dm25s1

Topic 08 : Classification1

Part 01 : Overview

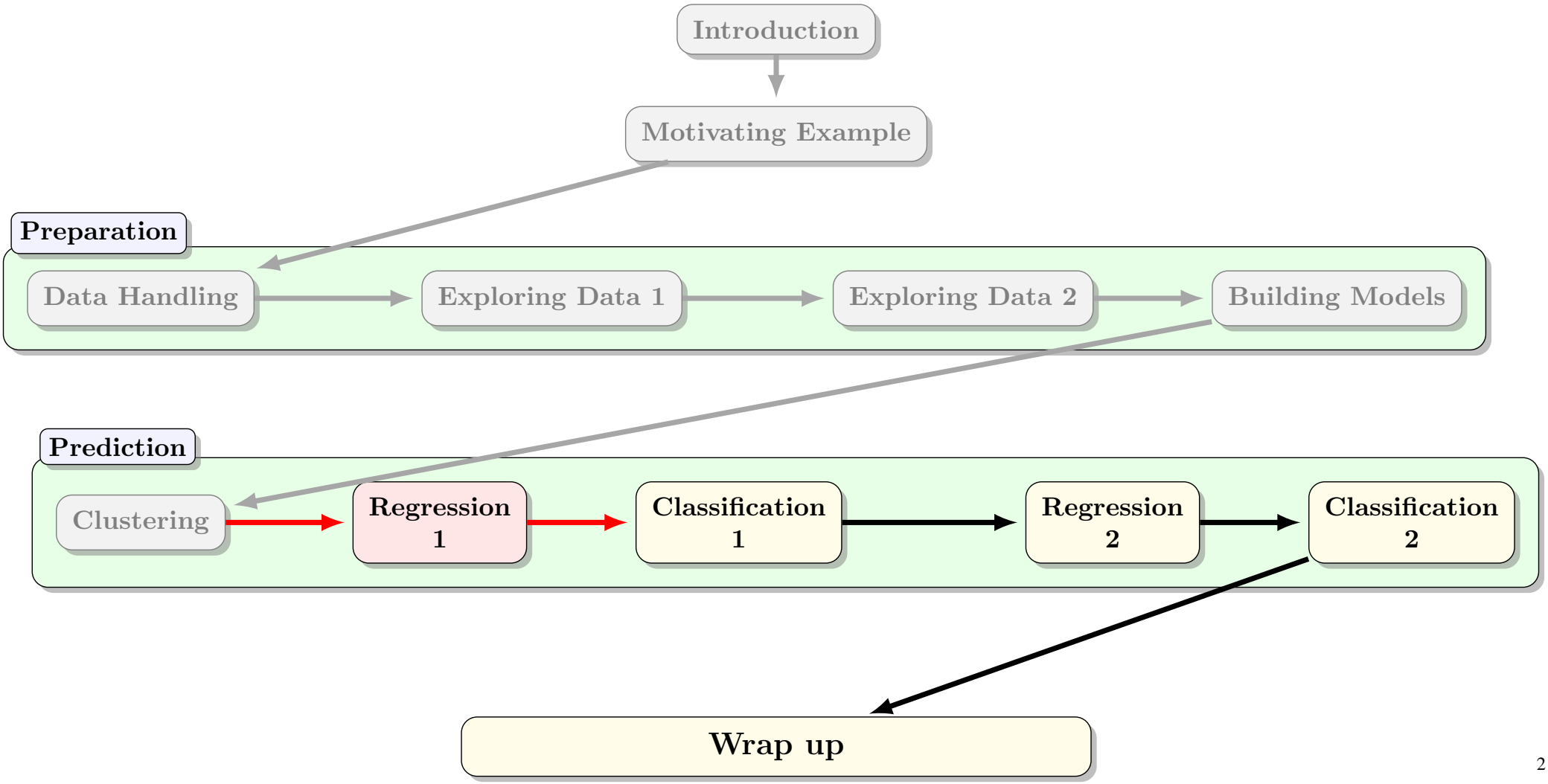
Dr Bernard Butler
Department of Computing and Mathematics, WIT.
(bernard.butler@setu.ie)

Autumn Semester, 2025

Outline

- How classification differs from regression
- Classification metrics

Data Mining (Week 8)



Acknowledgment

Thanks to Dr Kieran Murphy for some of today's slides.

