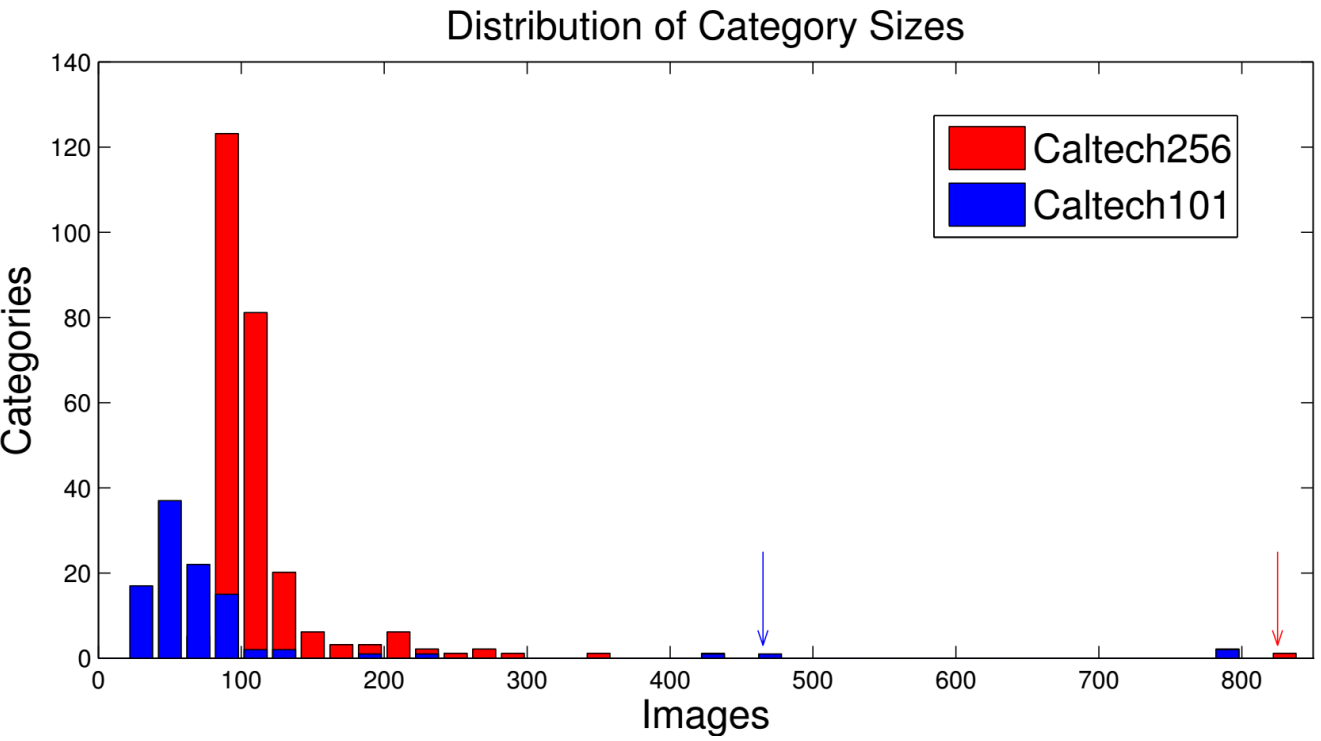




In anomaly detection, medical diagnosis, drug discovery, spam filtering and many other domains, data is heavily biased

- *Lots of data may be available in total but very less (0.01% or lesser) may belong to the "rare" class e.g. 100000 red vs 10 green*
- *Trivial for a classifier to get 99.99% accuracy (simply predict everything red)*
  - May be disastrous, e.g. if green class denotes a rare disease or fraudulent transaction
  - Many ML algorithms that are designed to greedily chase classification accuracy may completely ignore green training data points since it is easy to do well on red data points
  - The whole purpose of learning is lost if the above happens ☹

Imbalance is common – Multi Classfn



GRIFFIN, HOLUB AND PERONA, CALTECH-256 OBJECT CATEGORY DATASET

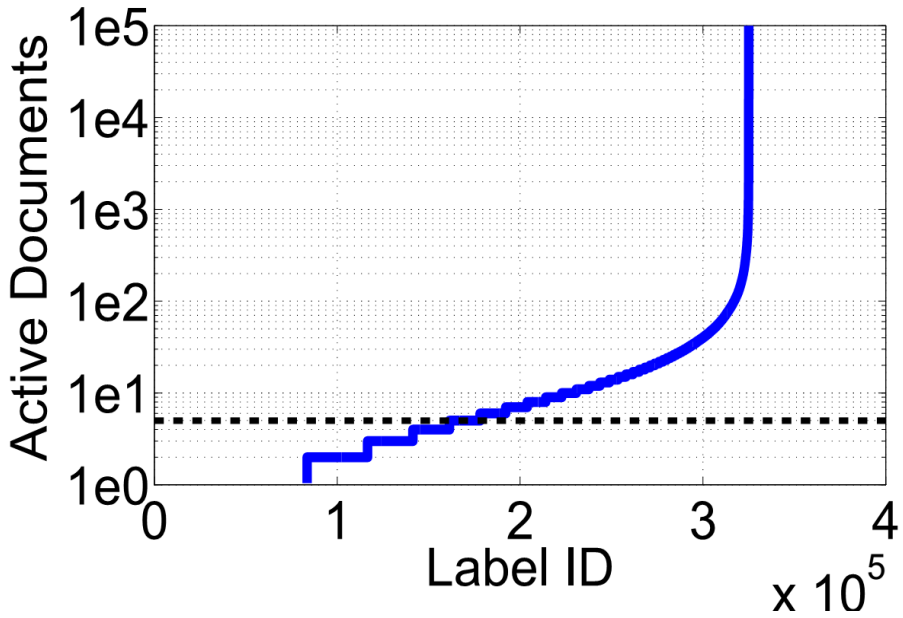
Caltech-256 is an object detection dataset with 256 classes

