Model Evaluation

Semester 1, 2021

Kris Ehinger

Announcements

- Assignment 1 released tomorrow (Friday) night
  - Pose classification using naïve Bayes
  - Due April 12

- New lecturer next week! Ling Luo

Outline

- Selecting test data
- Evaluation methods
- Comparisons: baselines and benchmarks
- Final thoughts

What is a good classifier?

- Supervised classifier learns to map attributes to class labels
- Goal is to generalise: assign correct class labels to instances never seen before
- How to identify a good classifier?
  - **Train** on some training data
  - **Test** on test data (not seen during training)
  - **Evaluate** performance on the test data

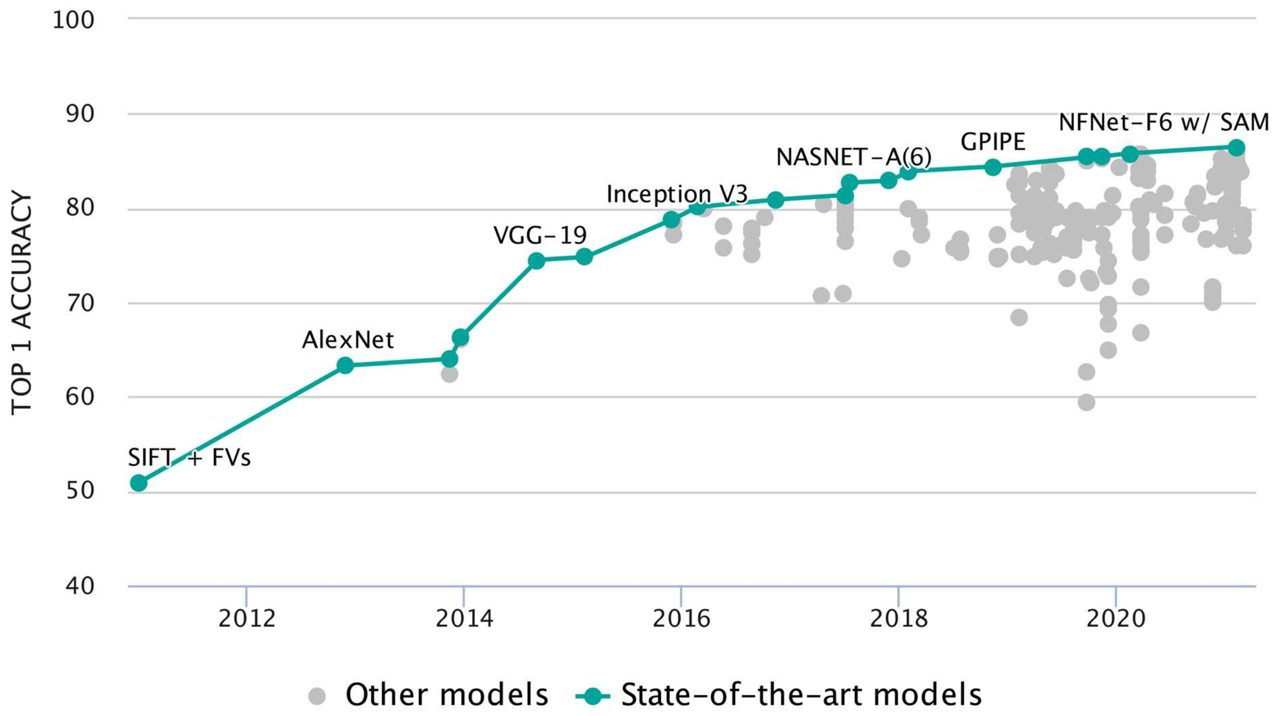
What is a good classifier?

- Basic evaluation metric: Accuracy

$$\text{Accuracy} = \frac{\text{Number of correctly labeled test instances}}{\text{Total number of test instances}}$$

- Measures the percent of time the classifier is correct

Accuracy on ImageNet



Source: https://paperswithcode.com/sota/image-classification-on-imagenet

True or false?

not always the full picture

- If two models have the same accuracy on a test set, they have learned the same thing.

- If two models make the same pattern of errors on a test set, they have learned the same thing.

- A model with higher accuracy on a test set will generalise to novel situations better than a model with lower accuracy.

Selecting test data

Train/test split

- Classifier trains on “training” data and tests on “test” data
- Usually, we just have *data* – a collection of instances
  - How to get “training” and “test” data?

Do we even need “test” data?

- Why not just train on the entire dataset and present that result?