Introduction to Classification

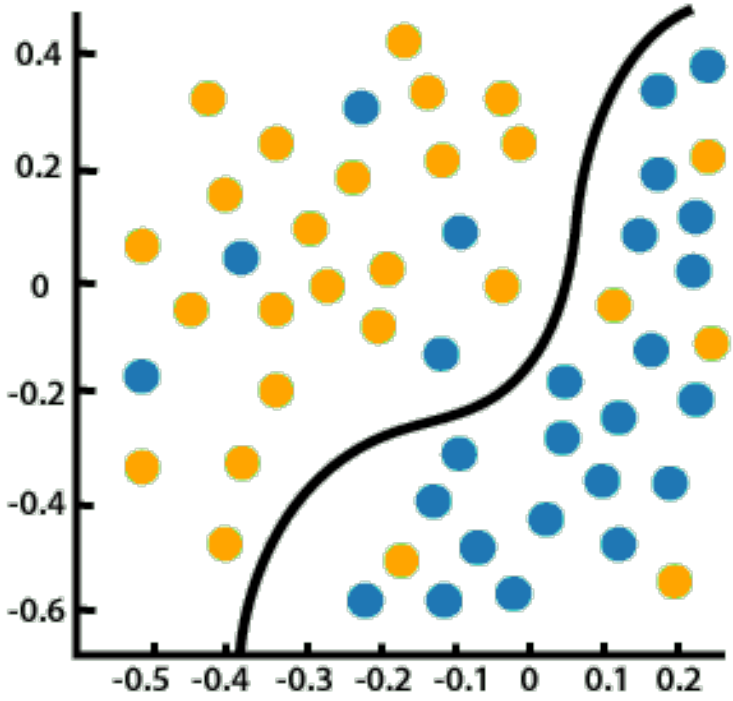
Definition 1 (Classification)

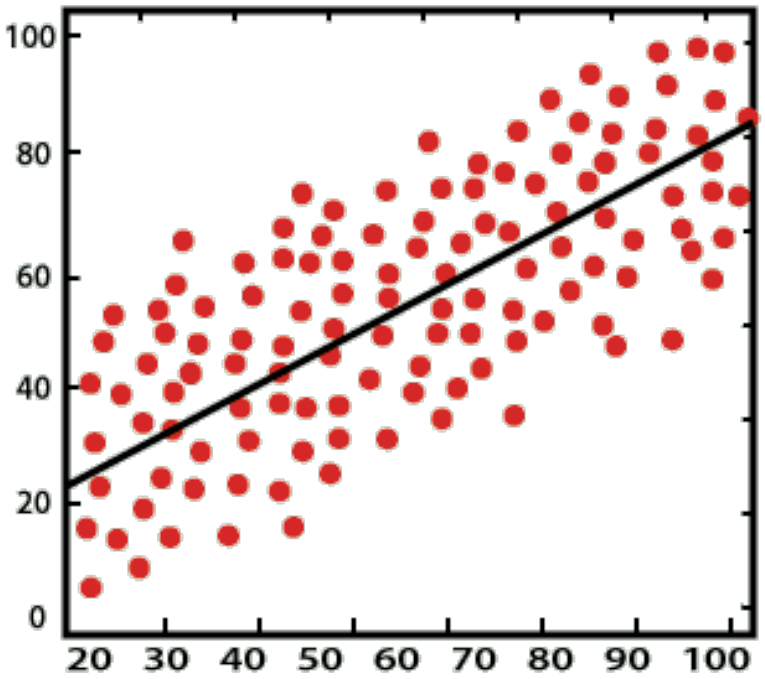
Classification aims to learn a function that takes attribute values and predicts a categorical/qualitative value, such as membership of a class, existence of an effect, etc. The attributes can be categorical or numeric. As with linear regression, classification is an example of supervised learning. It differs from regression because regression predicts a numeric response.