- *Simple averaging argument shows that 50% of classes must each have less than 0.8% of the data*
- *Lots of data for few classes but most classes impoverished*
- *In fact, rarest class has only 80 images, most popular 800 images*

Makes learning challenging

- *Easier to do well on popular classes with lots of data points e.g. cats*
- *Difficult to handle rare classes*

Imbalance is common – Multilabel Classfn



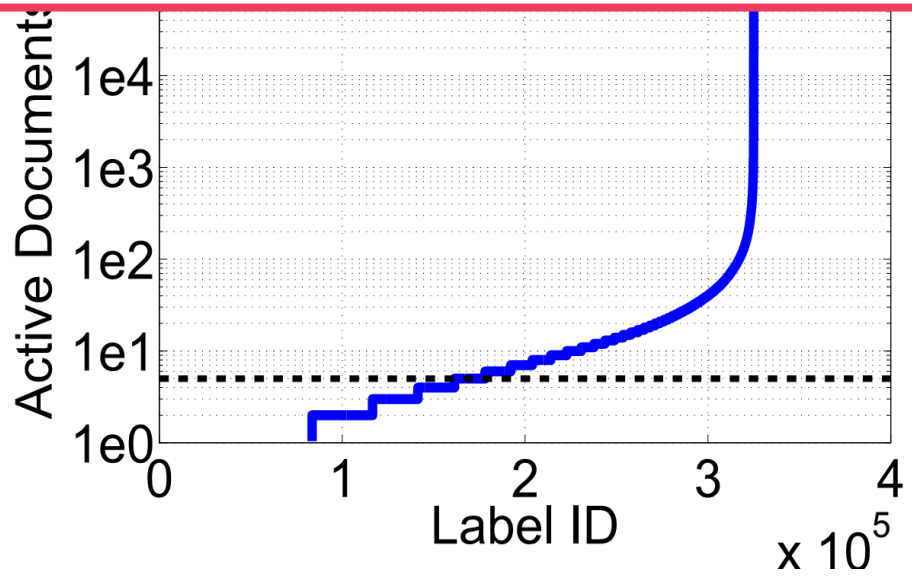
0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0

Wikipedia LSHTC is a multilabel dataset with 300000 labels

- *As many as 150000 labels occur only in less than 5 documents*
- *Most popular label ("living persons") occurs in 100000 documents*
- *Similar situation in recommendation – most people like iPhone but most items liked by a very select group of individuals*
- *Lots of train data to find out who likes iPhone, not enough for most other items*

In

Ironically, whereas we often proudly refer to these as "big-data" problems, they can often be seen as collections of a huge number of "tiny-data" problems (one tiny data problem per rare label) and a few "reasonable-data" problems



0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0

Wikipedia LSHTC is a multilabel dataset with 300000 labels

As many as 150000 labels occur only in less than 5 documents

Most popular label ("living persons") occurs in 100000 documents

Similar situation in recommendation – most people like iPhone but most items liked by a very select group of individuals

Lots of train data to find out who likes iPhone, not enough for most other items

Solutions to Label Imbalance

Get more data for the rare classes so that they don't remain rare ☺

Generate artificial data for rare classes to make them more prominent

Change training algorithms that artificially make rare classes prominent

Includes using loss functions that are aware of class imbalance

Use different learning strategies for rare classes and popular classes

A combination of all of the above may be required in practice

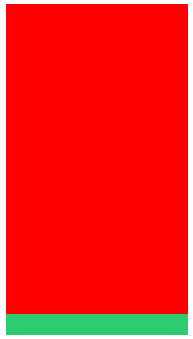
Extreme classification algos (assignment 2) do indeed practice some of these

Dataset Modification Techniques

Dataset Modification Techniques

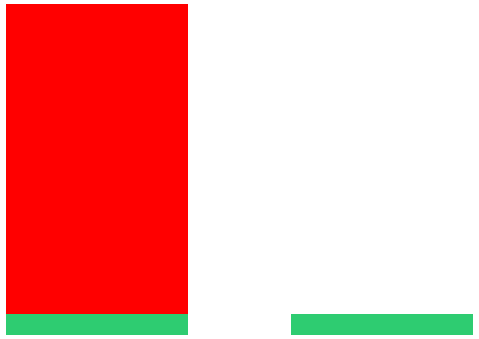
Undersample the popular class so that it does not dominate

Dataset Modification Techniques



Undersample the popular class so that it does not dominate

Dataset Modification Techniques



Undersample the popular class so that it does not dominate

Dataset Modification Techniques

Undersample the popular class so that it does not dominate

Dataset Modification Techniques

Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

Data

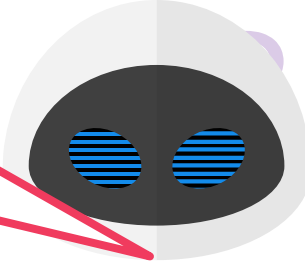
A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead

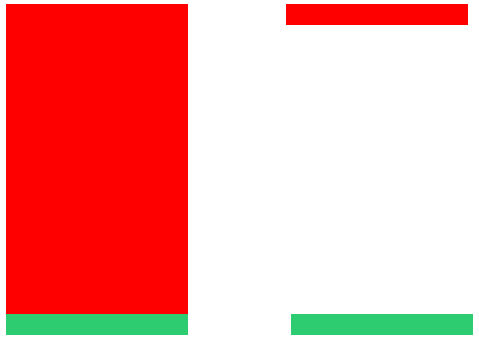
Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead





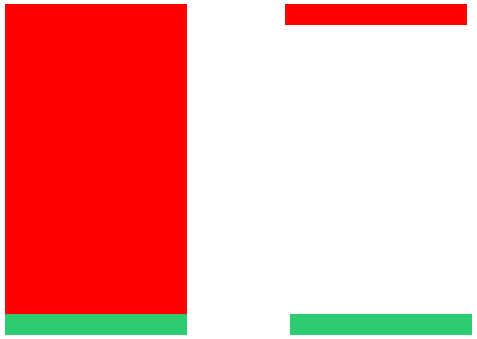
Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

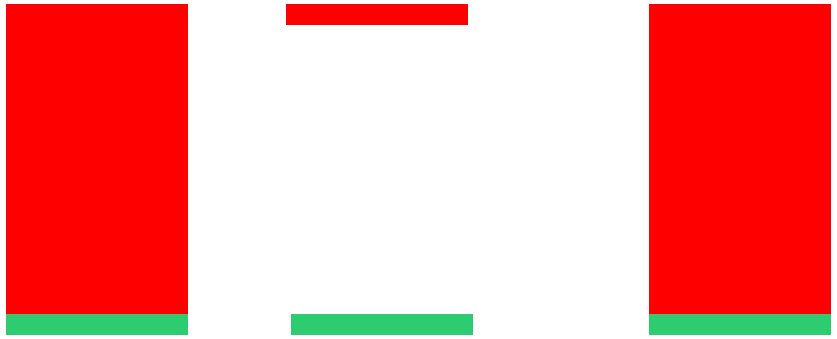
Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

Oversample the rare class by repeating each rare point (say 100 times)

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

Oversample the rare class by repeating each rare point (say 100 times)

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead

Undersample the popular class so that it does not dominate

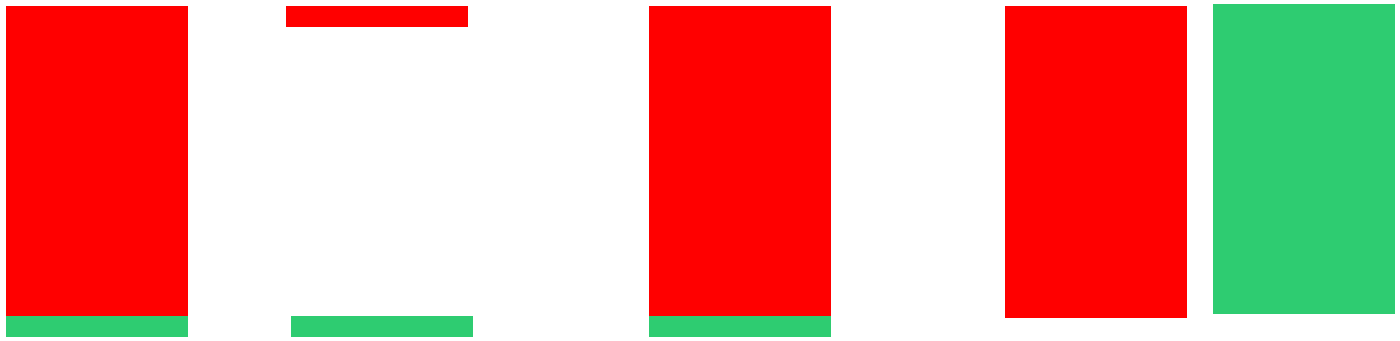
Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

Oversample the rare class by repeating each rare point (say 100 times)

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

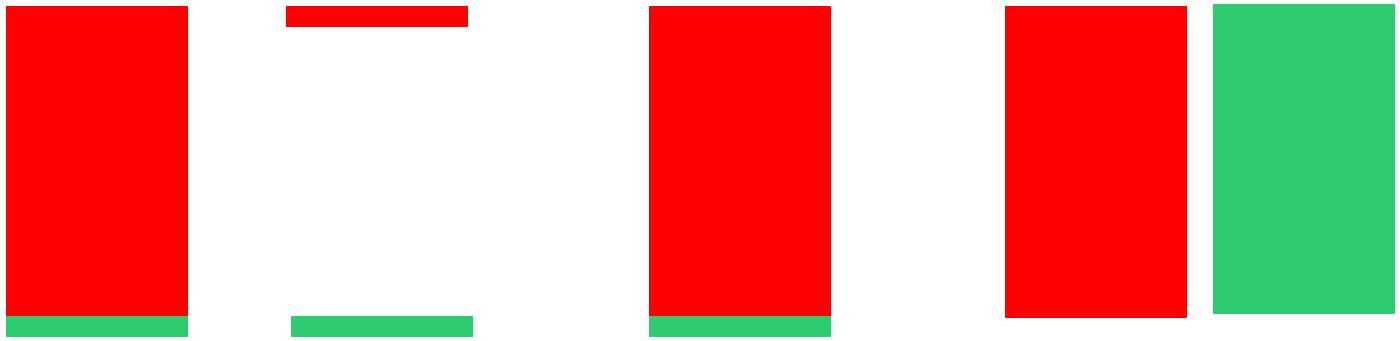
Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

Oversample the rare class by repeating each rare point (say 100 times)

