Contents

1 Mathematical preliminaries

1.1 Calculus

- A function $f(x)$ is a convex function iff for every pair of points on $f(x)$ the line joining them is not below the curve $f(x)$ at any point between them
- $\boldsymbol{\nabla} f(x) =$ vector of partial derivatives $= (\frac{\partial f(x)}{\partial x_1}, ..., \frac{\partial f(x)}{\partial x_n}) =$ grad $f(x)$
- Single-variable chain rule: Let $f(x) = g(h(t))$, then $\frac{\mathrm{d}}{\mathrm{d}t}(f(x)) = \frac{\mathrm{d}g}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}t} = g'(h(t))h'(t)$.
  Thus, $f(x) = f(h(t)) \Rightarrow \frac{\mathrm{d}}{\mathrm{d}t}(f(x)) = \frac{\mathrm{d}f}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}t} = f'(h(t))h'(t)$
- Multi-variable chain rule: Let $f(x) = f(h(t)) = f(v_1(t), ..., v_n(t))$ be a multi-variable function with input dimension n and scalar output, then $\frac{\mathrm{d}}{\mathrm{d}t}(f(x)) = \frac{\partial f}{\partial v_1}\frac{\mathrm{d}v_1}{\mathrm{d}t} + ... + \frac{\partial f}{\partial v_n}\frac{\mathrm{d}v_n}{\mathrm{d}t} = \boldsymbol{\nabla} f(h(t)) \cdot h'(t)$

- Gradient descent: $x_{t+1} = x_t - \alpha \nabla f(x)$ until x stops changing (or equivalently $\nabla f(x)$ converges to 0) finds an x for which $f(x)$ is at a local minima
  - $f(x)$ is convex $\Rightarrow$ has a single local minimum $\Rightarrow$ local minimum is a global minimum $\Rightarrow$ gradient descent finds global minimum
- Lagrange multiplier method for converting a constrained optimization problem into a unconstrained optimization problem:
  1. Make all constraints equality constraints (e.g. introduce slack variables)
  2. Rewrite all constraints to be equal to 0
  3. For each constraint i introduce a new variable the Lagrange multiplier α_i

4. Let $g_i(x) = 0$ be the i^{th} constraint. Subtract $\alpha_i g_i(x)$ from the objective function (we could equivalently add this and the solution would just have α_i with opposite sign)
5. max/min the resulting objective function $f(x, \alpha)$ by solving $\boldsymbol{\nabla} f(x, \alpha) = \vec{\mathbf{0}}$ — can solve analytically (if possible), or use gradient descent to approximate a solution (if not)
   - All solutions to the original problem will be given by this but this may also give extraneous (that is non-optimal) solutions so it is necessary to evaluate each candidate solution with the original objective function

1.2 Linear algebra

- A minor of a matrix is the determinant of a sub-matrix
- A minor of a matrix A is a principal minor iff all (note there are possibly none) of the deletions were of rows and columns with the same index
- The leading principal minor of order k is the one obtained by keeping the first k rows and columns
- A symmetric matrix is positive definite iff all its leading principal minors are strictly positive
- A symmetric matrix is positive semi-definite iff all its principal minors are non-negative

- A vector norm is a function from a vector space to the non-negative reals that behaves like the Pythagorean distance function:
  - $L_0(x_1, ..., x_n) = \#(\text{components that are non-zero}$
  - $L_1(x_1, ..., x_n) = |x_1| + ... + |x_n|$
  - $L_2(x_1, ..., x_n) = \sqrt{|x_1|^2 + ... + |x_n|^2}$
  - $L_m(x_1, ..., x_n) = (|x_1|^m + ... + |x_n|^m)^{\frac{1}{m}} = \|x\|_m$
  - $L_\infty(x_1, ..., x_n) = \max\{|x_1|, ..., |x_n|\}$

