Predictors

Jong-Han Kim

LEO6006 Data Science and Machine Learning for Aerospace Systems

Inha University

Predictors

Data fitting

- we think $y \in \mathbf{R}^m$ and $x \in \mathbf{R}^d$ are (approximately) related by

$$y \approx f(x)$$

- x is called the *independent variable* or *feature vector*
- y is called the *outcome* or *response* or *target* or *label* or *dependent variable*
- very often $m = 1$, *i.e.*, the outcome is scalar
- y is something we want to predict, given x
- we don't know the 'true' relationship between x and y

Features

often x is a vector of features:

- documents
  - x is word count histogram for a document
- patient data
  - x are patient attributes, test results, symptoms
- customers
  - x is purchase history and other attributes of a customer

Where features come from

- we use u to denote the raw input data, such as a vector, word or text, image, video, audio, ...
- $x = \phi(u)$ is the corresponding *feature vector*
- the function ϕ is called the *embedding* or *feature function*
- ϕ might be very simple or quite complicated
- often we take $\phi(u)_1 = x_1 = 1$, the *constant feature*
- similarly, the raw output data v can be featurized as $y = \psi(v)$
- (much more on these ideas later)

Data and prior knowledge

- we are given data $x^1, \ldots, x^n \in \mathbf{R}^d$ and $y^1, \ldots, y^n \in \mathbf{R}$
- (x^i, y^i) is the ith *data pair* or *observation* or *example*
- collectively we call $x^1, \ldots, x^n$ and $y^1, \ldots, y^n$ a *data set*

- we also (might) have *prior knowledge* about what f might look like
  - *e.g.*, f is smooth or continuous: $f(x) \approx f(\tilde{x})$ when x is near $\tilde{x}$
  - or we might know $y \geq 0$

Predictor

- we seek a *predictor* or *model* $g : \mathbf{R}^d \to \mathbf{R}$
- for feature vector x, our prediction (of y) is $\hat{y} = g(x)$
- predictor g is chosen based on both data and prior knowledge
- in terms of raw data, our predictor is
$$\hat{v} = \psi^{-1}(g(\phi(u)))$$
(with a slight variation when ψ is not invertible)

- $\hat{y}^i \approx y^i$ means our predictor does well on ith data pair
- *but our real goal is to have $\hat{y} \approx y$ for (x, y) pairs we have not seen*

Information flow

raw data	*feature*	*outcome*	*raw output*

$$u \longrightarrow \boxed{\phi(\cdot)} \xrightarrow{\;x\;} \boxed{f(\cdot)} \xrightarrow{\;y\;} \boxed{\psi^{-1}(\cdot)} \longrightarrow v$$

$$x = \phi(u) \qquad\qquad y = \psi(v)$$

student	hrs. sleep	exam score	grade
	hrs. study		
	hrs. instagram		
	alcohol freq.		
	$\vdots$		

Parametrized predictors

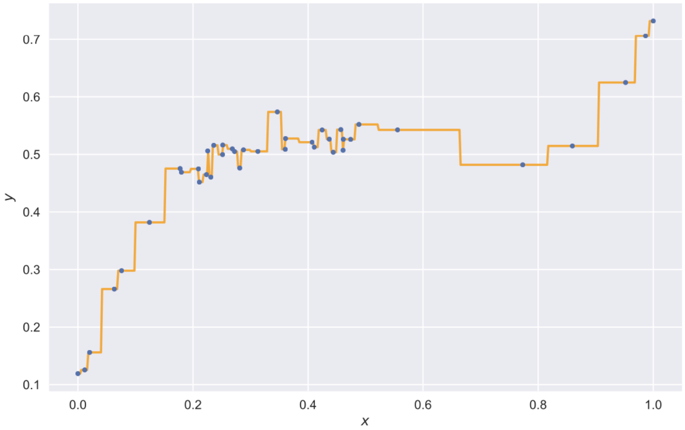
- many predictors have the form $\hat{y} = g(x;\theta)$, also written as $\hat{y} = g_\theta(x)$
- the function g fixes the *structure* or *form* of the predictor
- $\theta \in \mathbf{R}^p$ is a *parameter* (vector) for the prediction model
- choosing a particular $\theta \in \mathbf{R}^p$ is called *tuning* or *training* or *fitting* the model
- a *learning algorithm* is a recipe for choosing θ given data
- example: *linear regression model*
  - $\hat{y} = g_\theta(x) = \theta_1 x_1 + \cdots + \theta_d x_d$
  - you can fit a linear regression model using least squares
  - (and other methods too; much more on that later)