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

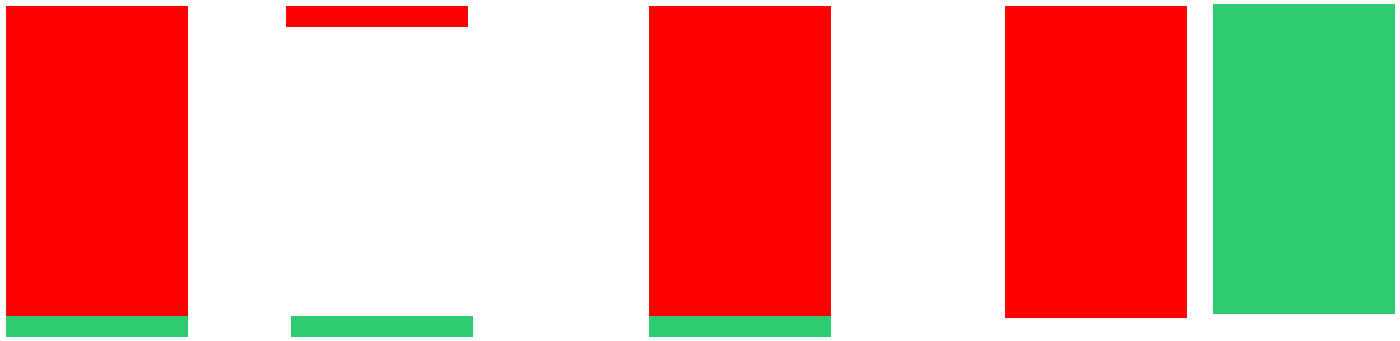
In practice we may give smaller weights to popular class points

Oversample the rare class by repeating each rare point (say 100 times)

Not done explicitly since that would needlessly increase train set size+time

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

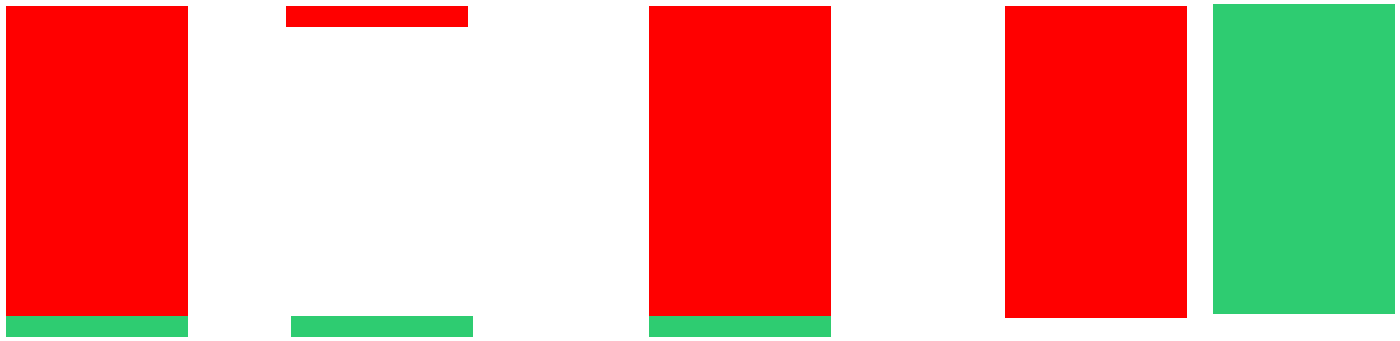
Oversample the rare class by repeating each rare point (say 100 times)

Not done explicitly since that would needlessly increase train set size+time

Implicitly done by giving higher weights to rare class points

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

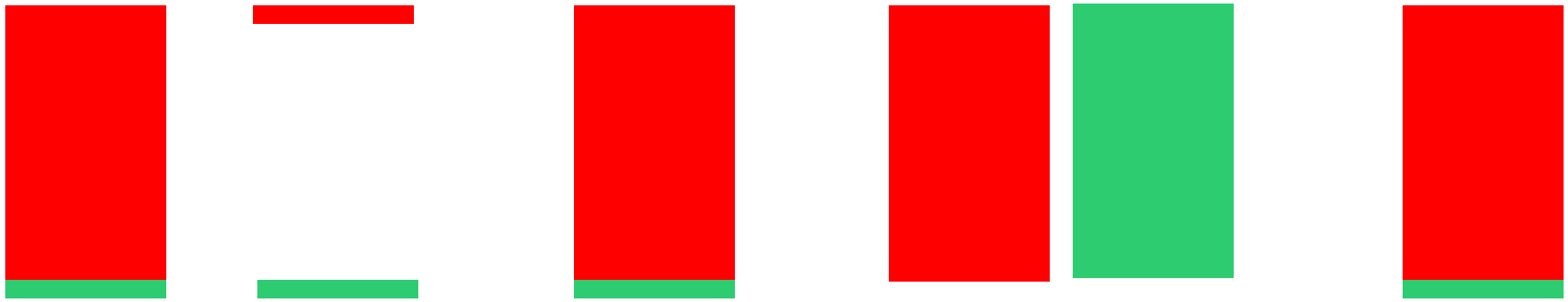
- *Actually throwing out data not a very good idea since it throws away data*
- *In practice we may give smaller weights to popular class points*

Oversample the rare class by repeating each rare point (say 100 times)

- *Not done explicitly since that would needlessly increase train set size+time*
- *Implicitly done by giving higher weights to rare class points*
- *A technique SMOTE (Synthetic Minority Oversampling TEchnique) however actually samples new data points around rare class points*

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

Oversample the rare class by repeating each rare point (say 100 times)

