MATH 565: Lecture 1 (01/13/2026)

Today: * logistics, syllabus, ...
* problems in ML, optimization for them

This is Optimization for Machine Learning (Math 565)

I'm Bala Krishnamoorthy (call me Bala).

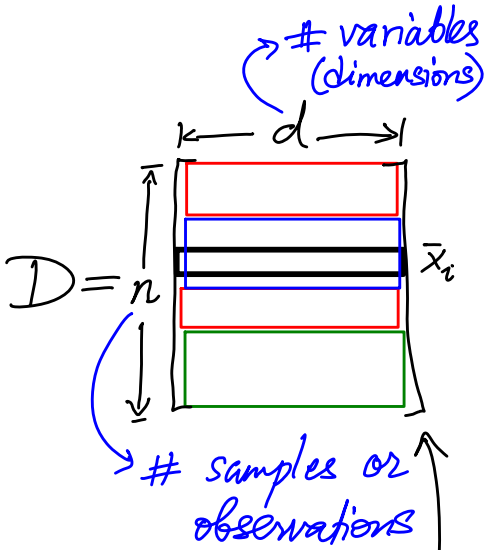
We will **not** use Canvas. All materials for the class will be posted on the course web page: https://bala-krishnamoorthy.github.io/Math565.html

Download the book PDF (accessible via WSU Libraries)!

* Check the syllabus on the course web page.
* homework assignments will be posted at least a week before its due.

Machine Learning Problems

1. **Clustering** We are given a data matrix that is $n \times d$ for n observations (or samples) each having values for d variables (dimension=d). The i^{th} observation is denoted $\bar{x}_i$ (d-vector), which forms the i^{th} row of D (when written as $\bar{x}_i^T$).

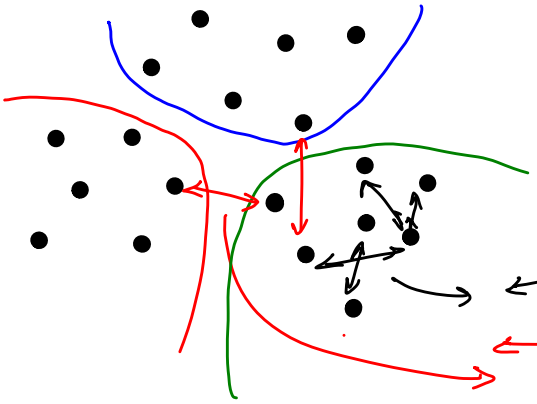


The goal of clustering is to divide the n observations into k disjoint subsets or clusters. The # clusters k may be user-defined, e.g., in k-means clustering, or may be determined as part of the clustering method. Equivalently, the rows of D are to be divided into k disjoint sets (e.g., into red, blue, and green clusters as illustrated here).

Notation: A, X, Y: matrices or sets (uppercase letters)
$\bar{x}, \bar{y}, \bar{\alpha}, \bar{\theta}$: vectors (lower case letters with a bar)
x, β, γ, a: scalars (lower case letters)

Another ("metric") way to think about clustering is illustrated in 2D.

$\bullet \bar{x}_i$



division into 3 clusters (red, blue, green)

$\longleftrightarrow$: within cluster distances
$\longleftrightarrow$: distance to nearest point outside cluster

One objective function used in clustering is to minimize the sum of all within cluster pairwise distances while also maximizing the sum of distances of each point to the nearest point outside its cluster.

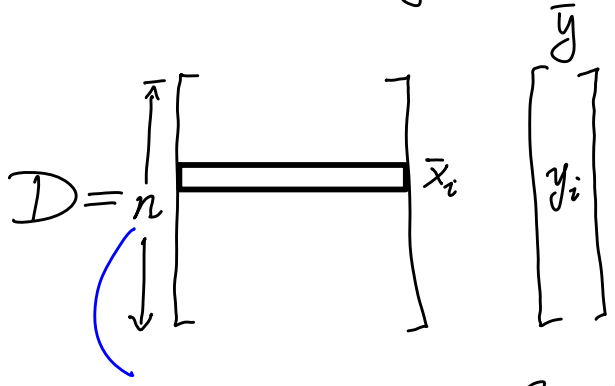
An example: Each $\bar{x}_i$ represents a consumer, and the entries are the $ amounts they spent on fruits, meat, toiletteries, or other items (in a retail or grocery store). The store may be interested in the clusters to target ads for specific products.

Clustering is considered an unsupervised learning problem, since only the data ($\bar{x}_i$) is used without the membership information (of which cluster each $\bar{x}_i$ belongs to).

As opposed to classification/regression problems being introduced now, which are supervised learning problems.

2. Classification

Here, apart from the $n \times d$ data matrix D, we are given also an n-vector $\bar{y}$ of labels for each observation.



For instance, $y_i \in \{-1, 1\}$ (or $\{0, 1\}$) to capture a YES/NO aspect of each observation, e.g., whether customer i is possibly responsive to ads. In another instance, $y_i \in \{R, G, B\}$ (3 colors). Note that such labels are categorical, and it is not obvious how to represent these labels using numerical values. The YES/NO aspect could still be modeled as a continuous numerical value coupled with a cut-off value (to determine YES).

