# Lecture Notes I – Examples of Predictors. Nearest Neighbor and Kernel Predictors

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

January 6, 2026

Prediction problems by the type of output

The Nearest-Neighbor and kernel predictors

Some concepts in Classification

**Reading** HTF Ch.: 2.3.2 Nearest neighbor, 6.1–3. Kernel regression, 6.6.2 kernel classifiers,, Murphy Ch.: , Bach Ch.:

## Prediction problems by the type of output

In supervised learning, the problem is *predicting* the value of an **output** (or **response** – typically in regression, or **label** – typically in classification) variable $Y$ from the values of some observed variables called **inputs** (or **predictors, features, attributes**) $(X_1, X_2, \ldots X_d) = X$. Typically we will consider that the input $X \in \mathbb{R}^d$.

## Prediction problems by the type of output

In supervised learning, the problem is *predicting* the value of an **output** (or **response** – typically in regression, or **label** – typically in classification) variable $Y$ from the values of some observed variables called **inputs** (or **predictors, features, attributes**) $(X_1, X_2, \dots X_d) = X$. Typically we will consider that the input $X \in \mathbb{R}^d$. Prediction problems are classified by the type of response $Y \in \mathcal{Y}$:

- *regression*: $Y \in \mathbb{R}$
- *binary classification*: $Y \in \{-1, +1\}$
- *multiway classification*: $Y \in \{y_1, \dots y_m\}$ a finite set
- *ranking*: $Y \in \mathbb{S}_p$ the set of permutations of $p$ objects
- *multilabel classification* $Y \subseteq \{y_1, \dots y_m\}$ a finite set (i.e. each $X$ can have several labels)
- *structured prediction* $Y \in \Omega_V$ the state space of a graphical model over a set of [discrete] variables $V$

## Example (**Regression.**)

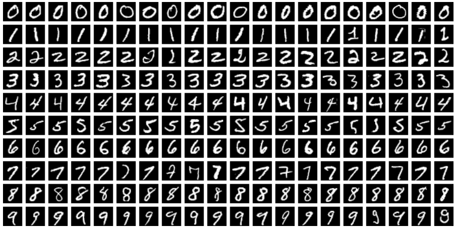
- $Y$ is the proportion of high-school students who go to college from a given school in given year. $X$ are school attributes like class size, amount of funding, curriculum (note that they aren't all naturally real valued), median income per family, and other inputs like the state of the economy, etc. Note also that $Y \in [0, 1]$ here.
- $Y \geq 0$ is the income of a person, and $X_j$ are attributes like education, age, years out of school, skills, past income, type of employment.
  Economic forecasts are another example of regression. Note that in this problem as well as in the previous, the $Y$ in the previous period, if observed, could be used as a predictor variable for the next $Y$. This is typical of structured prediction problems.
- Weather prediction is typically a regression problem, as winds, rainfall, temperatures are continuous-valued variables.
- Predicting the box office totals of a movie. What should the inputs be?
- Predicting perovskite degradation. Perovskites are a type of crystal considered promising for the fabrication of solar cells. In standard use, such a material must have a life time $Y$ of 30 years. How can one predict which material will last that long without waiting for 30 years?
  $Y$ is time to degradation, $X_j$ are material composition, experimental conditions, and measurements of initial values of physical parameters.

## Example (**(Anomaly) detection.**)

This is a binary classification problem. $Y \in \{\mathrm{normal}, \mathrm{abnormal}\}$. For instance, $Y$ could be "HIV positive" vs "HIV negative" (which could be abbreviated as "+", "-") and the inputs $X$ are concentration of various reagents and lymph cells in the blood.
Anomaly detection is a problem also in artificial systems, as any device may be functioning normally or not. There are also more general detection problems, where the object detected is of scientific interest rather than an "alarm": detecting Gamma-ray bursts in astronomy, detecting meteorites in Antarctica (a robot collects rocks lying on the ice and determines if the rock is terrestrial or meteorite). More recently, *Artificial Intelligence* tasks like detecting faces/cars/people in images or video streams have become possible.

## Example (**Multiway classification.**)

Handwritten digit classification: $Y \in \{0, 1, \ldots 9\}$ and $X$=black/white 64$\times$ 64 image of the digit. For example, ZIP codes are being now read automatically off envelopes.
OCR (Optical character recognition). The task is to recognized printed characters automatically. $X$ is again a B/W digital image, $Y \in \{a-z, A-Z, 0-9, "." , "," , \ldots\}$, or another character set (e.g. Chinese).