1.3 Statistics

- Covariance is a measure of the strength (and direction) of linear relationship between two variables
- $\text{cov}(x, y) = \text{E}[(x - \text{E}[x])(y - \text{E}[y])](= \text{E}[xy] - \text{E}[x]\text{E}[y])$
- Deduce that $\text{var}(x) = \text{cov}(x, x)$

- The Pearson correlation coefficient ρ from A-Level is the normalised (to range from -1 to $+1$) covariance ($\rho_{x,y} = \frac{\mathrm{cov}(x,y)}{\sigma_x \sigma_y}$)

2 Model evaluation

2.1 Evaluation procedure

- Validation data can be used for hyperparameter tuning but testing data should not be used in any way during training
- The test data prevents us overfitting/p-hacking the hyperparameters
- k-fold cross validation splits the data into k disjoint folds of approximately equal size and trains k models (each with a different fold as the test data) to get k measurements of a metric (measured over the current test fold each time) so the mean and standard deviation can be calculated

2.2 Metrics

2.2.1 Primitives

- (Binary) accuracy $= \frac{\#(\text{true positives})+\#(\text{true negatives})}{\#(\text{tests}) \ (=\#(\text{true positives})+\#(\text{true negatives})+\#(\text{false positives})+\#(\text{false negatives}))}$
- Problems with binary accuracy:
  - Is the data set really balanced?
  - Are false positives and false negatives really equally bad?
  - Is a classifier giving $+0.1$ instead of a negative number really as bad as giving $+1000$?

- Recall = sensitivity = true positive rate = $\frac{\#(\text{true positives})}{\#(\text{positive examples}) \ (=\#(\text{true positives})+\#(\text{false negatives}))} =$
  $\Pr(\text{Positive test result}|\text{Positive example})$

- Specificity = true negative rate = $\frac{\#(\text{true negatives})}{\#(\text{negative examples}) \ (=\#(\text{true negatives})+\#(\text{false positives}))} =$
  $\Pr(\text{Negative test result}|\text{Negative example})$

- Precision = positive predictive value = $\frac{\#(\text{true positives})}{\#(\text{test positives}) \ (=\#(\text{true positives})+\#(\text{false positives}))} =$
  $\Pr(\text{Positive example}|\text{Positive test result})$

- Negative predictive value = $\frac{\#(\text{true negatives})}{\#(\text{test negatives}) \ (=\#(\text{true negatives})+\#(\text{false negatives}))} =$
  $\Pr(\text{Negative example}|\text{Negative test result})$

2.2.2 Composites

- Balanced accuracy = arithmetic mean of sensitivity and specificity = accuracy corrected for class imbalance
- F-score = harmonic mean of precision and recall = $\frac{2}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}} = 2\left(\frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}\right)$
- Positive likelihood ratio = $\frac{\underline{\text{true}}\text{ positive rate}}{\underline{\text{false}}\text{ positive rate}}$
- Negative likelihood ratio = $\frac{\underline{\text{false}}\text{ negative rate}}{\underline{\text{true}}\text{ negative rate}}$

2.3 Curves

2.3.1 ROC

- We have a threshold of 0 by default for our class boundary (when working with SVMs, or 0.5 when working with probabilistic models), but we will analyse how the false/true positive/negative rates vary as we vary the threshold
- At $-\infty$ rates are 0%/100%
- At $+\infty$ rates are 100%/0%
- Note that the rates are monotonically non-decreasing/non-increasing as threshold increases
- ROC (receiver-operating characteristic) curve = $\pi_{FPR,TPR}\{$(FPR, TPR, threshold) for all thresholds$\}$ with FPR on the x-axis and TPR on the y-axis

- The ROC curve describes the available trade-offs between false negatives (note that FNR $= 100 -$ TPR and FPR $= 100 -$ TNR) and false positives
- The optimal threshold is typically deemed to be the one that maximises Youden's J statistic (Sensitivity + Specificity $- 1 =$ TPR $-$ FPR) as this corresponds to maximizing the geometric mean $\left(\sqrt{\text{sensitivity} \times \text{specificity}}\right)$
- Area under ROC curve $\left(\text{AUC}_{(\text{ROC})}\right)$ is a model evaluation metric that removes the effect of the model threshold and accounts for binary class imbalance! — it can be seen that ROC curves are themselves invariant under changes in binary class prevalence
- >2 class extension to ROC: Plot a curve for each class as the positive and all others as the negative then take the weighted average relative to the prevalence of each classes — this no longer fully removes the effect of class imbalance!