Nearest neighbor predictors

Nearest neighbor predictor

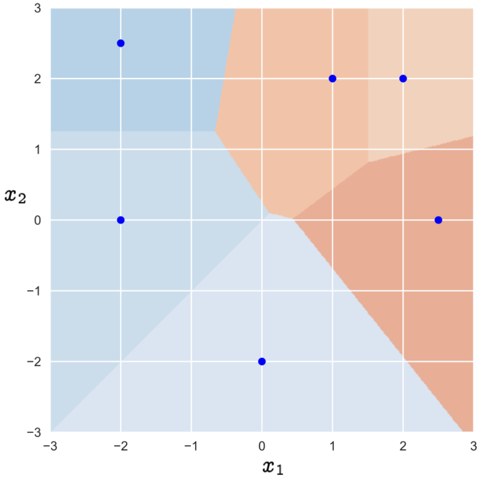
- we are given data set $x_1, \ldots, x_n$, $y_1, \ldots, y_n$
- *nearest neighbor predictor*:
  - given x, find its nearest neighbor x_i among given data
  - then predict $\hat{y} = g(x) = y_i$
- extremely intuitive
- parameter is full data set: $\theta = (x_1, \ldots, x_n, y_1, \ldots, y_n)$
- 'training' is easy; it requires no computation
- g is a piecewise constant function of x, since $g(x) = y_i$ when x is closer to x_i than the other x_js

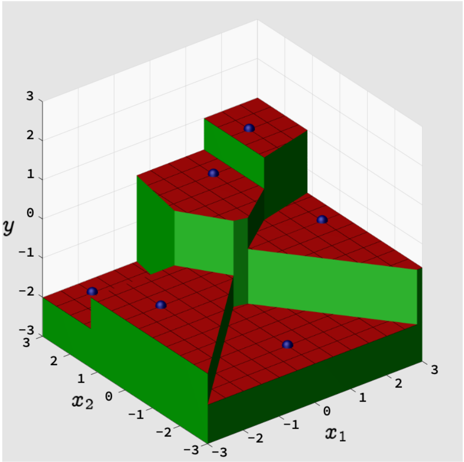
Example



- dots show data points (x_i, y_i), $x_i \in \mathbf{R}$ $(d = 1)$
- line shows $\hat{y} = g(x)$

Example





- dots show data points (x_i, y_i), $x_i \in \mathbf{R}^2$ $(d = 2)$, red surface is $\hat{y} = g(x)$

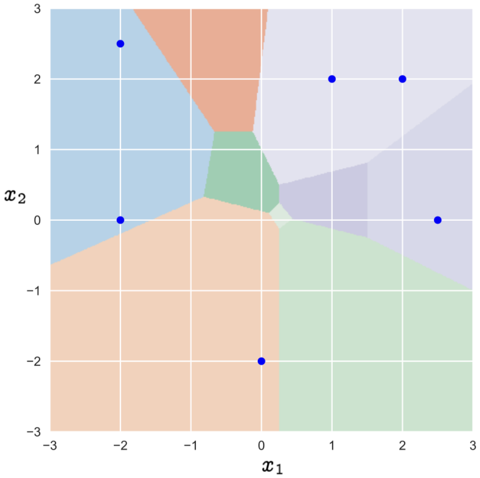
k-nearest neighbor predictor

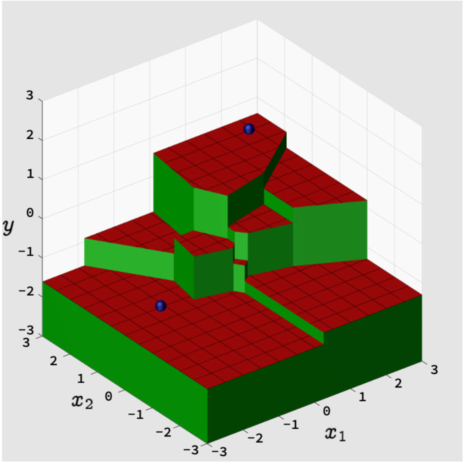
- given x, find its k nearest neighbors $x^{i_1}, \ldots, x^{i_k}$ among given data
- k-*nearest neighbor predictor* (k-NN) predicts the average of the associated outcomes