Random holdout

- Randomly **partition** data into "training" or "test"
  - A portion of the data is "held out"; never seen during training
  - Model is tested only on the unseen "holdout" data
- Common splits (train-test): 50-50, 80-20, 90-10
  - Leave-one-out: (N-1) – 1
- Trade-off between having enough data for training and a representative test set

Repeated random subsampling

- Random holdout repeated multiple times:
  - Randomly assign data to "training" and "test" (usually with a fixed split, like 50-50)
  - Train a new model on "training" data
  - Test on the "test" data
- Final evaluation: average over all iterations
- Slower, but result should be more reliable than one random holdout

Cross-validation

- Preferred alternative to repeated random subsampling: **Cross-validation**
- Data is split into m partitions, and iteratively:
  - One partition is held out as a test set
  - The other m-1 partitions are used as training data
- Evaluation metric is aggregated across m partitions
  - Sometimes this means averaging, but more often results are saved for each partition then concatenated
- Every item appears as a test item exactly once

Cross-validation

- How big is m (m-fold cross validation)?
- More folds = fewer test instances / more training instances per partition
- Common choices: m=10 or m=5
  - Mimics 90-10 or 80-20 holdout, but more reliable
- Best choice: **Leave-one-out cross-validation**
  - m = N (number of instances)
  - Maximises training data
  - Far too slow to be used in practice, unless N is tiny

Practical issues

- Should we ensure the proportion of classes is identical in the training vs. test set?
  - Random sampling may produce different proportions
  - **Stratification** or **vertical sampling** – training data and test data both have the same class distribution as the dataset as a whole

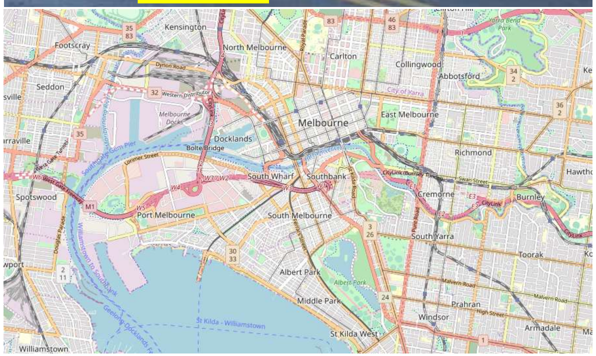
Advantage.
① ensure each subgroup receive proper representation within the group
② better coverage of population

分类筛选 ⇒ ① used to eliminate sampling bias
② allows to create a test set with a population best represents the entire population being studies.

sampling bias: the samples of a variable don't represent the true distribution
⇒ because certain values of the variable are systematically under-represented or over represented

Disadvantage
① can't confidently classify every member into a subgroup
② overlapping can be a issue if subjects are fallen into multiple subgroup (more easy to be chosen).

Case study



Task: An international company is developing a pedestrian detection system for driverless cars.

Dataset: 3 million video frames from videos taken over 5 days driving around Melbourne

Suggestion: train on odd frames, test on even frames

Is this a good suggestion? Or would you try a different train/test split? What other video data would you collect for further tests?

Map: www.openstreetmap.org

Case study

Training instance

Frame 127,405

Test instance

Frame 127,406
0.033 seconds later

Image: cvcl.mit.edu/searchmodels/

Practical issues

- **Inductive Learning Hypothesis:** Any model which approximates the target function well over a large training set will also generalise to unseen examples
- However, machine learners also suffer from **inductive bias** – assumptions made about the data to build the model and make predictions
  - Different assumptions -> different predictions

assumption don't match the real world

Validation set?

- Sometimes data is split into train/validation/test
- Validation set is a "test" set for your training data
  - Used to choose weights or parameters
  - Check if the model converged to a good solution
- Why use a validation set?
  - Why not just train N models and see which is best on test set?
  - Why not just choose parameters on the whole training set?

parameters overfit on training set and can't generalise on test set

Evaluation methods

Evaluation metrics

- Accuracy is a good start, but we'd like to know more about what the model is doing
- Consider a two-class problem where the goal is to find a class of interest ("positive" class) among uninteresting distractors ("negative" class). Examples:
  - Pedestrian (+) vs. not a pedestrian (-)
  - Has a disease (+) vs. does not have the disease (-)
  - Purchased product (+) vs. did not purchase (-)

Types of errors

- Possible classification results:
  - Positive case classified as "positive" (true positive, TP)
  - Positive case classified as "negative" (false negative, FN)
  - Negative case classified as "positive" (false positive, FP)
  - Negative case classified as "negative" (true negative, TP)

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

Accuracy and error rate

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

TP + TN ← **Correct responses**

TP + FP + FN + TN ← All responses

= 1 - Accuracy

$$\text{Error rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

FP + FN ← **Incorrect responses**

TP + FP + FN + TN ← All responses

$$\text{Error rate reduction} = \frac{\text{ER}_0 - \text{ER}}{\text{ER}_0}$$

Change in error rate relative to a base model's error rate

Types of errors

- Are some types of errors more important than others?
- Example: An autonomous vehicle uses a computer vision system to detect pedestrians in the road (positive class = pedestrian)

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

Types of errors

- Example: We’ve developed two machine learning methods to detect a disease. We test each algorithm on a set of 1000 cases (1% of which have the disease). Each model is 99% accurate.