2.3.2 ROC interpretation

- A random classifier will have ROC curve of $y = x$ (AUC $= 0.5$), a better than random classifier will bow to the left (AUC > 0.5)
- A classifier that is always below the $y = x$ line can be made better than random by flipping its predictions — it can make more sense to think of $|\text{area} - 0.5|$ as the amount of discriminating power a classifier has than to think of plain area as an accuracy proxy
- Equal error rate = value of y (or equiv x) at which $y = x$ on a given ROC curve

2.3.3 PR

- Precision-recall curve is recall (TPR) on X-axis and precision (P(True positive|Positive prediction)) on Y-axis — as with ROC the points are generated by varying the threshold
- Unlike ROC is not monotonic and does not account for imbalanced class prevalence even in the binary case!
- Model A dominates Model B in ROC $\Leftrightarrow$ Model A dominates Model B in PR

2.4 How not to report machine learning results

- Use an inappropriate performance metric for the context (e.g. accuracy when there is class imbalance)
- Tune hyperparameters until you get test results you like (this is closer to over-fitting on the test data (not that this is any better) than pure p-hacking, although there could still be an element of later)
- Forget to reset the model between folds causing all the validation data to have been used as training data by the end
- Split related data (e.g. multiple samples from the same patient) across the dataset splits e.g. folds
- Don't compare performance to other models (e.g. simpler less in-vogue methods)

- Compare to other models, but only based on published results on a different (but hey its the same problem so what could possibly go wrong...) dataset to what you are using
- Train a collection of models on your dataset, but don't spend as long on hyperparameter tuning the baselines as you do your favourite model
- Don't bother with sensitivity analysis ("just use this magic number to seed the random number generator, then everything will work")
- Don't bother understanding how or why your model works
- Run hypothesis tests without verifying that the data obeys the necessary conditions (e.g. normally distributed (fortunately the central limit theorem will probably save you here), independent experiments (this will not be strictly true if we use the same test set for every run, but talking genuinely now this is acceptable to ignore) etc)

3 Dimensionality reduction

3.1 Principal component analysis (PCA)

3.1.1 PCA

- PCA linearly transforms a zero-centred dataset so that the greatest variance is along the first axis (first principal component), the greatest remaining variance is along the second axis (second principal component), ... up to the n^{th} axis n^{th} principal component). By truncating this, dimensionality reduction is achieved with minimal impact

- A matrix X of data vectors can be projected onto the chosen principal components by creating the matrix W with column set the chosen eigenvectors then evaluating $W^T X$

- Variance of data after being projected onto a single component $w = w^T C w$ where C is the covariance matrix (matrix of pairwise covariances) — $C = \frac{1}{\#(\text{data points})} \left(X^T X\right)$ where X is the dataset
- A scree plot is a plot of proportion of variance explained by chosen principal components against number of principal components chosen

3.1.2 Finding principal components

- Finding the first principal component can be written as a constrained optimization problem:
  $\max_w w^T C w$ (variance after projection) s.t. $\|w\|_2^2 = 1$ (normalization so there is a unique solution)
- Lemma: As C is symmetric, $\boldsymbol{\nabla}_w(w^T C w) = 2Cw$
  Proof: Let $D = \boldsymbol{\nabla}_w(w^T C w)$ and chose an arbitrary entry D_k, we will show that $D_k = (2Cw)_k$ and thus be able to deduce that the lemma holds. By the definition of $\boldsymbol{\nabla}$, $D_k = \frac{\partial}{\partial w_k}\left(w^T C w\right)$.