$$\hat{y} = g(x) = \frac{1}{k}(y^{i_1} + \cdots + y^{i_k})$$

- a useful generalization of nearest neighbor predictor
- many variations, *e.g.*,
  - use a weighted average to form $\hat{y}$
  - pre-process by clustering the original data set

Example: $k = 2$





Soft nearest neighbor predictor

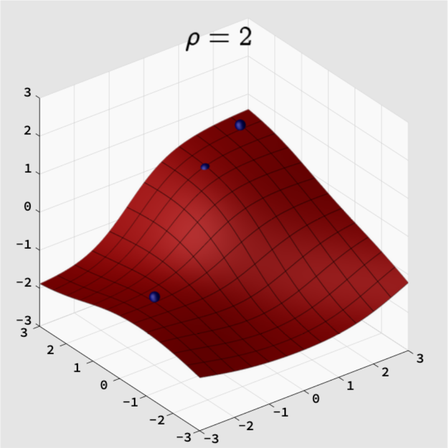
- prediction is weighted average, $\hat{y} = g(x) = \sum w^i y^i$, with weights

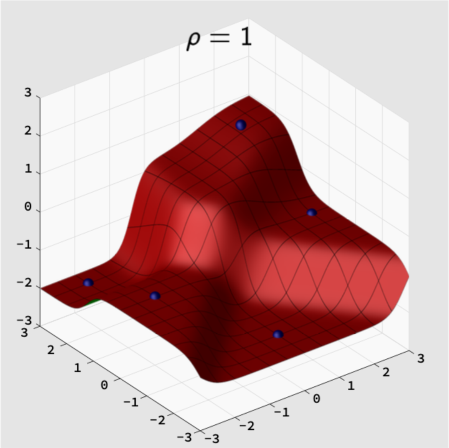
$$w^i = \frac{e^{-||x-x^i||^2/\rho^2}}{e^{-||x-x^1||^2/\rho^2} + \cdots + e^{-||x-x^n||^2/\rho^2}}$$

  that depend on x

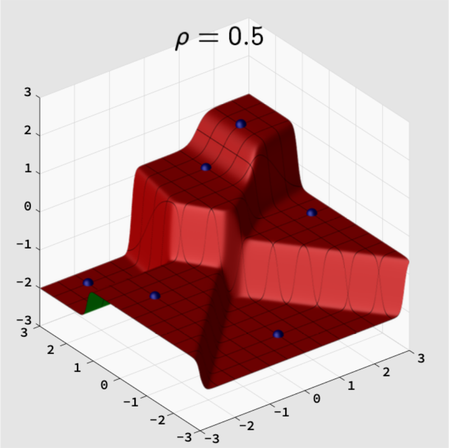
- $\rho > 0$ is a parameter, a characteristic length
- weight w^i is larger when x is near x^i
- for small ρ, this reverts to nearest neighbor predictor

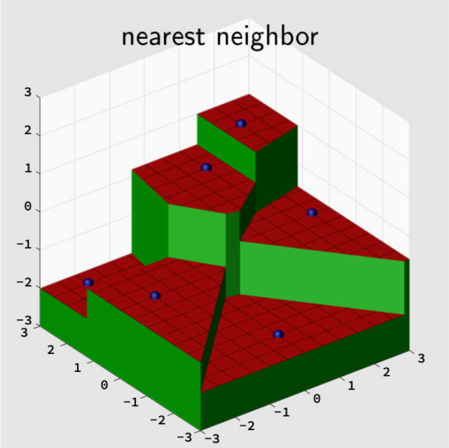
Example





Example





Linear predictors

Linear predictor

- a linear predictor has the form $g(x;\theta) = \theta^T x$
- for $m = 1$ (scalar y), parameter is a vector $\theta \in \mathbf{R}^d$
- for $m > 1$ (vector y), parameter is a matrix $\theta \in \mathbf{R}^{d \times m}$
- also called a *linear regression model*
- prediction is a linear combination of features

$$\hat{y} = g(x) = \theta_1 x_1 + \cdots + \theta_d x_d$$

  - for $m = 1$, θ_i are entries of θ
  - for $m > 1$, θ_i^T are rows of θ
- there are many ways to fit a linear regression model to data, including least squares