- Some *classifiers* generate class membership probabilities (numeric) en route to predicting class membership (of the most likely class), so the distinction is not always clear-cut.
- There is essentially one regression algorithm (with many variants/enhancements/implementations) but there are *many* classification algorithms.
- You have seen 1 already (KNN) and we introduce another algorithm today.

Classification vs Regression

Supervised data models have a target.
If target is quantitative (continuous) then have a regression model, if categorical then classification model.





Classification models aim to:
- predict class/label for each new observation,
- define a decision boundary between classes,
- and possibly the probability of being in each class.

Regression models aim to
- predict a continuous value for each new observation.

Classification vs Regression

- Unlike regression, statistical distributions play a limited role in evaluating a classifier:
  - Scope for hypothesis testing is limited (there is no equivalent of the `statsmodels` diagnostic output (covered in topic 7).
  - Rely on empirical metrics — accuracy, precision, recall, f1-score, auc, ...
- Classification metrics tend to be easier to use/understand than those in regression — classification metrics are based on counts of correct (or incorrect) cases divided by a subset of cases.
- Central concept in classification model is the confusion matrix:

Actual \ Predicted	Negative	Positive	
Negative	✔ True Negative (*TN*)	Type I error False Positive (*FP*)	N
Positive	Type II error False Negative (*FN*)	✔ True Positive (*TP*)	P
	$\hat{N}$	$\hat{P}$	T

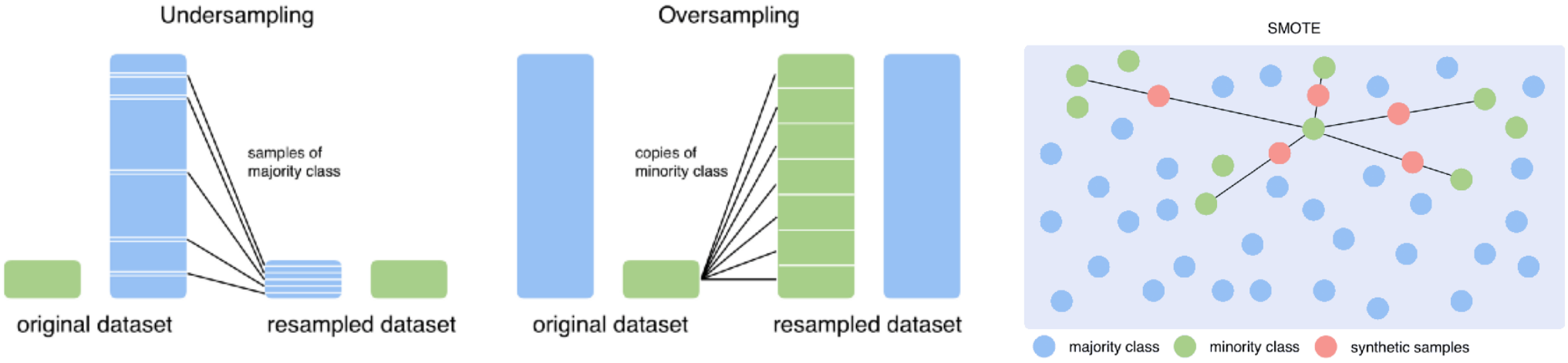
Unbalanced Classification Datasets

Practical classification datasets are often unbalanced — where the frequency of the classes in the target are very uneven:

- Telecommunication customer churn datasets. Churn rate of 2%–10%.
- Credit Card Fraud Detection 0.172% (492 frauds / 284,807 transactions).
- National Institutes of Health Chest X-Ray Dataset 14 cases, (size 13 to 3,044) in 5,606 cases