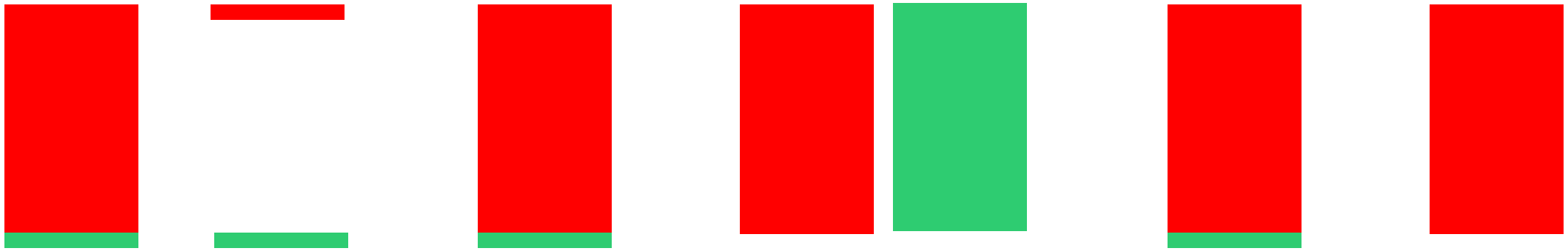
Not done explicitly since that would needlessly increase train set size+time

Implicitly done by giving higher weights to rare class points

A technique SMOTE (Synthetic Minority Oversampling TEchnique) however actually samples new data points around rare class points

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

Oversample the rare class by repeating each rare point (say 100 times)

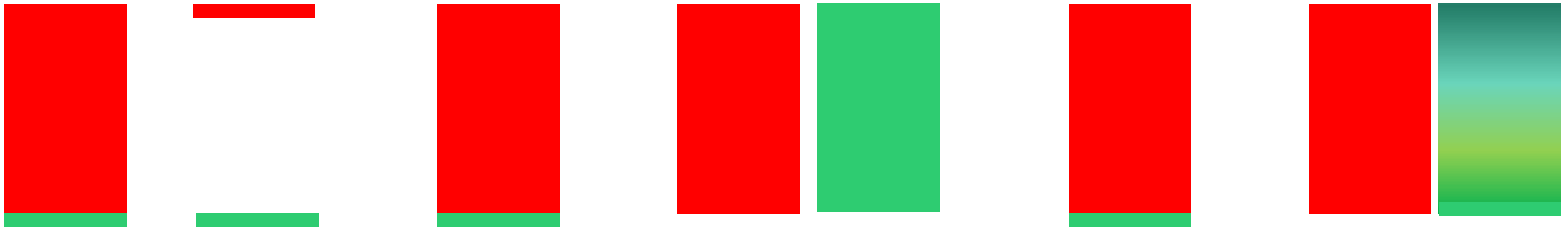
Not done explicitly since that would needlessly increase train set size+time

Implicitly done by giving higher weights to rare class points

A technique SMOTE (Synthetic Minority Oversampling TEchnique) however actually samples new data points around rare class points

Data

A bit pointless to handle label rarity problem by making all labels rare. Would be like a hypothetical administration that aims at equidistribution of wealth but ends up with equidistribution of poverty instead



Undersample the popular class so that it does not dominate

Actually throwing out data not a very good idea since it throws away data

In practice we may give smaller weights to popular class points

Oversample the rare class by repeating each rare point (say 100 times)

Not done explicitly since that would needlessly increase train set size+time

Implicitly done by giving higher weights to rare class points

A technique SMOTE (Synthetic Minority Oversampling TEchnique) however actually samples new data points around rare class points

Imbalance-aware Training Techniques

Simplest are reweighing techniques: add more weight to rare points

Loss functions like hinge loss are very majoritarian – neglect rare classes

Introduce a weight per data point (many libraries already support this)

Classical SVM: $\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \cdot \sum_{i=1}^{n}\left[1 - y^i \cdot \langle \mathbf{w}, \mathbf{x}^i \rangle\right]_+$

Reweighted SVM: *give weight C_+ to positives and C_- to negatives*

$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C_+ \cdot \sum_{i:y^i=+1}\left[1 - \langle \mathbf{w}, \mathbf{x}^i \rangle\right]_+ + C_- \cdot \sum_{i:y^i=-1}\left[1 + \langle \mathbf{w}, \mathbf{x}^i \rangle\right]_+$

How to tune C_+, C_-?

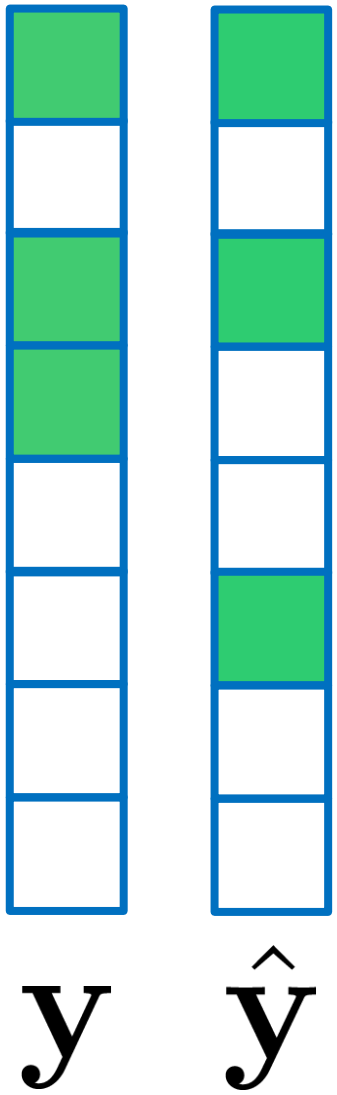
Use inverse popularity. Let $p_+ = \frac{n_+}{n}, p_- = \frac{n_-}{n}$ be proportion of pos/neg

Use $C_+ = \frac{1}{p_+}, C_- = \frac{1}{p_-}$

Other more sophisticated techniques have been discovered recently (see next slides on loss functions for imbalanced training)

