

BCB726

Lecture 1

October 20, 2025

Welcome and Logistics

- 10 meetings here in MEJ 3116 on Mondays and Wednesdays from 11-12:15pm until Monday November 19
- Lectures will be taught primarily by Natalie, Alex, and Dhuvi (your TA!)
- Class notes and schedule on our course page https://github.com/natalies-teaching/BCB726_Fall2025 and we will also use canvas for homework submissions and discussions
- Components of final grade.
 - Two homework assignments (20% each, 40% in total). These will be practical and focused on implementation
 - 60 % will be based on class attendance. You must attend 8/10 lectures. If you need to miss class please communicate with me as soon as possible

Topics and what to expect

- First half of course → more classical topics and ML fundamentals
 - classical unsupervised and supervised learning
 - components of the optimization problems happening under the hood
 - practical considerations (e.g. cross-validation, bootstrapping, data leakage) to keep in mind when training your models to facilitate robust scientific discovery.
- Second half of course → more modern ML topics
 - Deep learning basics, explainability, large language models
- Implementation: we will use standard libraries such as scikit-learn and pytorch. You will use modal notebooks for GPU computing.

Resources

- Office hours with Dhuvi (time TBD). Dhuvi's email → `dkarthikeyan1@unc.edu`
- Textbooks/ free resources listed on our course page
 - Murphy book (all encompassing, CS-y, practical, and implementation focused) : `https://probml.github.io/pml-book/book1.html`

Some motivating applications of ML in biomedicine....

Creating diagnostic predictors of disease

Ideally, we want to train models of disease status based on samples from donors in a training or *discovery cohort* and evaluate the model on a completely independent validation (v) cohort. A model that performs well on an independent cohort suggests having discovered a diagnostic for future studies.

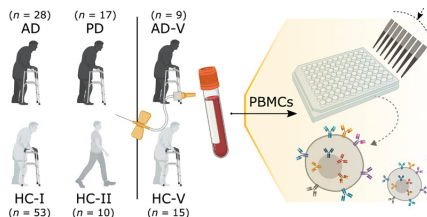


Figure: from Phongpreecha *et al.* Science Advances. 2020

ML approaches can be used to make hypotheses about biological mechanisms

A trained model can be used to evaluate how cell-type specific contributions to a donor's phenotype. This example is related to aging and therefore highlights cell-types that are likely to have a pro-aging effect.

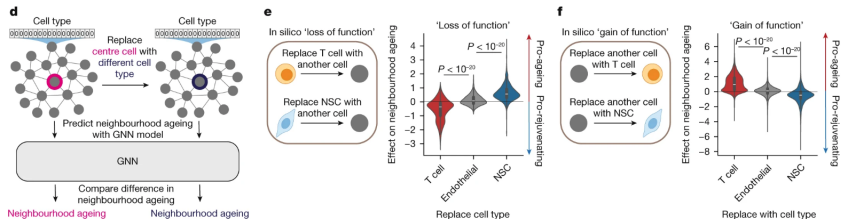


Figure: from Sun *et al.* Nature. 2025.

Vocabulary for discussing machine learning problems

- Data matrix (**X**) is an array of N **data instances** \times d **features** measured about these data instances. We often say this matrix, $\mathbf{X} \in \mathbb{R}^{N \times d}$
 - Example : If we are measuring 20K (features) in 100 mice (data instances), this would produce a matrix $\mathbf{X} \in \mathbb{R}^{100 \times 20K}$
- Response variable or target variable, $\mathbf{Y} \in \mathbb{R}^{N \times 1}$ is the vector of labels for each of our N data instances
 - $\mathbf{Y} \in \mathbb{R}^{100 \times 1}$ could be the ages of each of the 100 mice.

Vocabulary for discussing ML problems (continued)

- Parameters : This is the collection of learnable values that help to translate the *input features* to the target variable.
 - Example : regression coefficients (β), which provide a weight for each input feature. When you fit a regression model, you are trying to learn an appropriate β so that $\mathbf{X}\beta$ approximates \mathbf{Y} as closely as possible.
- Hyperparameters : additional parameters that dictate how to set up the model (e.g. a regularization parameter)
- **Training** refers to optimizing the parameters (e.g. specifying the model) on a collection of data instances.
- Inference or Prediction: Can the model be used in a useful way after it has been trained on a collection of unseen data instances?