## Example (**Diagnosis**)

Diagnosis is multiway classification + anomaly detection. $Y = 0$ means "normal/healthy", while $Y \in \{1, 2, \ldots\}$ enumerates failure modes/diseases.

## Example (**Structured prediction.**)

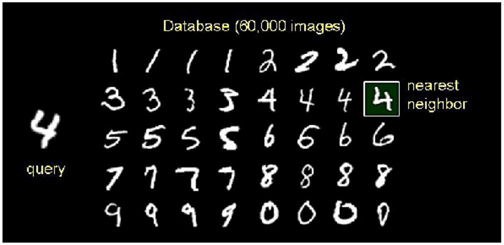
Speech recognition. $X$ is a segment of digitally recorded speech, $Y$ is the word corresponding to it. Note that it is not trivial to *segment speech*, i.e to separate the speech segment that corresponds to a given word. These segments have different lengths too (and the length varies even when the same word is spoken).
The classification problem is to associate to each segment $X$ of speech the corresponding word. But one notices that the words are not indepedent of other neighboring words. In fact, people speak in sentences, so it is natural to recognize each word in dependence from the others. Thus, one imposes a graphical model structure on the words corresponding to an utterance $X^1, X^2, \dots X^m$. For instance, the labels $Y^{1:m}$ could form a *chain* $Y^1 - Y^2 - \dots Y^m$. Other more complex graphical models structures can be used too.

## Example (**Large Language Models (LLM)**)

LLMs are machines for structured prediction, where the label space $\mathcal{Y}$ coincides with the input space $\mathcal{X}$, e.g., both the input and the output are sequences of English words (called **tokens**). The output tokens $y^1, y^2, \dots y^t \dots$ depend on the previous output tokens (the context), as well as on the entire sequence of input tokens.

## Supervised Learning

- Nearest neighbour:



## The Nearest-Neighbor predictor

- **1-Nearest Neighbor** The label of a point $x$ is assigned as follows:
  1. find the example $x^i$ that is nearest to $x$ in $\mathcal{D}$ (in Euclidean distance)
  2. assign $x$ the label $y^i$, i.e.

$$\hat{y}(x) = y^i$$

## The Nearest-Neighbor predictor

- **1-Nearest Neighbor** The label of a point $x$ is assigned as follows:
  1. find the example $x^i$ that is nearest to $x$ in $\mathcal{D}$ (in Euclidean distance)
  2. assign $x$ the label $y^i$, i.e.

$$\hat{y}(x) = y^i$$

- **K-Nearest Neighbor** (with $K = 3, 5$ or larger)
  1. find the $K$ nearest neighbors of $x$ in $\mathcal{D}$: $x^{i_1, \ldots i_K}$
  2. - for classification $f(x) =$ the most frequent label among the $K$ neighbors (well suited for multiclass)
     - for regression $f(x) = \frac{1}{K} \sum_{i \text{ neighbor of } x} y^i$ = mean of neighbors' labels

## The Nearest-Neighbor predictor

- **1-Nearest Neighbor** The label of a point $x$ is assigned as follows:
  1. find the example $x^i$ that is nearest to $x$ in $\mathcal{D}$ (in Euclidean distance)
  2. assign $x$ the label $y^i$, i.e.
  $$\hat{y}(x) = y^i$$
- **K-Nearest Neighbor** (with $K = 3, 5$ or larger)
  1. find the $K$ nearest neighbors of $x$ in $\mathcal{D}$: $x^{i_1, \ldots i_K}$
  2. - for classification $f(x) =$ the most frequent label among the $K$ neighbors (well suited for multiclass)
     - for regression $f(x) = \frac{1}{K}\sum_{i \text{ neighbor of } x} y^i$ = mean of neighbors' labels
- No parameters to estimate!
- No training!
- But all data must be stored (also called memory-based learning)

## Kernel regression and classification

- Like the $K$-nearest neighbor but with "smoothed" neighborhoods
- The predictor

$$f(x) = \sum_{i=1}^{n} \beta_i b(x, x^i) y^i \tag{1}$$

where $\beta_i$ are coefficients

## Kernel regression and classification

- Like the $K$-nearest neighbor but with "smoothed" neighborhoods
- The predictor
$$f(x) \; = \; \sum_{i=1}^{n} \beta_i b(x, x^i) y^i \tag{1}$$
where $\beta_i$ are coefficients
- Intuition: center a "bell-shaped" *kernel* function $b$ on each data point, and obtain the prediction $f(x)$ as a weighted sum of the values $y^i$, where the weights are $\beta_i b(x, x^i)$
- Requirements for a kernel function $b(x, x')$
  1. non-negativity
  2. symmetry in the arguments $x, x'$
  3. optional: radial symmetry, bounded support, smoothness
- A typical kernel function is the **Gaussian kernel** (or **Radial Basis Function (RBF)**)

$$b(z) \; \propto \; e^{-z^2/2} \tag{2}$$

$$b_h(x, x') \; \propto \; e^{-\frac{||x-x'||^2}{2h^2}} \quad \text{with} \quad h = \textbf{the kernel width} \tag{3}$$

