Machine Learning

Course summary

Regression • SVM and kNN • Ensemble methods

Notation: m examples (x_i, y_i), $x_i \in \mathbb{R}^d$, design matrix $X \in \mathbb{R}^{m \times d}$ (row i is $x_i^\top$, a column of ones for the intercept), parameters θ, cost $J(\theta)$, step size η, penalty λ.

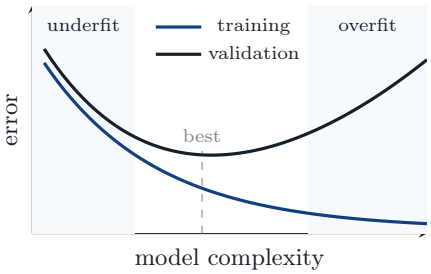
1. LEARNING WORKFLOW AND EVALUATION

We fit θ by minimising $J(\theta) = \frac{1}{m}\sum_i \ell(\theta; x_i, y_i)$.

Splits

- **Training set**: fit θ.
- **Validation set** or k-fold cross-validation: choose hyperparameters and the model.
- **Test set**: used once, at the end. Never tune on it.

Standardise each feature with the mean and standard deviation of the *training* set only.



More complex when: $\lambda \downarrow$, $C \uparrow$, $\sigma \downarrow$, $k \downarrow$, depth $\uparrow$.

Bias-variance trade-off

Too simple: high bias, *underfitting*. Too complex: high variance, *overfitting*.

Metrics

Regression

- MSE $= \frac{1}{m}\sum_i (y_i - \hat{y}_i)^2$
- MAE $= \frac{1}{m}\sum_i |y_i - \hat{y}_i|$, less sensitive to outliers
- $R^2 = 1 - \dfrac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$, share of variance explained

Classification

	pred. +	pred. −
true +	TP	FN
true −	FP	TN

confusion matrix

- Accuracy $= \frac{\text{TP}+\text{TN}}{m}$
- Precision $= \frac{\text{TP}}{\text{TP}+\text{FP}}$: when we say +, how often right?
- Recall $= \frac{\text{TP}}{\text{TP}+\text{FN}}$: share of true + found
- $F_1 = \dfrac{2\,\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

2. CONVEXITY AND OPTIMISATION

Convexity decides uniqueness

- **Convex**: $\nabla^2 J(\theta) \succeq 0$ for all θ. Every stationary point is a global minimum.
- **Strictly convex** (e.g. $\nabla^2 J \succ 0$ everywhere): *at most one* minimiser.
- **μ-strongly convex**: $\nabla^2 J \succeq \mu I$, $\mu > 0$. *Exactly one* minimiser.
- Sums of convex functions and convex $\circ$ affine maps are convex. Convex + strongly convex is strongly convex.

Gradient toolbox

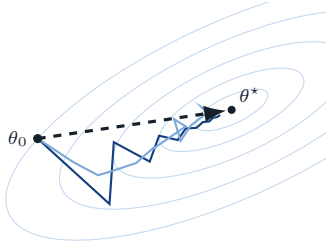
- $\nabla(a^\top\theta) = a$, $\quad \nabla(\theta^\top A\theta) = (A + A^\top)\theta$
- $\nabla\|y - X\theta\|^2 = -2X^\top(y - X\theta)$, $\quad$ Hessian $2X^\top X$
- $\nabla(\lambda\|\theta\|^2) = 2\lambda\theta$, $\quad$ Hessian $2\lambda I$

Linear predictor + scalar loss. Let $\ell(y, t)$ be a loss of the scalar score t, and $J(\theta) = \frac{1}{m}\sum_i \ell(y_i, x_i^\top\theta)$. The chain rule gives

$$\nabla J(\theta) = \frac{1}{m}\sum_i \partial_t \ell(y_i, x_i^\top\theta)\, x_i,$$

Three algorithms

Gradient descent	$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$. Uses all m examples: cost $O(md)$ per step.
Stochastic GD	$\theta_{t+1} = \theta_t - \eta_t \frac{1}{b}\sum_{i \in B_t} \nabla\ell(\theta_t; x_i, y_i)$, random mini-batch B_t of size b. Unbiased, noisy gradient. Cost $O(bd)$, *independent of* m.
Newton	$\theta_{t+1} = \theta_t - \eta_t[\nabla^2 J(\theta_t)]^{-1}\nabla J(\theta_t)$. Uses curvature. Needs $\nabla^2 J \succ 0$. High cost per step but very few steps.