Solutions

Use suitable metrics and/or



Summary of Classification Models

Model	Data Pre-processing*		Impact from		Summary
	Normalisation	Scaling	Collinearity	Outliers	
KNN	✔	✔	✔	✘	• Local approximation, lazy learner • Heavy computational requirements
Logistic Regression	✔	✘	✔	✔	• Descriptive with good accuracy • Reasonable computational requirements
Naïve Bayes	NA	NA	✔	✘	• Works with categorical features only • Suitable for small train datasets
Decision Tree	✘	✘	✔	✔	• Easy to setup and interpret (XAI). • Robust to missing data but not to noise. • Slow training for larger sets.
Random Forest (Not this module)	✘	✘	✘	✘	• High prediction accuracy • Limited explainability • Works with both continuous and categorical features
Support Vector Classifier (Not this module)	✘	✘	✘/✔	✔	• High prediction accuracy • Explainability depends on kernel • Computational effort depends on kernel
Neural Networks (Not this module)	✘	✔	✔	✔	• High prediction accuracy • Self-extract features • Heavy computational requirements

*Use `StandardScaler`, or `RobustScaler` if have outliers.

Lazy vs Eager Learners

Lazy learner

Stores training data (or only minor processing) and uses this to compute prediction when given test data.

- Does not generalise until after training
- Does not produce a standalone model
- Training data must be kept for prediction
- Local approximations
- Often based on *search*
- If new data is just added to the training data, it can respond more easily to changing conditions

Eager learner

Builds a model from the train set, before receiving new data for prediction

- Training has an extra goal: to generalise from the data
- Training has an extra output: standalone model
- Training data can be discarded after use
- Local and/or global approximations
- Based on *computation*
- Models *drift* with time, so not suited to highly dynamic contexts, as it needs retraining

Usually an (eager) model requires much less memory than a (lazy) training set.

A Non-perfect Test — Type I and Type II Errors

Consider an imperfect test with two outcomes, there are four possible outcomes:

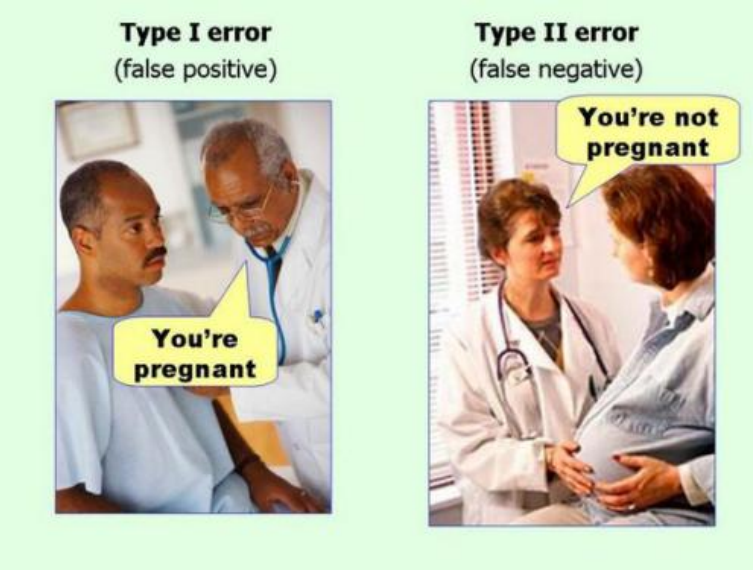
Confusion Matrix

		Predicted: Negative	Predicted: Positive	
Actual	Negative	✔ True Negative (TN)	Type I error False Positive (FP)	N
	Positive	Type II error False Negative (FN)	✔ True Positive (TP)	P
		$\hat{N}$	$\hat{P}$	T

- If the test is applied to $T = P + N = \hat{P} + \hat{N}$ observations / subjects / instances then we have four independent quantities TP, TN, FP, and FN.
- How do we combine these quantities into a single metric ?
- The fraction of correct results seems like a good idea

$$\text{accuracy} = \frac{TP + TN}{P + N}$$

But what happens, if we are testing for an rare event? Maximising accuracy will result in the test always returning negative.



- Ideally we want the probability of either error to be zero but that may not be possible.
- Depending on the conditions we often modify the test to reduce probability of the type of error we don't want at the expense of increasing the probability of the other — think court case vs medical condition.

Confusion matrix (Contingency table) Metrics

Accuracy — how well model is trained and performs in general

$$\textbf{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

(How often is the classifier correct?)