Imbalance-aware Training Techniques

Use loss fns that themselves penalize models that neglect rare classes



$\left\lvert\left\{ i : \begin{matrix} \hat{\mathbf{y}}_i = 1 \\ \mathbf{y}_i = 1 \end{matrix} \right\}\right\rvert$ a	$\left\lvert\left\{ i : \begin{matrix} \hat{\mathbf{y}}_i = 1 \\ \mathbf{y}_i = -1 \end{matrix} \right\}\right\rvert$ b
$\left\lvert\left\{ i : \begin{matrix} \hat{\mathbf{y}}_i = -1 \\ \mathbf{y}_i = 1 \end{matrix} \right\}\right\rvert$ c	$\left\lvert\left\{ i : \begin{matrix} \hat{\mathbf{y}}_i = -1 \\ \mathbf{y}_i = -1 \end{matrix} \right\}\right\rvert$ d

$$r_{\text{acc}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{a+d}{a+b+c+d} = p_+ \cdot r_{\text{TPR}} + p_- \cdot r_{\text{TNR}}$$

Classification Accuracy

$$r_{\text{bal}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{r_{\text{TPR}} + r_{\text{TNR}}}{2}$$

Balanced Accuracy

$$r_{\text{TPR}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{a}{a+c}$$

True Positive Rate (= Recall)

$$r_{\text{TNR}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{d}{b+d}$$

True Negative Rate

$$r_{\text{prec}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{a}{a+b}$$

Precision

$$F(\mathbf{y}, \hat{\mathbf{y}}) = \frac{2 \cdot r_{\text{prec}} \cdot r_{\text{rec}}}{r_{\text{prec}} + r_{\text{rec}}}$$

F-measure

$$r_{\text{G}}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{r_{\text{TPR}} \cdot r_{\text{TNR}}}$$

G-mean

$$r_{\text{Min}}(\mathbf{y}, \hat{\mathbf{y}}) = \min\{r_{\text{TPR}}, r_{\text{TNR}}\}$$

MinPerf

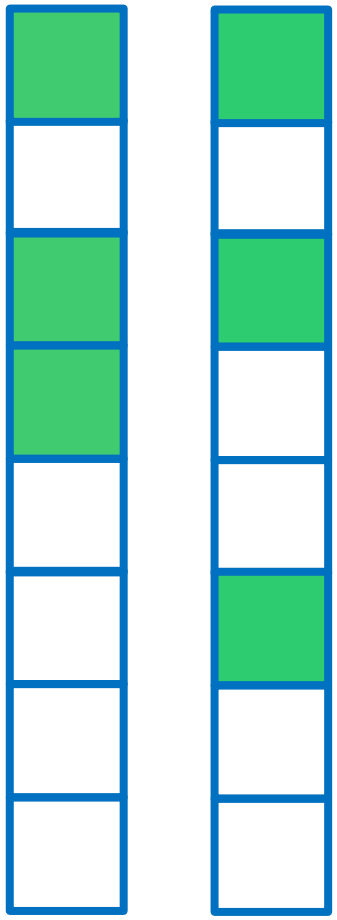
$$r_{\text{rec}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{a}{a+c}$$

Recall

Imbala

This is called the confusion matrix of a classifier and records how many data points that truly belong to class c got classified as class c'. For a multiclass problem with C classes, confusion matrix is $C \times C$

Use loss fns that themselves penalize models that neglect rare classes



$\mathbf{y}$ $\hat{\mathbf{y}}$

$a = \left\lvert \left\{ i : \begin{matrix} \hat{\mathbf{y}}_i = 1 \\ \mathbf{y}_i = 1 \end{matrix} \right\} \right\rvert$	$b = \left\lvert \left\{ i : \begin{matrix} \hat{\mathbf{y}}_i = 1 \\ \mathbf{y}_i = -1 \end{matrix} \right\} \right\rvert$
$c = \left\lvert \left\{ i : \begin{matrix} \hat{\mathbf{y}}_i = -1 \\ \mathbf{y}_i = 1 \end{matrix} \right\} \right\rvert$	$d = \left\lvert \left\{ i : \begin{matrix} \hat{\mathbf{y}}_i = -1 \\ \mathbf{y}_i = -1 \end{matrix} \right\} \right\rvert$

$$r_{\text{acc}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{a+d}{a+b+c+d} = p_+ \cdot r_{\text{TPR}} + p_- \cdot r_{\text{TNR}}$$

Classification Accuracy

$$r_{\text{bal}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{r_{\text{TPR}} + r_{\text{TNR}}}{2}$$

Balanced Accuracy

$$r_{\text{TPR}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{a}{a+c}$$

True Positive Rate (= Recall)

$$r_{\text{TNR}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{d}{b+d}$$

True Negative Rate

$$r_{\text{prec}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{a}{a+b}$$

Precision

$$F(\mathbf{y}, \hat{\mathbf{y}}) = \frac{2 \cdot r_{\text{prec}} \cdot r_{\text{rec}}}{r_{\text{prec}} + r_{\text{rec}}}$$

F-measure

$$r_{\text{G}}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{r_{\text{TPR}} \cdot r_{\text{TNR}}}$$

G-mean

$$r_{\text{Min}}(\mathbf{y}, \hat{\mathbf{y}}) = \min\{r_{\text{TPR}}, r_{\text{TNR}}\}$$

MinPerf

$$r_{\text{rec}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{a}{a+c}$$

Recall

Imbalance-aware Training Techniques

Hinge, logistic loss promote higher classification accuracy

$r_{\text{acc}}(\mathbf{y}, \hat{\mathbf{y}}) = p_+ \cdot r_{\text{TPR}} + p_- \cdot r_{\text{TNR}}$

However, if $p_+ \to 0$ *then* $r_{\text{acc}}(\mathbf{y}, \hat{\mathbf{y}}) \approx p_- \cdot r_{\text{TNR}}$ *which is why these loss functions have no incentive to do well on rare classes*