By the definition of matrix multiplication, $w^T C w = \sum_{i\in[n]} \sum_{j\in[n]} w_i C_{ij} w_j$. Thus, $D_k = \sum_{i\in[n]:i\neq k} \sum_{j\in[n]:j\neq k} \frac{\partial}{\partial w_k}(w_i C_{ij} w_j) + \sum_{j\in[n]} \frac{\partial}{\partial w_k}(w_k C_{kj} w_j) + \sum_{i\in[n]} \frac{\partial}{\partial w_k}(w_i C_{ik} w_k) = 0 + \sum_{j\in[n]} C_{kj} w_j + \sum_{i\in[n]} w_i C_{ik} = (Cw)_k + (C^T w)_k$. Finally, as C is symmetric, $C = C^T$, so $D_k = (Cw)_k + (Cw)_k = (2Cw)_k$ as required.

- The Lagrangian multiplier method tells us that every optimal solution obeys $\boldsymbol{\nabla}\left(w^T C w - \alpha\left(\|w\|_2^2 - 1\right)\right) = \vec{\mathbf{0}}$ i.e. $\begin{pmatrix} 2Cw - 2\alpha w + 0 \\ 0 - \|w\|_2^2 + 1 \end{pmatrix} = \vec{\mathbf{0}}$ i.e. $\begin{Bmatrix} Cw = \alpha w \\ \|w\|_2^2 = 1 \end{Bmatrix}$ i.e. w is an eigenvector of C with eigenvalue α — we don't need to use numerical methods to solve this particular optimization problem as good algorithms are known for finding eigenvectors!

- In fact finding all the eigenvectors tells us all the principal components and the associated eigenvalues tell us the explained variance ($w^T C w = $ [eigen] $w^T \alpha w = $ [α is a scalar] $\alpha\|w\|_2^2 = $ [normalization constraint on w] α) so

ordering the eigenvectors in descending order of eigenvalue gives us the principal components in decreasing order of importance

- The first vector actually determines the direction of all the others as they must all be perpendicular to each other, but we go ahead and calculate the entire eigen-decomposition as then we have the corresponding eigenvalues to know which ones (the ones with the lowest variances) to drop for dimensionality reduction

3.2 Linear Discriminant Analysis (LDA)

- LDA works similarly to PCA but, unlike PCA which is unsupervised, LDA is supervised

- LDA finds a projection that (by maximising the ratio of these two quantities) maximises the distance between the means of the two classes but also minimises the variance within each class

- To find LDA: Compute the eigen-decomposition of $W^{-1}(T - W)$ (or equivalently $W^{-1}T - I$) where T is the covariance matrix of all the data and W is the (weighted by number of samples) average of the covariance matrices within each group — deduce that $T - W =$ covariance between groups

4 Classification

4.1 Foundations

4.2 Perceptron

4.3 SVM

4.3.1 SVM

4.3.2 The kernel trick

5 Regression

5.1 Linear regression

5.2 Regularisation

5.3 Generalising regression

6 Topics in machine learning

6.1 Clustering

6.2 Novelty detection

6.3 Ranking

6.4 Recommender systems

7 Neural networks

7.1 Foundations

7.2 Backwards-propagation (backprop)

7.3 Optimizers

7.4 Deep learning

8 Convolutional Neural Networks (CNNs)

8.1 Motivation

8.2 Tricks of the trade

8.3 ResNet

9 Large Language Models (LLMs)

9.1 The transformer architecture

9.2 (Self-)Attention

9.3 MLP blocks

10 Graph Neural Networks (GNNs)

10.1 GNNs generalise CNNs

10.2 Message passing

10.2.1 GCN

10.2.2 GAT