The goal is to build a "model" on D and $\bar{y}$, which is the training set, and use that model to predict the $\bar{y}_t$ values for an external test data set D_t ($n_t \times d$) for which $\bar{y}_t$ is not known.

The model can be described as

$$y_i \approx f(\bar{x}_i).$$

The function to be learned, f, is often parametrized using a weight vector $\bar{w}$, and hence we write

$$y_i \approx f_{\bar{w}}(\bar{x}_i)$$

For instance, in the case of binary classification with $y_i \in \{-1, 1\}$, we could take

$$y_i = \text{sign}\{f_{\bar{w}}(\bar{x}_i)\}$$

Q. How does one choose $\bar{w}$?

We usually pick $\bar{w}$ such that the mismatch between y_i and $f_{\bar{w}}(\bar{x}_i)$, i.e., the "loss", is minimized. The form of f and this loss function are often carefully chosen/constructed to solve

$$\min_{\bar{w}} \sum_{i=1}^{n} \ell\left(y_i - f_{\bar{w}}(\bar{x}_i)\right)$$

ℓ → loss function

Such classification tasks are considered supervised learning, since we are guided by the y_i's (known labels). The goal is to use the built function $f_{\bar{w}}(\bar{x})$ to predict the labels $\bar{y}_t$ for an external or test data set D_t $(n_t \times d)$ (for which the labels are unknown). Since D_t is not used in the training process, such predictions for test sets are called as generalizations.

The step where we want to minimize a loss function makes classification an optimization problem! Before considering details of optimization, we present a simpler ML problem: regression.

3. Regression

We're given D $(n \times d)$ just as in classification, but now, $y_i \in \mathbb{R}$, i.e., it is a numerical value (rather than a label).

The simplest version of regression is linear regression, where the task is to fit a line through the given set of n points (in 2D, and a plane in d-dimensions).

$$y_i = f_{\bar{w}}(\bar{x}_i) = \bar{w}^T \bar{x}_i = \bar{x}_i^T \bar{w}$$

Equivalently, $\bar{y} = D\bar{w}$.

And the loss function usually used is the sum of squared errors:

$$J = \frac{1}{2} \|D\bar{w} - \bar{y}\|^2.$$

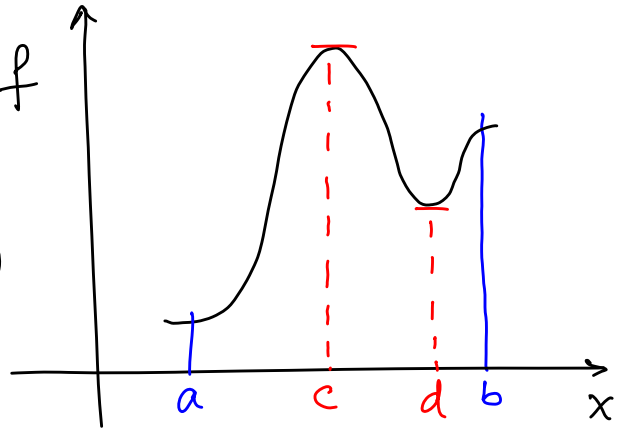
Hence, linear regression becomes the optimization problem:

$$\min_{\bar{w}} \frac{1}{2} \|D\bar{w} - \bar{y}\|.$$

Optimization in 1D Calculus

Find $\min f(x)$ for $a \le x \le b$

* We set $f'(x) = 0$ to find critical points.
* $f''(x) > 0$: local minima (e.g., d)
  $f''(x) < 0$: local maxima (e.g., c)
  $f''(x) = 0$: saddle points



Compare the function values at local minima with those at the end points (a and b) to determine the true ("global") minimum (@ $x = a$ in the figure here).

Optimization in d-dimensions

In general, we consider problems of the form

$$\begin{aligned} &\min f(\bar{x}) \\ &\text{s.t.} \quad \bar{g}(\bar{x}) \leq \bar{0} \\ &\quad\;\; \bar{h}(\bar{x}) = \bar{0} \end{aligned}$$

Under appropriate assumptions, we can specify optimality conditions (local by default, global when the functions are "nice"), e.g., Karush-Kuhn-Tucker (KKT) conditions.

Corresponding to $f'(x)=0$, we have $\nabla f = \bar{0}$ for

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}, \text{ the gradient of } f.$$

Back to linear regression:

Recall: $\min_{\bar{w}} J = \frac{1}{2} \| D\bar{w} - \bar{y} \|^2$

$$\Rightarrow \nabla J = D^T D \bar{w} - D^T \bar{y} = \bar{0}$$

$$\Rightarrow \bar{w} = (D^T D)^{-1} D^T \bar{y}$$

closed form expression!

It turns out that this critical point is the (unique) global minimizer of J. In this sense, linear regression is perhaps the easiest ML problem!

But more generally, we do not get closed form solutions in this form. In fact, we usually use gradient descent to iteratively update $\bar{w}$:

We choose an initial $\bar{w}$ in some way, e.g., randomly. Then, in each iteration, we update

$$\bar{w} \leftarrow \bar{w} - \alpha \nabla J(\bar{w})$$

α is the step size, also called the learning rate. We usually need to choose α carefully to ensure the iterations converge. We also have to make assumptions about the function f to guarantee efficient convergence of this gradient-descent method.