Even a useless model that predicts everything -ve will get $r_{\text{acc}}(\mathbf{y}, \hat{\mathbf{y}}) \approx p_- \to 1$

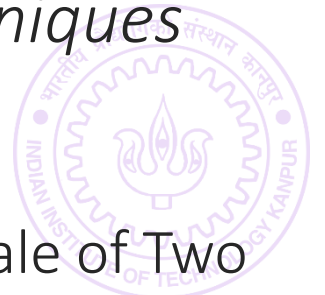
Other reward functions heavily penalize such neglectful behavior

Easy to verify that the above useless model will get heavily penalized since it would have $a = 0$ *i.e.* $r_{\text{TPR}} = 0, r_{\text{prec}} = 0, r_{\text{rec}} = 0$ *which means that it would get* $r_{\text{Min}}(\mathbf{y}, \hat{\mathbf{y}}) = \min\{r_{\text{TPR}}, r_{\text{TNR}}\} = 0$ *and F-measure = 0*

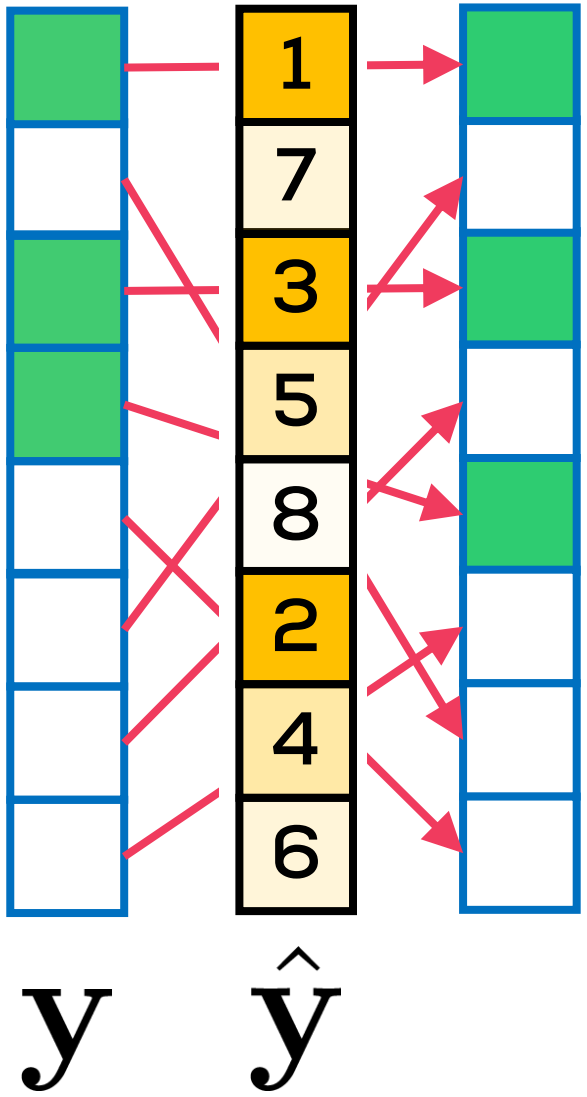
Using these reward functions in training requires advanced opt. techniques

Narasimhan et al. On the Statistical Consistency of Plug-in Classifiers for Non-decomposable Performance Measures, NIPS 2014

Narasimhan et al. Optimizing Non-decomposable Performance Measures: A Tale of Two Classes. ICML 2015



Imbalance-aware Training Techniques



Another technique is to treat classification problems as ranking problems instead

For binary problems, this is done by asking the model to assign scores to data points and try to give high scores to +ve points and low scores to -ve points

Next, data points are ranked according to their scores

Note: here we do not threshold scores at 0 to arrive labels

This ranking is used to assign loss/reward to the model

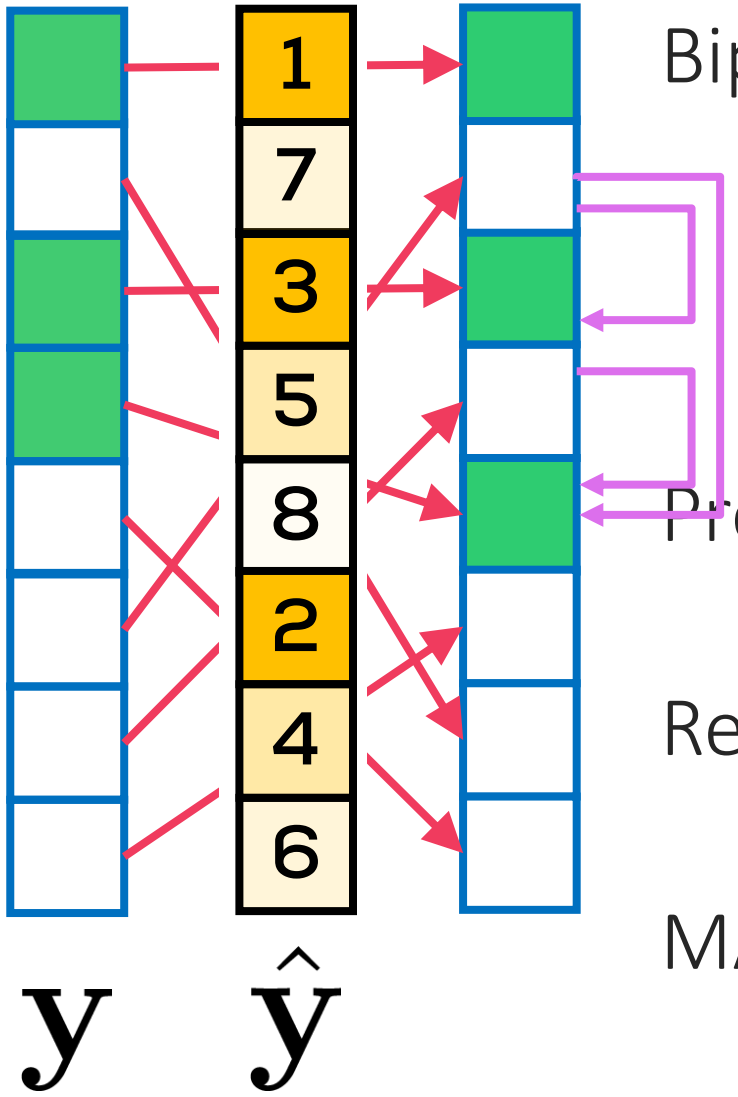
Model is rewarded if lots of positives are in top positions