GD: zig-zag if η large
SGD: cheap but noisy steps
Newton: follows the curvature

3. LINEAR REGRESSION AND PENALISATION

Model $y = X\theta + \varepsilon$. **OLS** minimises $\|y - X\theta\|^2$. Setting the gradient to zero gives the normal equations

$$X^\top X\,\hat\theta = X^\top y, \qquad \hat\theta = (X^\top X)^{-1} X^\top y,$$

the closed form holding when $X^\top X$ is invertible.

When OLS breaks. If $d > m$ or features are collinear, $X^\top X$ is singular: infinitely many solutions.

Ridge: penalise the size of θ

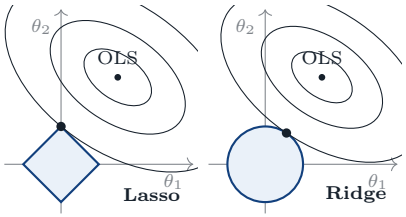
$$\hat\theta_\lambda = \arg\min_\theta \|y - X\theta\|^2 + \lambda\|\theta\|^2 = (X^\top X + \lambda I)^{-1} X^\top y.$$

- Hessian $2(X^\top X + \lambda I) \succeq 2\lambda I$: strongly convex, unique solution, always.
- $\lambda \to 0$: OLS. $\lambda \to \infty$: $\hat\theta_\lambda \to 0$. Coefficients shrink but never reach 0.
- $\lambda \uparrow$: training error $\uparrow$, bias $\uparrow$, variance $\downarrow$.
- Do not penalise the intercept: centre y, standardise X.

Lasso: sparse solutions

$$\arg\min_\theta \|y - X\theta\|^2 + \lambda\|\theta\|_1,$$

with $\|\theta\|_1 = \sum_j |\theta_j|$. Convex, no closed form in general. Some $\hat\theta_j$ are *exactly* 0: built-in variable selection.



The level sets of the fit term typically first touch the ℓ^1 ball at a corner, where $\theta_1 = 0$. The ℓ^2 ball has no corner.

4. LOGISTIC AND SOFTMAX REGRESSION

Binary labels $y \in \{0, 1\}$. Squash a linear score into a probability:

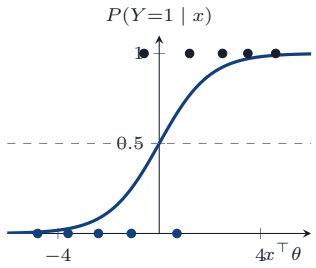
$$P(Y = 1 \mid x) = \sigma(x^\top\theta), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

σ is increasing, $1 - \sigma(z) = \sigma(-z)$, $\sigma' = \sigma(1 - \sigma)$. Predict 1 iff $x^\top\theta > 0$: a linear boundary.

Maximum likelihood with $Y_i \mid x_i \sim \text{Bernoulli}(p_i)$, $p_i = \sigma(x_i^\top\theta)$, means minimising the **cross-entropy**

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} \big[y_i \log p_i + (1 - y_i)\log(1 - p_i)\big].$$

Convex, no closed form: gradient descent or Newton.



Data in $\{0, 1\}$, fitted probability in $(0, 1)$.

Softmax regression: K classes

One parameter vector θ_k per class, $\Theta = [\theta_0, \dots, \theta_{K-1}] \in \mathbb{R}^{d\times K}$, one-hot labels $Y_{ik} = \mathbf{1}\{y_i = k\}$.

$$P(Y = k \mid x) = \frac{e^{x^\top \theta_k}}{\sum_j e^{x^\top \theta_j}}, \qquad J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m} \log P(Y = y_i \mid x_i), \qquad \nabla_\Theta J = \frac{1}{m} X^\top (P - Y),$$

with $P_{ik} = P(Y = k \mid x_i)$. Prediction is $\arg\max_k x^\top \theta_k$.