Actual \ Predicted	Negative	Positive	
Negative	✔ True Negative (*TN*)	Type I error False Positive (*FP*)	*N*
Positive	Type II error False Negative (*FN*)	✔ True Positive (*TP*)	*P*
	$\hat{N}$	$\hat{P}$	*T*

- False negative rate (FNR) $= \frac{FN}{\text{P}} = 1 - TPR$
- **Sensitivity** = **Recall** = True positive rate (TPR) $= \frac{TP}{\text{P}} = 1 - FNR$

  (Of positive cases that exist how many did we mark positive?)

Recall — important when the costs of false negatives are high

- **Specificity** $= \frac{\text{TN}}{\text{N}} = 1 - \text{FPR}$

  (When it's actually no, how often do we predict no?)

  (Of cases that are negative, how many did we mark negative?)
- False positive rate (FPR) = false acceptance $= \frac{FP}{\text{N}} = 1 - \text{Specificity}$
- **Precision** = positive predictive value (PPV) $= \frac{TP}{\hat{\text{P}}} = \frac{TP}{\text{TP} + \text{FP}}$

  (Of cases that we marked positive, how many were correct?)

Precision — important when the costs of false positives are high

Confusion matrix (Contingency table) Metrics II

F_1 Score

The F-measure or balanced F-score (F_1 score) is the harmonic mean of precision and recall:

$$F_1 = 2\left[\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}\right] = 2\left[\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right]$$

A word of Caution ...

Consider the three binary classifiers A, B and C

	A		B		C	
	T	F	T	F	T	F
T	0.9	0.1	0.8	0	0.78	0
F	0	0	0.1	0.1	0.12	0.1

Metric	**A**	**B**	**C**	(best)
Accuracy	0.9	0.9	0.88	**A,B**
Precision	0.9	1.0	1.0	**B,C**
Recall	1.0	0.888	0.8667	**A**
F-score	0.947	0.941	0.9286	**A**

Clearly classifier *A* is useless since it always predicts label T regardless of the input. Also, B is slightly better than C (lower off- diagonal total). Yet look at the performance metrics – *B is never the clear winner, and A has the best Accuracy, Recall and F_1-score, even though it just predicts the majority class! You can see why imbalanced data causes problems...*

We use some metrics because they are easy to understand, and not because they always give the "correct" result.

Mutual Information is a Better Metric

The mutual information between predicted and actual label (case) is defined

$$I(\hat{y}, y) = \sum_{\hat{y}=\{0,1\}} \sum_{y=\{0,1\}} p(\hat{y}, y) \log \frac{p(\hat{y}, y)}{p(\hat{y})p(y)}$$

where $p(\hat{y}, y)$ is the joint probability distribution function.
This gives the intuitively correct rankings $B > C > A$

Metric	**A**	**B**	**C**	**(best)**
Accuracy	0.9	0.9	0.88	A,B
Precision	0.9	1.0	1.0	B,C
Recall	1.0	0.888	0.8667	A
F-score	0.947	0.941	0.929	A
Mutual information	**0**	**0.1865**	**0.1735**	**B**

Cross Entropy Loss

Cross-entropy loss (also known as log loss) is the classification equivalent of RMS Error

Definition 2 (Cross Entropy Loss)

If the classifier returns a *probability* estimate of the class membership, we can compute a metric from this directly, without calculating the confusion matrix first.
Let m be the number of cases in the training set, $\{y_i, i = 1, \ldots, m\}$ be the *actual* target values of the training set, $\{\hat{y}_i, i = 1, \ldots, m\}$ be the *predicted* target values of the training set, C be the number of possible class values, $\{c_k, k = 1, \ldots, C\}$ be those class values, $\{w_{i,k} = 1$ if $\hat{y}_i (\equiv \max_k p_{i,k}) = y_i$ and is 0 otherwise, and $0 \leq p_{i,k} \leq 1$ be the probability predicted by the model that training case i takes class value c_k. Then

$$L_{CE}(\boldsymbol{X}, \boldsymbol{y}, f, \boldsymbol{a}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{C} w_{i,k} p_{i,k} \geq 0 \tag{1}$$

The cross entropy loss is the scaled sum of the log probabilities assigned to incorrect predictions.