Interpreting a linear predictor

- we consider scalar y ($m = 1$); similar results hold for vector y
- linear predictor has form
$$\hat{y} = g(x) = \theta_1 x_1 + \cdots + \theta_d x_d$$
- θ_3 is the amount prediction $\hat{y} = g(x)$ increases when x_3 increases by 1 (particularly interpretable when x_3 is Boolean, *i.e.*0 or 1)
- $\theta_7 = 0$ means that the prediction does not depend on x_7
- θ small means predictor is insensitive to changes in x:
$$|g(x) - g(\tilde{x})| = |\theta^\mathsf{T} x - \theta^\mathsf{T} \tilde{x}| = |\theta^\mathsf{T}(x - \tilde{x})| \leq \|\theta\|_2 \, \|x - \tilde{x}\|_2$$

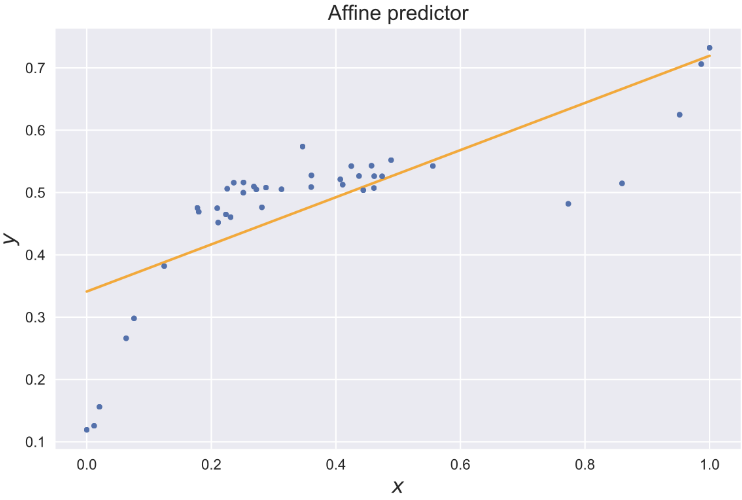
Affine predictor

- in many cases the first feature is constant, *i.e.*, $x_1 = 1$
- the linear predictor g is then an *affine function* of $x_{2:d}$, *i.e.*, linear plus a constant

$$g(x) = \theta^\top x = \theta_1 + \theta_2 x_2 + \cdots + \theta_d x_d$$

- θ_1 is called the *offset* or *constant term* in the predictor
- θ_1 is the prediction when all features (except the constant) are zero

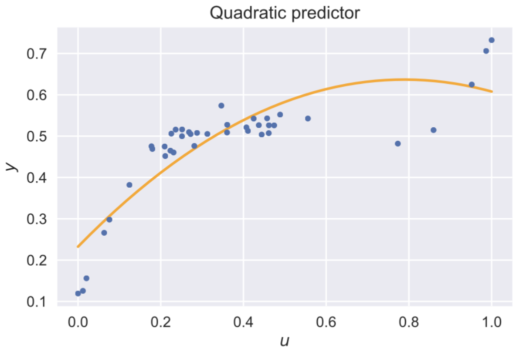
Example



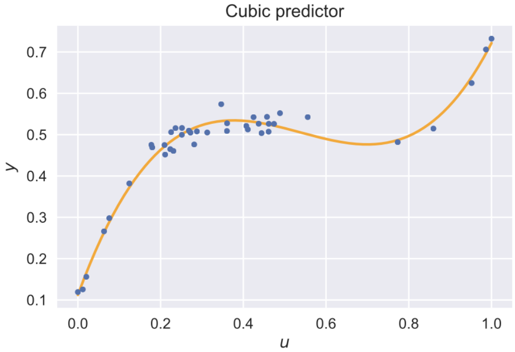
Polynomial predictor

- with appropriate embedding of u, can get nonlinear function of u with a linear predictor of x
- common example with $u \in \mathbf{R}$:
$$x = \phi(u) = (1, u, u^2, \ldots, u^{d-1})$$
($\phi : \mathbf{R} \to \mathbf{R}^d$ is called *polynomial* or *power* embedding)
- linear predictor has form
$$\hat{y} = \theta^\top x = \theta_1 + \theta_2 u + \theta_3 u^2 + \cdots + \theta_d u^{d-1}$$
- this is a linear function of x, but a polynomial function of u
- (much more on this topic later)