Example: Types of errors

Model 1		Predicted	
		Positive	**Negative**
Actual	**Positive**	10	0
	Negative	10	980

Model 2		Predicted	
		Positive	**Negative**
Actual	**Positive**	0	10
	Negative	0	990

Model 2 just says "negative" to all cases!

Precision and recall

- **Precision:** How often is the model correct, when it predicts a positive case?
- **Recall:** What proportion of the true positive cases in the dataset was the model able to detect?

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Precision and recall

Model 1		Predicted Positive	Predicted Negative
Actual	**Positive**	10	0
	Negative	10	980

Precision

$$\frac{10}{10 + 10} = 0.5$$

Model 2		Predicted Positive	Predicted Negative
Actual	**Positive**	0	10
	Negative	0	990

$$\frac{0}{0 + 0} = undefined$$

(would probably be reported as 0)

Precision and recall

Model 1		Predicted	
		Positive	Negative
Actual	Positive	10	0
	Negative	10	980

Recall

1 0
9999 1

$$\frac{10}{10+0} = 1.0$$

Model 2		Predicted	
		Positive	Negative
Actual	Positive	0	10
	Negative	0	990

$$\frac{0}{0+10} = 0$$

Precision and recall

- Trade-off between precision and recall:

$\frac{tp}{tp+fp} = \frac{1}{1+\frac{fp}{tp}} \downarrow$

$\frac{tp \downarrow}{tp+fn}$

  - High precision + low recall means the model requires a lot of evidence to say "positive"

$1 + \frac{fp 更大}{tp} \uparrow$ $tp \uparrow$

  - Low precision + high recall means the model doesn't need much evidence to say "positive"

tp 小
fp 更小
⇒ require evidence to be positive

- Ideally, we'd like both precision and recall to be high. A popular metric that combines both is the F-score:

$$F_\beta = \frac{(1+\beta^2)PR}{(\beta^2 P) + R} \qquad F_1 = \frac{2PR}{P+R}$$

Sensitivity and specificity

查正

- **Sensitivity:** Another name for recall – the proportion of true positive cases the model was able to detect

查负

- **Specificity:** Proportion of true negative cases that the model was able to detect

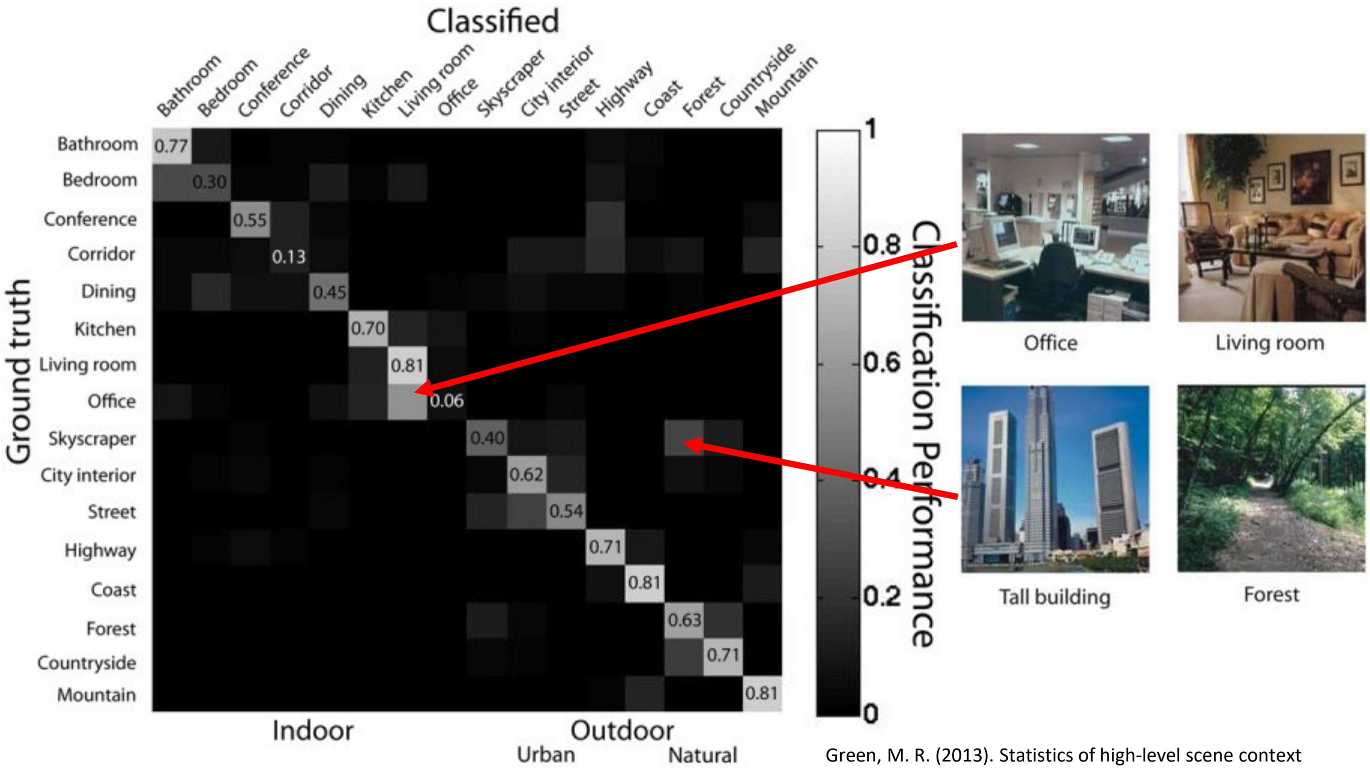
$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Multiclass evaluation