Binary Cross Entropy Loss

Definition 3 (Cross Entropy Loss)

In the special case of *binary* classification, $C \equiv 2$ and $p_{i,2} \equiv 1 - p_{i,1}$ (because $p_{i,1} + p_{i,2} = 1$), so we can drop the second subscript for p and just use p_i and $(1 - p_i)$. In this case

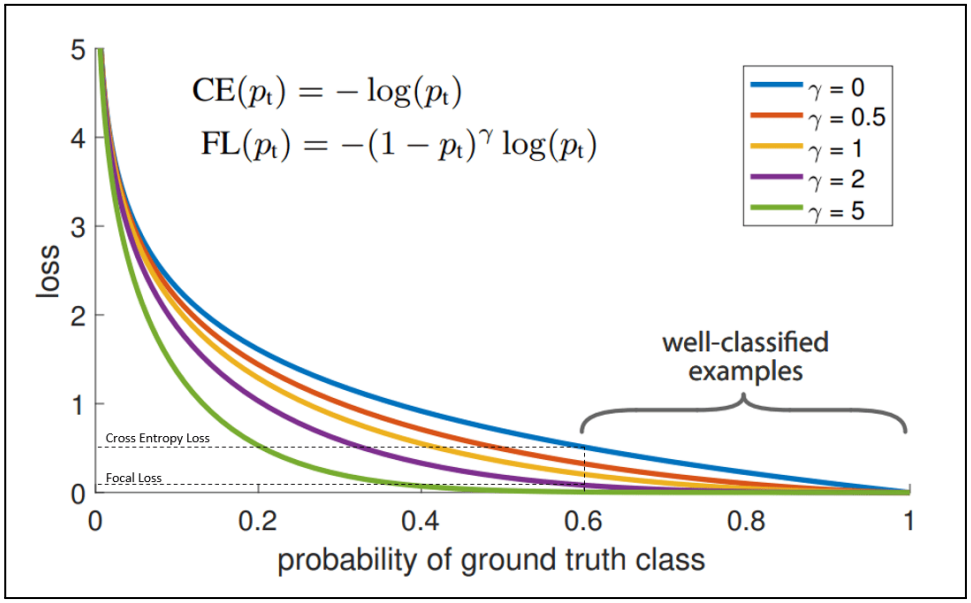
$$L_{CE}(\boldsymbol{X}, \boldsymbol{y}, f, \boldsymbol{a}) = -\frac{1}{m} \sum_{i=1}^{m} \left(w_{i,1} p_i + w_{i,2}(1 - p_i) \right) \geq 0 \tag{2}$$

(Binary) cross entropy loss is often chosen as the loss function for deep learning classifiers.

Focal Loss

See What is Focal Loss and when should you use it? for discussion and motivation

- Cross entropy loss can be over confident because $\log(p)$ falls so gently as p increases.
- This makes it more sensitive to outliers.
- It is also more likely to choose the majority class when the data is unbalanced.
- *Focal loss* was invented by the Computer Vision community to solve this problem.
- It weights the log term (with hyperparameters α and γ) so the weighted term falls more rapidly.



α is just a multiplier, and is implicitly $\alpha = 1$ above.

The focal loss α and γ hyperparameters can be tuned like any other hyperparameters.

Micro Average vs Macro Average Performance

In a multi-class classifier we have more than two classes.

To combine the metrics for individual classes to get an overall system metrics, we can apply either

Micro-Average Method `weighted avg`

Sum up the individual true positives, false positives, and false negatives of the system for different classes and then apply totals to get the statistics.

Macro-average Method `macro avg`

Average the precision and recall of the system on different classes.

	precision	recall	f1-score	support
setosa	1.00	0.95	0.97	19
versicolor	0.81	0.74	0.77	23
virginica	0.71	0.83	0.77	18
accuracy			0.83	60
macro avg	0.84	0.84	0.84	60
weighted avg	0.84	0.83	0.84	60