Types of learning

- **Supervised** : Leverage labels of data instances to learn parameters. We are training by showing several instances of a predicted data input and what the output label is.
 - diagnostic test trained to predict treatment response based on an omic modality from many donors
 - training a model to predict protein structure from sequence
 - training a model to predict age based on methylation signatures in the blood.
- **Unsupervised** : Looking for patterns across data instances based on features. Usually involves clustering or dimensionality reduction.
 - Clustering cells based on gene or protein expression patterns to identify cell-types
 - Dimensionality reduction (e.g. PCA) of donors based on a large number of measured omics features to identify subgroups of donors. Then investigate sub-groups to find drivers of variation.

Regression as another supervised learning example

Models of age were trained with supervision (e.g. using ages of donors) from single-cell measurements. The model can be used to predict the age in a new donor.

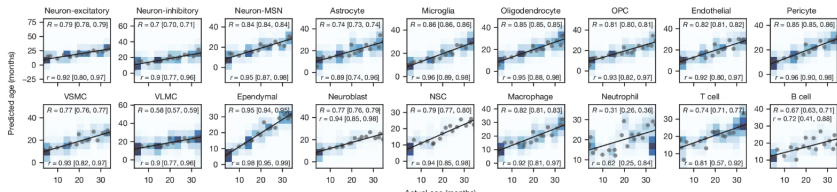
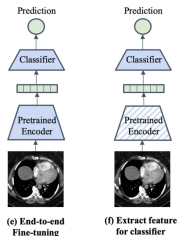


Figure: from Sun *et al.* 2025. Shown here are predicted ages under cell-type specific *aging clocks*, or machine learning models of age.

How do we create features about our data instances

- **Feature engineering.** We define or measure features about each data instance that we suspect are important for differentiating classes in the data
 - Canonical example : omics features measured for each data instance.
 - Summaries of the data : for example, counts across cell-types.
- **Feature learning:** Learning representations of the data.
 - Here, features represent mathematically abstract dimensions that were trained to be helpful for some task, such as, classification



The simplest way to label data : k -nearest neighbors classification

If we have an unlabeled point, x , we can simply classify x , according to the labels of its k nearest neighbors. The higher k is, the less of a possibility we have of making a prediction based on noise.

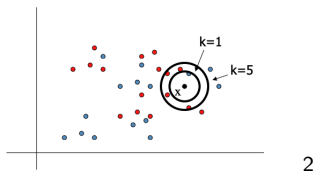


Figure: In this example, we see that we would make different predictions for the black point, depending on if we had used 1 or 5 nearest neighbors.

²from

<https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture10.pdf>

Advantages and disadvantages of this simple approach

- If the data are well separated, then a kNN classifier should do a great job.
- The query time could be expensive.
 - We need a distance metric to quantify the distance of a point in the high-dimensional space to every other point.
 - Then, we need to use these distances to determine the k nearest neighbors.
 - What k should we use?
- This leads us to consider more sophisticated approaches that can be used to learn how to classify or label points in the data.

Mapping data input to label

- Classifiers and regression models specify a set of rules of take input data and map it to a label
- Consider a simple linear classifier, f as, $f(x, W) = Wx + b$
 - x is the data instance
 - w is the weights for the map

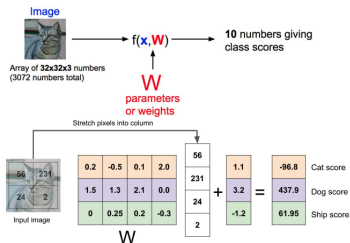


Figure: This example aims to classify images into one of 10 objects- only 3 shown here for example ³

³from https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf.

Training and evaluating the model

- What happens if we train and test the model on the same data?
- The model would effectively memorize the data , but not generalize.
- **Overfitting** is when a complex model with many parameters effectively memorizes the data.

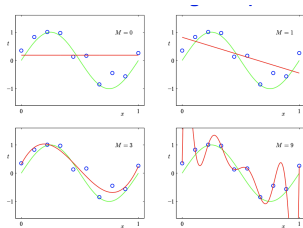


Figure: Higher degree polynomials have increased capacity to overfit the data. ⁴

⁴from

<https://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture02.pdf>

Adopting a proper training/testing procedure

- In an ideal setting with enough data points, we would like to divide the data into three disjoint components :
 - **Training:** A set of data points that will be used to learn model parameters.
 - **Validation:** A subset of the training set that can be used to select hyperparameters
 - **Testing:** A never before seen set of data points used to quantify how well the model generalizes and can predict in a new setting.
- For a **classification** task where we are predicting discrete labels, the simplest *metric of success* is **accuracy**, or the fraction of **testing** data points with a correctly predicted label.