5. ENTROPY AND KULLBACK-LEIBLER DIVERGENCE

For distributions p, q on a finite set (with $0 \log 0 = 0$):

$$\underbrace{H(p) = -\sum_x p(x) \log p(x)}_{\text{entropy: uncertainty of } p} \qquad \underbrace{H(p,q) = -\sum_x p(x) \log q(x)}_{\text{cross-entropy}} \qquad \underbrace{\mathrm{KL}(p \,\|\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}}_{\text{divergence of } q \text{ from } p}$$

- $H(p,q) = H(p) + \mathrm{KL}(p \,\|\, q)$. For fixed p, minimising the cross-entropy in q is minimising $\mathrm{KL}(p \,\|\, q)$.
- $\mathrm{KL}(p \,\|\, q) \geq 0$, with equality iff $p = q$. It is not symmetric, so it is not a distance.

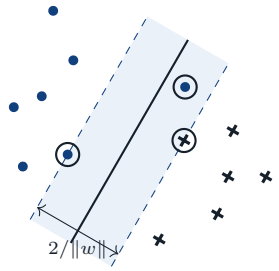
6. SUPPORT VECTOR MACHINES AND KERNELS

Labels $y_i \in \{-1, +1\}$, classifier $\mathrm{sign}(w^\top x + b)$.

- **Margin**: the band $\{|w^\top x + b| \leq 1\}$ has width $2/\|w\|$. The SVM picks the widest band.
- **Soft margin**: allow violations and pay for them,

$$\min_{w,b} \ \tfrac{1}{2}\|w\|^2 + C m_i \max\big(0,\, 1 - y_i(w^\top x_i + b)\big).$$

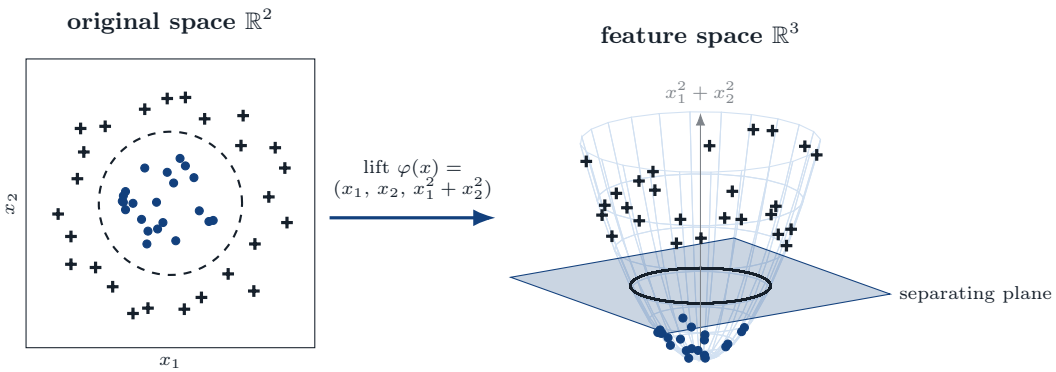
  Penalty + hinge loss, as Ridge: C plays the role of $1/\lambda$. Large C: narrow band, risk of overfitting.
- **Support vectors**: the points with $y_i(w^\top x_i + b) \leq 1$. Only they define the solution, $w = \sum_i \alpha_i y_i x_i$.
- The solution uses the data *only through inner products* $\langle x_i, x_j \rangle$.



Circled: support vectors, including one violation inside the band.

Kernel trick

Idea. When the classes cannot be separated by a hyperplane, map the data with a feature map φ into *another space, usually of higher dimension*, where they can be separated linearly. A hyperplane in the new space is a curved boundary in the original one.



No line separates the two classes in $\mathbb{R}^2$. After the lift, a horizontal plane does. Seen back in $\mathbb{R}^2$, this plane is the dashed circle.