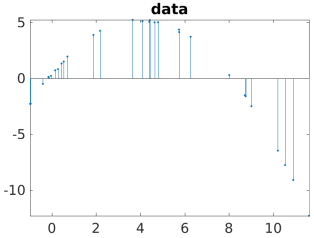
## Regression example

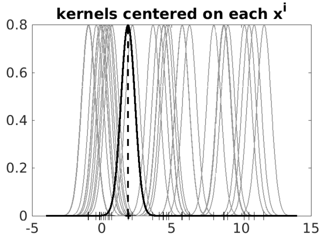
A special case in wide use is the Nadaraya-Watson regressor

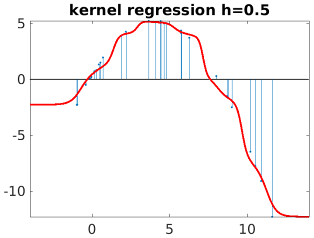
$$f(x) \;=\; \frac{\sum_{i=1}^{n} b\left(\frac{||x-x^i||}{h}\right) y^i}{\sum_{i=1}^{n} b\left(\frac{||x-x^i||}{h}\right)}. \tag{4}$$

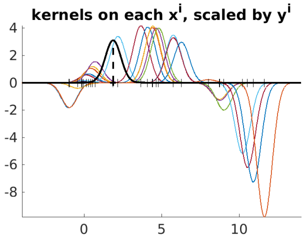
In this regressor, $f(x)$ is always a convex combination of the $y^i$'s, and the weigths are proportional to $b_h(x, x^i)$.
The Nadaraya-Watson regressor is biased if the density of $P_X$ varies around $x$.

## An example: noisy data from a parabola









## Local Linear Regression

To correct for the bias (to first order) one can estimate a regression line around $x$.

1. Given **query point** $x$
2. Compute kernel $b_h(x, x^i) = w_i$ for all $i = 1, \dots N$
3. Solve weighted regression $\min_{\beta,\beta_0} \sum_{i=1}^{d} w_i \left(y^i - \beta^T x^i - \beta_0\right)^2$ to obtain $\beta, \beta_0$ ( $\beta, \beta_0$ depend on $x$ through $w_i$)
4. Calculate $f(x) = \beta^T x + \beta_0$

Exercise Show that Nadaraya-Watson solves a local linear regression with fixed $\beta = 0$

## Kernel binary classifiers

- Obtained from Nadaraya-Watson by setting $y^i$ to $\pm 1$.
- Note that the classifier can be written as the difference of two non-negative functions

$$f(x) \propto \sum_{i:y^i=1} b\left(\frac{||x-x^i||}{h}\right) - \sum_{i:y^i=-1} b\left(\frac{||x-x^i||}{h}\right). \tag{5}$$

## (Radial) Basis Function predictors

A regressor similar to the kernel predictor is

$$f(x) \;=\; \sum_{i=1}^{M} b(x, \xi^i)\beta_i \tag{6}$$

The difference is that the "bumps" are not placed on data points, but at a set of $M$ points $\xi^i$ to be determined.
The $f$ in (6) is an example of **function basis** (or **basis functions** approach to prediction. In this approach, we choose a set $\mathcal{B} = \{b(\,;\xi),\; \xi \in \Xi\}$ called a **basis** or a **dictionary**. $\mathcal{B}$ is parametrized by $\xi \in \Xi$; $\Xi$ can be finite or infinite. The elements of $\mathcal{B}$ are called **basis functions**. The predictor class $\mathcal{F}$ is the set of linear combinations of elements of $\mathcal{B}$.