Example



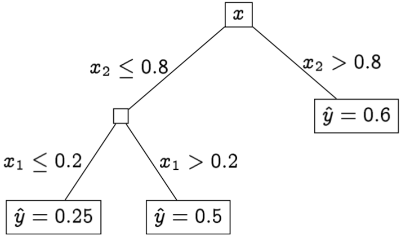
Quadratic predictor
0.7
0.6
0.5
0.4
0.3
0.2
0.1
y
0.0
0.2
0.4
0.6
0.8
1.0
u

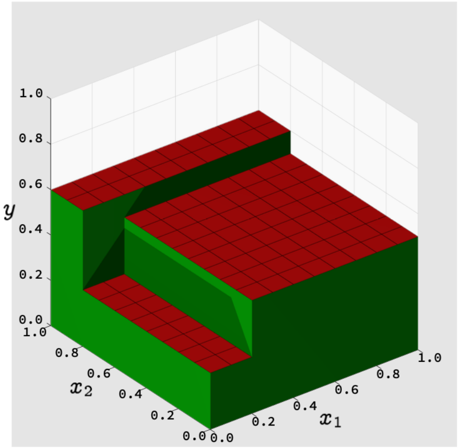


Cubic predictor
0.7
0.6
0.5
0.4
0.3
0.2
0.1
y
0.0
0.2
0.4
0.6
0.8
1.0
u

Tree-based predictors

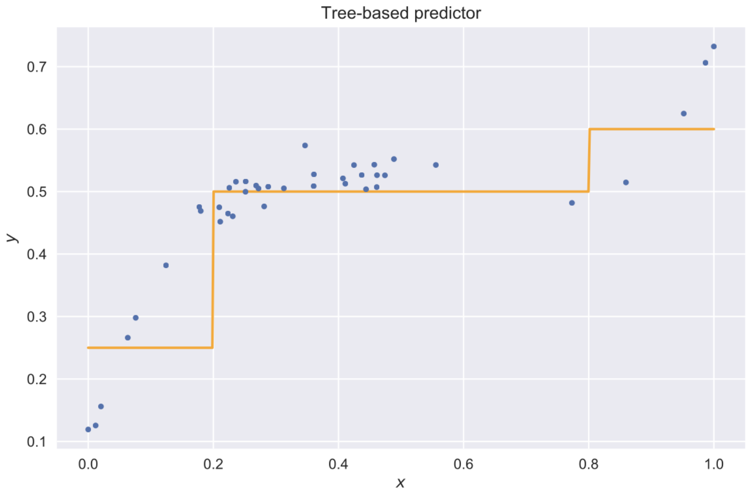
Tree-based predictors





- predictor represented by partially developed Boolean tree
- non-leaf nodes associated with an index i and threshold t
- each leaf has a value $\hat{y}$
- parameter θ encodes tree, thresholds, leaf values
- predictor is piecewise constant function of x, interpretable when the tree is small enough

Example



Neural network predictors

Neural network layers

- a (feedforward) *neural network* predictor consists of a composition of functions

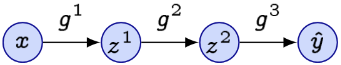
$$\hat{y} = g^3\left(g^2(g^1(x))\right)$$

  (we show three here, but we can have any number)