Trick. Since only inner products are needed, we never compute φ. We use a **kernel** $K(x, x') = \langle \varphi(x), \varphi(x') \rangle$, computed directly in the original space:

$$h(x) = \sum_i \alpha_i y_i \, K(x_i, x) + b.$$

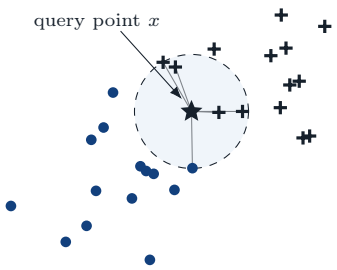
7. NEAREST NEIGHBOURS AND THE CURSE OF DIMENSIONALITY

Rule. To predict at x, find the k training points closest to x, written $N_k(x)$.

- Classification: majority vote in $N_k(x)$.
- Regression: $\hat{f}(x) = \frac{1}{k}\sum_{i \in N_k(x)} y_i$.
- No training: the model is the stored dataset. Distances need standardised features.

Role of k. Small k: flexible, low bias, high variance. $k = 1$ has training error 0, meaningless. Large k: smooth, high bias. $k = m$ predicts the same thing everywhere. Choose k by validation.

Curse of dimensionality. When d grows with m fixed, points get far from each other. The "nearest" neighbours are no longer close, and local averages become biased. Remedy: fewer, more relevant features.



$k = 5$: 4 crosses, 1 dot ⇒ cross

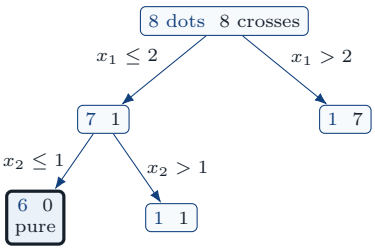
The 5 nearest neighbours of the star vote. The majority wins.

8. DECISION TREES AND ENSEMBLE METHODS

Decision tree. A sequence of yes/no questions "$x_j \leq s$?". Each leaf predicts the majority class of its training points (or their mean in regression). The space is cut into rectangles.

Choosing a question. A node is **pure** when all its training points belong to the same class. Each split is chosen so that the two children are as pure as possible. Impurity of a node with class proportions p_k: Gini $1 - \sum_k p_k^2$ or entropy $-\sum_k p_k \log p_k$. Both are 0 for a pure node and largest when the classes are balanced.

Shallow tree: too simple (bias). Deep tree: memorises the data (variance). Ensembles fix both.



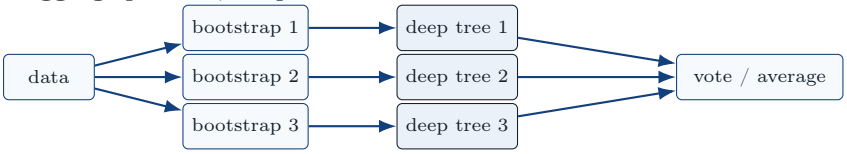
Each split makes the children purer than the parent.

Two ways to combine many trees

Boosting: sequential, small trees



Bagging: parallel, deep trees



Boosting. Build a strong model by *adding* weak ones (stumps, shallow trees) one at a time. Each new tree concentrates on what the current ensemble still gets wrong. **Reduces bias.**

- AdaBoost: misclassified points get more weight for the next tree. Accurate trees get a larger vote α_t.
- Gradient boosting: the next tree is fitted to the residuals $y_i - F(x_i)$, then $F \leftarrow F + \eta h$. XGBoost is a fast, penalised version.
- Too many trees or a large η overfit: stop early on validation data.

Bagging. Fit deep trees *independently* on bootstrap samples (m draws with replacement), then vote. Their errors partly cancel. **Reduces variance.**

- Random forest: at each node, search only a random subset of about $\sqrt{d}$ features. More diverse trees, better vote, faster.
- Each point is out of a bootstrap sample with probability $(1 - \frac{1}{m})^m \approx e^{-1} \approx 37\%$: free out-of-bag error estimate.
- More trees never hurt, they only cost time.