**Example**

- Fourier basis, wavelet bases
- spline families
- two layer linear output neural networks

## Classifiers with real-valued output

### Binary classification

- Since $y \in \{\pm 1\}$, naturally $f : \mathbf{X} \to \{\pm 1\}$
- But sometimes we prefer a classifier $f : \mathbf{X} \to \mathbb{R}$ (from a predictor class $\mathcal{F}$ of real-valued functions)
- In this case, the prediction $\hat{y}$ is usually

$$\hat{y} \;=\; \operatorname{sgn}(f(x)) \tag{7}$$

  This is sometimes known as the sign trick.

Examples of real-valued classifiers

- Logistic Regression
- Naive Bayes
  in both of the above, $f(x) = P[Y = 1|X = x] \in [0, 1]$. Hence

$$\hat{y} \;=\; \operatorname{sgn}\left(f(x) - \frac{1}{2}\right) \tag{8}$$

- Support Vector Machines
- Kernel classifiers
- Neural Networks

Sign trick

The *sign* function $\operatorname{sgn}(y) = y/|y|$ if $y \neq 0$ and 0 iff $y = 0$ turns a real valued variable $Y$ into a discrete-valued one.

## Why real valued $f$?

- for statistical models $f(x) = P[Y = 1 \mid X = x]$ Example: Logistic regression
- for non-statistical models, $|f(x)|$ measures **confidence** in prediction $\hat{y}$, with $|f(x)| \approx 0$ meaning low confidence. Example: SVM
- if $f$ is differentiable[1], the gradient $\nabla f$ is used in **learning algorithms** Examples: Logistic Regression, neural networks, some forms of linear regression such as Lasso

**The margin** (assuming $y \in \{\pm 1\}$)

- The **margin** of a classifier $f$ at point $x \in \mathbf{X}$ is defined as

$$z \;=\; yf(x). \tag{9}$$

- Note that $\mathrm{sgn}(z) = y\hat{y}$.

- If $z > 0$, $\hat{y} = y$ and $f(x)$ is correct
- If $z \gg 0$, then $f(x)$ is correct, and classifier has high confidence
- If $z < 0$, then $f(x)$ is incorrect, and $|z|$ measures "how wrong" is $f$ on this $x$
- Note also that $z \approx 0$ means that the classification $\hat{y}$ is not robust, whether correct or not

[1] and $\nabla f$ not 0 almost everywhere

## Real valued multi-way classifiers

- We train $m$ classifier $f_{1:m} : \mathbf{X} \rightarrow \mathbb{R}$. Then (typically)

$$\hat{y} \;=\; \operatorname*{argmax}_{c=1:m} f_{1:m}(x). \tag{10}$$

- $\hat{y} = y$ means the classifier is correct
- the training can be done
  - independently for each $f_c$, $c = 1 : m$ (e.g. generative classifiers – in Lecture II)
  - or at the same time (e.g. neural networks, SVM)

- The **margin** is defined as

$$z(x) \;=\; f_y - \max_{c \neq y} f_c(x) \tag{11}$$

In other words

- if $\hat{y} = y$ (correct), then $z = f_{\text{true}} - f_{\text{nextbest}} \; > 0$
- if $\hat{y} \neq y$ (mistake), then $z = f_{\text{true}} - f_{\hat{y}} \; < 0$ (since $f_{\hat{y}}(x)$ is the max of $f_c(x)$)

## Decision regions, decision boundary of a classifier

Let $f(x)$ be a classifier (not necessarily binary)

- $\hat{y}(x)$ takes a finite set of values
- The **decision region** associated with class $y$ = the region in $X$ space where $f$ takes value $y$, i.e. $D_y = \{x \in \mathbb{R}^d,\, f(x) = y\} = f^{-1}(y)$.
- The boundaries separating the decision regions are called **decision boundaries**.

## Decision regions, decision boundary of a classifier

Let $f(x)$ be a classifier (not necessarily binary)

- $\hat{y}(x)$ takes a finite set of values
- The **decision region** associated with class $y$ = the region in $X$ space where $f$ takes value $y$, i.e. $D_y = \{x \in \mathbb{R}^d,\ f(x) = y\} = f^{-1}(y)$.
- The boundaries separating the decision regions are called **decision boundaries**.

- For a binary classifier, we have two decision regions, $D_+$ and $D_-$. By convention $f(x) = 0$ on the decision boundary.
- For binary classifier with real valued $f(x)$ (i.e $\hat{y} = \mathrm{sgn} f(x)$) we define $D_+ = \{x \in \mathbb{R}^d,\ f(x) > 0\}$, $D_- = \{x \in \mathbb{R}^d,\ f(x) < 0\}$ and the decision boundary $\{x \in \mathbb{R}^d,\ f(x) = 0\}$