What happens across different train/test splits?

Different train/test splits can cause variability in the trained models and lead to varying test-set accuracies. Hence, good to *bootstrap* and compute a mean or median across train/test splits or cross validation folds (more on that later!).

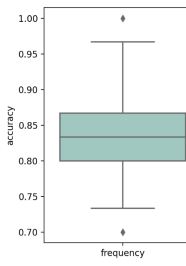


Figure: Each point in the boxplot is the accuracy obtained in an individual train/test split.

Cross validation - common for small datasets in biology

- Divide the dataset into K chunks. Train model K times. In the i -th iteration, train with all of the data except points in the i -th fold and test on data points in the i -th fold.
- Report mean accuracy across folds.

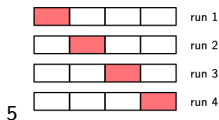


Figure: Colored instances representing testing chunks of the data across 4 cross-validation folds.

⁵from Pattern recognition and machine learning textbook (Bishop)

Leave-one-out cross validation (LOOCV)

- Realistically in biology, we often have super small datasets (< 30 data points)
- We can make as many folds as there are datapoints. So, for N datapoints, we would essentially be doing N fold cross validation.
- For data point, i , train the model with all N datapoints, *except* i . Test the model on datapoint i .
- Evaluate accuracy as the fraction of datapoints with the correctly predicted label from the iteration where they were in the test set.

Loss functions

- **Loss function:** A quantity that we define, which reflects how well the current model explains explains the labels (either discrete or continuous) on our training data.
- The loss function implements 'book-keeping' for quantifying how much predictions of training images violate their true labels




				
cat	3.2	1.3	2.2	
car	5.1	4.9	2.5	
frog	-1.7	2.0	-3.1	6

Figure: How do we quantify how good these predictions are? Each number in the matrix the output given by the linear model for $f(x, W) = Wx + b$.

⁶from https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf.

Loss function notation

- Consider your dataset of data points and labels $\{(x_i, y_i)\}_{i=1}^N$
 - x_i is image i , y_i is the label for image i .
- The loss function for the linear model in our example is computed across all data points as, $L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$
- **Mean squared error loss is common for regression when we are predicting continuous outputs:** $\frac{1}{N} \sum_{i=1}^N (y_i - [Wx_i + b])^2$

Coming back to our example...

Since this is classification, we need to compare the values predicted by $f(x, W) = Wx + b$.




				
cat	3.2	1.3	2.2	
car	5.1	4.9	2.5	
frog	-1.7	2.0	-3.1	7

Figure: Each number in the matrix the output given by the linear model for $f(x, W) = Wx + b$.

⁷from https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf.

Hinge loss

- Consider our example of the 3 images and their predicted scores across cat, car, and frog classes
- We will define $s = f(x_i, W)$ as the model evaluated for image x_i under weights, W .

Then the loss is,

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Intuition of what this is doing...

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases} \\ &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \end{aligned}$$

- Since we are summing over instances where $j \neq y_i$, then we are incurring a penalty if the predicted score for image i of being in class j , (s_j) is larger than the score for its actual class label, (s_{y_i}).
- Otherwise, we add 0 if the score for the predicted class (s_{y_i}) is larger than for the class j . In a perfect world, where our classifier predicted correctly, we would have a 0 for the evaluated loss function.

Computing the loss for the cat image



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9		

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$
$$\begin{aligned} &= \max(0, 5.1 - 3.2 + 1) \\ &\quad + \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$

8

Finding the best model parameters via the loss function

- Let $E(\mathbf{w})$ be the error function evaluated for a particular set of parameters, \mathbf{w} .
- Ideally, we want to walk in a productive direction so that $\nabla E(\mathbf{w}) = 0$. So, we make small steps in the direction of $-\nabla E(\mathbf{w})$.

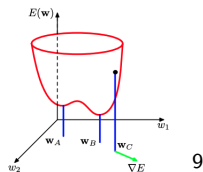


Figure: Geometric view of loss function, $E(\mathbf{w})$. w_A is a local minimum, w_B is a global minimum. At a point, W_C the gradient is given by $\nabla E(\mathbf{w})$.

Recap and next time

- Today → training/testing, cross validation, feature engineering vs feature learning, loss functions,
- Next time → statistics vs ML, optimization for ML, introduction to Modal notebooks.