- A **confusion matrix** shows the pattern of errors in a multiclass classification task
- How to compute precision and recall for this type of task?

Confusion matrix



Green, M. R. (2013). Statistics of high-level scene context

Multiclass evaluation

- What if you have more than two classes?
- If you have one class of particular interest, you can evaluate **one-vs.-rest**: treat the one as postive

Actual	Predicted: Pedestrian	Road	Sidewalk	…
Pedestrian	TP	FN	FN	…
Road	FP	TN	TN	…
Sidewalk	FP	TN	TN	…
…	…	…	…	…

Multiclass evaluation

- Usually, you care about all of the classes:

	Predicted			
Actual	**Pedestrian**	**Bus**	**Car**	**...**
Pedestrian	87	4	2	...
Bus	2	34	19	...
Car	1	22	27	...
...	...	...	...	...

Accuracy in one class: number of correct classifications in that row, over sum of that row

Multiclass evaluation

- Total accuracy: sum of diagonal cells (correct classifications) over sum of entire table
- Precision/recall/F-score are computed per class (using one vs. rest, with each class as the “positive” class and everything else as “negative”) and averaged across c classes…

Multiclass evaluation

- Macro-averaging: calculate P,R per class and take mean

$$Precision_M = \frac{\sum_{i=1}^{c} Precision(i)}{c} \qquad Recall_M = \frac{\sum_{i=1}^{c} Recall\ (i)}{c}$$

$$\text{Accuracy} = \frac{\overbrace{\text{TP} + \text{TN}}^{\text{Correct responses}}}{\underbrace{\text{TP} + \text{FP} + \text{FN} + \text{TN}}_{\text{All responses}}}$$

- Micro-averaging: combine all instances into one pool

$$Precision_\mu = \frac{\sum_{i=1}^{c} TP_i}{\sum_{i=1}^{c} TP_i + FP_i} \qquad Recall_\mu = \frac{\sum_{i=1}^{c} TP_i}{\sum_{i=1}^{c} TP_i + FN_i}$$

- Weighted averaging: calculate P,R per class and take mean, weighted by proportion of instances in that class

$$Precision_W = \sum_{i=1}^{c} \left(\frac{n_i}{N}\right) Precision(i) \qquad Recall_W = \sum_{i=1}^{c} \left(\frac{n_i}{N}\right) Recall(i)$$

Comparisons: baselines and benchmarks

Baseline vs. benchmark

- Baseline = simple naïve method that we would expect any machine learning method to beat
  - Example: random guessing
- Benchmark = established rival technique to which we are comparing our method
  - Example: current best-performing algorithm on a leaderboard
- In practice, people aren't strict about the usage of these terms ("baseline" is sometimes used for both)

Common baselines

- Random baseline
  - Guess a class label uniformly from the available labels
  - Guess labels based on class distribution in the training set
- Zero-R baseline
  - Always guess the most common label in the training set
- Other baselines
  - Regression – always guess the mean value
  - Object detection – always guess the middle of the image
  - …

Baseline example

- A regression model predicts outcomes on a scale from 1.0 - 5.0
- Test set error = mean absolute difference between true and predicted label
- Is a model with error = 1.5 good?

	Error
Proposed model	1.5
Baseline: guess a random value between 1-5	
Baseline: guess “3” for every item	

Final thoughts

Model evaluation

- Why evaluate on "test" data? Is there a mathematical way to know which model will generalize the best?
- The only way to guarantee optimal performance on a test set is to know *a priori* what the unseen data will look like
  - "No free lunch" theorems – Wolpert & Macready (1997)

Model evaluation

- How do we know if a model is solving a problem "correctly?" Can we know what a computer is thinking?

Haibe-Kains B., et al. (2020). Transparency and reproducibility in artificial intelligence. Nature, 586(7829), E14-E16.