MATH 495 Machine Learning — Lecture 1 Notes

Source: Tommi Jaakkola, *6.867 Machine learning, lecture 1*, MIT OpenCourseWare (Fall 2006). These bullets are a faithful, lecture-ready condensation of the original text.

Scenario & Data: Access Control via Face Images

- **Task.** Automated access control from a still face image: output label +1 (permit) or −1 (deny).
- **Available information.** Labeled images collected while access was manual: positives = allowed, negatives = denied.
- **Augmenting negatives.** Because denials are rare, include other face images of people not expected to be permitted; prefer similar camera/face orientation (e.g., other buildings with similar systems).
- **Objective.** Learn a classifier mapping image → $\{\pm 1\}$ using *only* the labeled training set.

Representation & Notation

- **Vectorization.** Grayscale image $\rightsquigarrow$ column vector $x \in \mathbb{R}^d$ by stacking pixel intensities (column by column).
- **Example.** 100×100 pixels $\Rightarrow d = 10{,}000$. All images assumed same size.
- **Classifier.** Binary-valued function $f : \mathbb{R}^d \to \{-1, 1\}$ chosen from training data alone.
- **Agnosticism about inputs.** From the classifier's perspective, inputs could be any measured features (weights, heights, . . .), not necessarily "image semantics."
- **Training set.** $\{(x_t, y_t)\}_{t=1}^n$ with $x_t \in \mathbb{R}^d$, $y_t \in \{\pm 1\}$; this is the *only* information constraining f.

Memorization vs. Generalization

- **Thought experiment (distinct-pixel rule).** With $n = 50$ images of size 128×128 (pixel values in $\{0, \ldots, 255\}$), it may be possible to find a pixel index i whose values are all distinct across the n training images.
- **A trivial perfect-fit rule.** Let x_i^t denote pixel i of training image t and x_i' that of a new image x'. Define

$$f_i(x') = \begin{cases} y_t, & \text{if } x_i^t = x_i' \text{ for some } t \in \{1, \ldots, n\} \text{ (in this order)}, \\ -1, & \text{otherwise.} \end{cases} \tag{1}$$

- **Why this fails.** Even same-person images vary (orientation, lighting, etc.). Rule (1) can be *perfect on training* yet useless on new images.
- **Goal re-stated.** We seek *generalization*: performance on the training set should be indicative of performance on *unseen* images from the same task.

Model Selection (Choosing a Function Class)

- **Key idea.** Constrain the set of candidate functions: if a function from this class performs well on training data, it is *likely* to perform well on new data.
- **Capacity trade-off.**
  - If the class is *too large*, we can fit idiosyncrasies (overfit) and fail to generalize.
  - If the class is *too small*, no function may fit even the training set well (underfit).
- **Problem name.** Choosing such a class is the *model selection* problem.

Linear Classifiers Through the Origin

- **Fix a class.** Thresholded linear maps:
$$f(x;\theta) = \text{sign}(\theta_1 x_1 + \cdots + \theta_d x_d) = \text{sign}(\theta^\top x), \quad \theta \in \mathbb{R}^d. \tag{2}$$
- **Parameterization.** Different θ yield different functions in the class; the class is $\{x \mapsto \text{sign}(\theta^\top x) : \theta \in \mathbb{R}^d\}$.
- **Geometry.**
  - Prediction changes only when the argument of sign crosses 0; the *decision boundary* is $\{x : \theta^\top x = 0\}$.
  - This boundary is a $(d-1)$-dimensional hyperplane through the origin ($x = 0$ satisfies the equation).
  - θ is *normal* to the hyperplane; direction of steepest increase of $\theta^\top x$.
- **What we lost by restricting to linear.**
  - No explicit access to pixel adjacency / local continuity (e.g., skin smoothness).
  - If we apply the *same fixed permutation* of pixel positions to all images, predictions are unchanged: permutation just reorders the sum in (2).

Training Error and Loss

- **Empirical 0–1 training error.**
$$\hat{E}(\theta) = \frac{1}{n}\sum_{t=1}^{n}\Big(1 - \delta\big(y_t, f(x_t;\theta)\big)\Big) \;=\; \frac{1}{n}\sum_{t=1}^{n}\text{Loss}\big(y_t, f(x_t;\theta)\big), \tag{3}$$
where $\delta(y, y') = 1$ if $y = y'$ and 0 otherwise.
- **Loss perspective.** Use a loss $\text{Loss}(y, \hat{y})$ to encode costs (e.g., false accept vs. false reject). Lecture 1 focuses on *zero–one* loss: 1 for mistakes, 0 otherwise.

Learning Algorithm: The Perceptron

- **Goal.** Find θ minimizing the training error (3) within the linear class (2).
- **Idea.** Adjust parameters on mistakes to reduce classification errors.
- **Algorithm (cycle through training examples).**

$$\theta \leftarrow \theta + y_t x_t \quad \text{if} \quad y_t \neq f(x_t; \theta). \tag{4}$$

- **Why the update helps.**
  - On a mistake, the signed score $y_t\, \theta^\top x_t < 0$; on a correct classification, $y_t\, \theta^\top x_t > 0$.
  - After an update $\theta' = \theta + y_t x_t$ on the same example x_t,

$$y_t\, \theta'^\top x_t \;=\; y_t\, (\theta + y_t x_t)^\top x_t \;=\; y_t\, \theta^\top x_t + y_t^2\, \|x_t\|^2 \;=\; y_t\, \theta^\top x_t + \|x_t\|^2. \tag{5}$$

  - Hence the signed score increases by $\|x_t\|^2$; repeatedly revisiting the *same* mistake eventually makes it correct.
- **Caveat.** Mistakes on other examples may move θ in competing directions; (5) alone does not prove convergence.

Analysis (Pointer to Next Lecture)

- **Stopping condition.** Perceptron stops updating only when all training images are classified correctly (no mistakes).
- **Guarantee.** If the training set is *linearly separable*, perceptron finds a separating classifier in a *finite* number of updates (proof deferred to Lecture 2).