Model is penalized if negatives pollute the ranking

Such loss fns are key in multiclass/label learning too

For multiclass/label, instead of ranking data points, we ask model to rank classes from most likely to least likely for each data point e.g. rank items for each user

Imbalance-aware Training Techniques



Bipartite Ranking Loss (AUC loss)

Counts # times a -ve point was ranked above a +ve point divided by product of # +ves and # -ves

*In this cartoon this mistake was committed 3 times and #+ves = 3, #-ves = 5 so AUC loss is 3/(3*5) = 0.2*

Precision@k

Counts # +ves present in top k positions divided by k

Recall@k

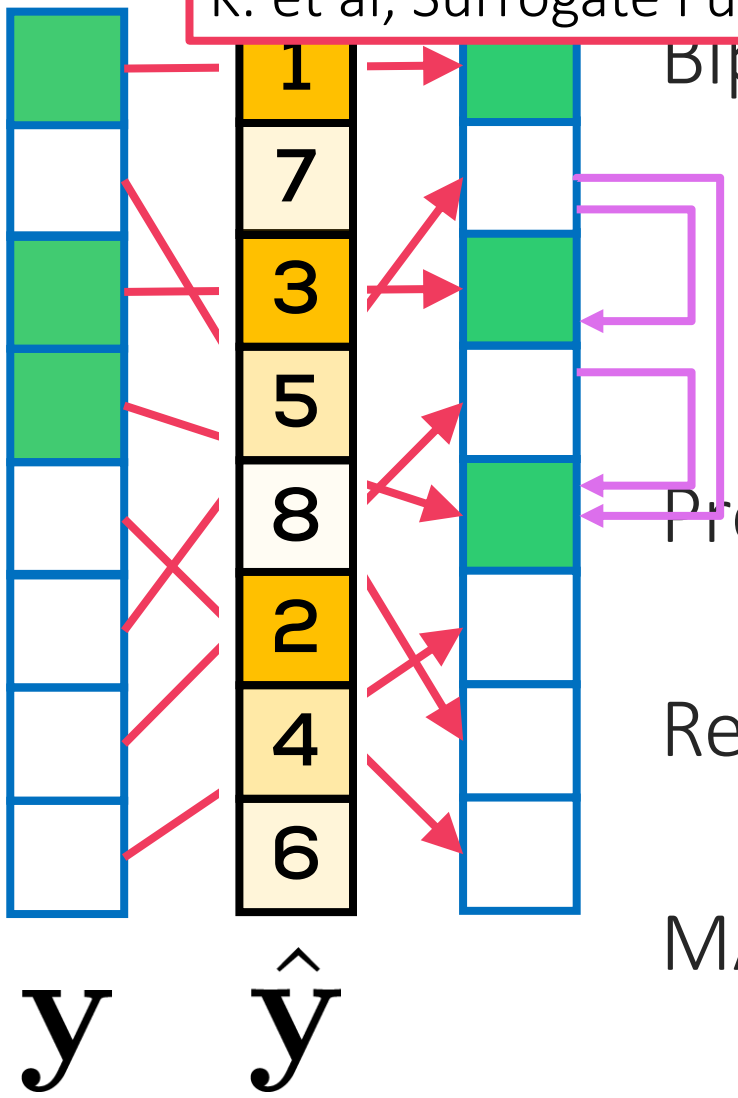
Counts # +ves present in top k positives divided by #+ves

MAP (Mean average Precision)

Finds the average reciprocal rank at which +ves were predicted. In this cartoon this is (1/1+1/3+1/5) = 1.533

Im

Advanced techniques exist to even optimize these on training data.
Zhao et al, Online AUC Maximization, ICML 2011
K. et al, Surrogate Functions for Maximizing Precision at the Top, ICML 2015



Bipartite Ranking Loss (AUC loss)

Counts # times a -ve point was ranked above a +ve point divided by product of # +ves and # -ves

*In this cartoon this mistake was committed 3 times and #+ves = 3, #-ves = 5 so AUC loss is 3/(3*5) = 0.2*

Precision@k

Counts # +ves present in top k positions divided by k

Recall@k

Counts # +ves present in top k positives divided by #+ves

MAP (Mean average Precision)

Finds the average reciprocal rank at which +ves were predicted. In this cartoon this is (1/1+1/3+1/5) = 1.533

Hybrid Learning Strategies

It is folklore in ML that discriminative models do better when lots of data is available but generative models do better with less data

Ng and Jordan, On Discriminative vs. Generative classifiers, NIPS 2001

For popular classes with lots of data, safe to use discriminative techniques like kernels, trees, deep nets along with one-vs-all etc

For rare classes with less data, may try generative techniques like naive Bayes, GMM or even LwP

Hybrid strategies are often used in recommendation systems

Rocchio classifiers, reranking algorithms etc

Use a standard discriminative algorithm e.g. trees etc to rank all labels but then use LwP-style techniques to boost the ranks of rare labels so that they do not get lost

Jain et al. Extreme Multi-label Loss Functions for Recommendation, KDD 2016