- written as $g = g^3 \circ g^2 \circ g^1$ (the symbol $\circ$ means function composition)
- each g^i is called a *layer*; here we have 3 layers
- we can write the predictor $\hat{y} = g^3\left(g^2\left(g^1(x)\right)\right)$ as

$$z^1 = g^1(x); \quad z^2 = g^2(z^1); \quad \hat{y} = g^3(z^2)$$

- the vector $z^i \in \mathbf{R}^{d^i}$ is called the *activation* or *output* of layer i
- we sometimes write $z^0 = x$, $d^0 = d$, and $z^3 = \hat{y}$, $d^3 = m$
  (so the predictor input x and predictor output y are also considered activations of layers)
- sometimes visualized as flow graph



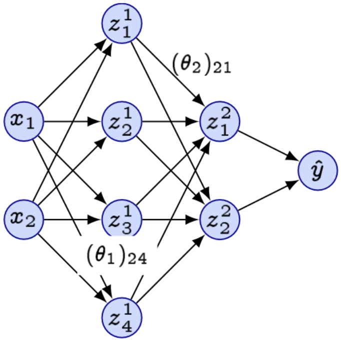
Layer functions

- each layer g^i is a composition of a function h with an affine function

$$g^i(z^{i-1}) = h\left(\theta_i^\mathsf{T}(1, z^{i-1})\right)$$

- the matrix $\theta_i \in \mathbf{R}^{(d^{i-1}+1)\times d^i}$ is the *parameter* (also called *weights*) for layer i
- the function $h : \mathbf{R} \to \mathbf{R}$ is a scalar *activation function*, which acts elementwise on a vector argument (*i.e.*, it is applied to each entry of a vector)
- common activation functions include
  - $h(x) = (x)_+ = \max(x, 0)$, called *ReLU* (rectified linear unit)
  - $h(x) = e^x/(1 + e^x)$, called *sigmoid* function
- an M-layer neural network predictor is parameterized by $\theta = (\theta_1, \ldots, \theta_M)$ (for M layers)

Network depiction



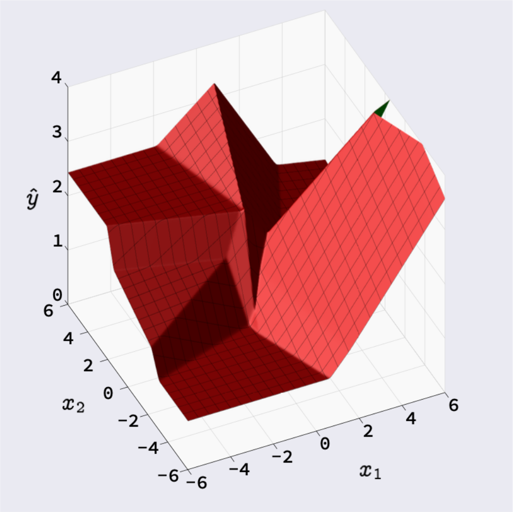
- neural networks are often represented by *network* diagrams
  - each vertex is a component of an activation
  - edges are individual weights or parameters
- example above has 3 layers, with $d^0 = 2$, $d^1 = 4$, $d^2 = 2$, $d^3 = 1$

Example

$$\theta_1 = \begin{bmatrix} 0.80 & 0.10 & 1.30 & 1.20 \\ -0.50 & 0.70 & 0.80 & 2.90 \\ -1.80 & 0.20 & -1.50 & -0.60 \end{bmatrix}$$

$$\theta_2 = \begin{bmatrix} 1.40 & 1.10 \\ -0.10 & -0.90 \\ 0.50 & 0.20 \\ -0.40 & 0.90 \\ -0.40 & -0.10 \end{bmatrix}$$

$$\theta_3 = \begin{bmatrix} 0.90 \\ 0.70 \\ 0.50 \end{bmatrix}$$



Neural network predictors

- neural network described above is called a *feedforward neural network* or *multi-layer perceptron*
- there are many variations on this basic neural network
- you'll see them in other classes

Summary

Summary

- a predictor is a function $g : \mathbf{R}^d \to \mathbf{R}^m$ meant to predict the outcome y, given feature vector x
- there are many types of predictors
  - nearest-neighbor
  - tree
  - linear
  - neural networks
  - ... and many others
- most predictors are parametrized, with the form $g_\theta(x)$
- g fixes the form of the predictor
- $\theta \in \mathbf{R}^p$ are parameters that we choose to fit the data, which is called training the predictor
- we'll see later how training is done