Intro to Machine Learning

Friday, August 23, 2024 4:45 PM

Goal is to train on training data, leading to a function (aka hypothesis), f

and often we wish this function to perform well on unseen data (i.e., to generalize)

We'll outline here the simplest ML setup, of batch (offline), statistical supervised learning

Running example: email SPAM classifier

Data "x" is an instance — ex: email — ex: person

"y" is the response / label — ex: 0 = not spam, 1 = spam — ex: income

An abstract instance (a person, an email) must be converted to a vector of features (email: → length, → % capital letters, → % misspelled words } classical ; → all the text (in order) } modern)

"feature engineering"

Very important! but more art than science (a great ML model won't matter if data are bad)

Supervised learning means we have a dataset to train on, $\{(x_i, y_i)\}_{i=1}^n$

Sometimes $y = x$ — "Self-supervised" — * important in modern ML

Ex. autoencoders
Ex. in NLP or imaging, artificially mask data.

Statistical learning means we assume our data were drawn i.i.d. from some fixed (but unknown) probability distr.

$$x, y \sim D$$

hence it's not Bayesian
We don't parameterize it

Real problems have many variants

- unsupervised, semi-supervised
- transfer learning, one-shot-learning
- online learning, adversarial learning
- covariate shift

Intro to ML (2)

Saturday, August 24, 2024 8:03 AM

Output

We learn a function f such that on (possibly new) data X,

$\hat{y} := f(X)$ is our prediction, y is true response,

and $\hat{y}$ and y are "close" to each other

i.e. want the loss function $\ell(\hat{y}; y)$ to be small

Goals (tasks)

Classification: is this email spam or not?

Binary classification vs Multiclass

ex: $\ell(\hat{y}; y) = \begin{cases} 0 & \hat{y} = y \\ 1 & \hat{y} \neq y \end{cases}$ "0-1" loss

Clustering: similar to classification but no right answer

"cluster this class into 2 groups"

ex: loss function penalizes variability within a cluster

Regression: like multiclass classification w/ ∞ classes, and being close counts. eg: predict temperature

ex: $\ell(\hat{y}; y) = (\hat{y} - y)^2$ squared loss

Generation: given data $X \sim D$, learn D

i.e., DALL-E, etc. Generate realistic artwork...

Diffusion models, GANs, VAE

Scientific use: UQ, ...

Training

The function we learn is usually parameterized

i.e., $f \in \mathcal{H}$ ← hypothesis space

or $f = f_\theta$ or f_w

parameters

Intro to ML (3)

Saturday, August 24, 2024 5:26 PM

choosing $\mathcal{H}$ is part of the art of ML

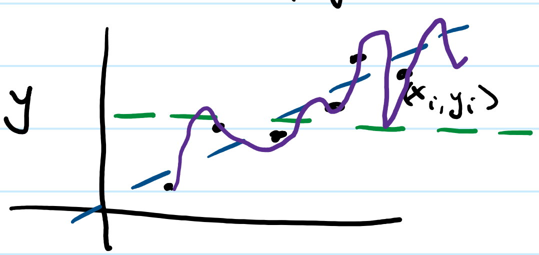
$\mathcal{H}$ too large (eg., $\mathcal{H} = \{$all functions from $\mathcal{X}$ to $\mathcal{Y}\}$)

- you can fit training data well...
- ....but unlikely to generalize to unseen data

$\mathcal{H}$ too small

- can't even fit training data, so can't hope to fit unseen data

ex: $\mathcal{H}^k = \{$all polynomials of degree k or less$\}$

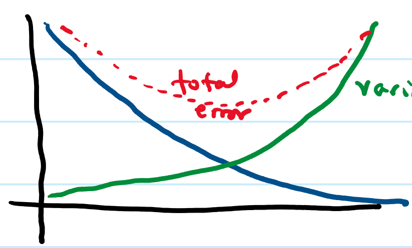


degree 0 polynomial — under fit, Too small

degree 1 polynomial — just right

degree 5 polynomial — Too large, over fit (won't generalize)

"a model that explains everything explains nothing"



how to measure? Theoretically, VC dimension

Take our "Theory of ML" course!

Practically, often use # of trainable parameters ("degrees of freedom")

How to evaluate?

We often care about

"true risk" $R(f) := \mathbb{E}_{(x,y)\sim\mathcal{D}}\, \ell(\underbrace{f(x)}_{=\hat{y}};y)$

typically unobservable

The sample/empirical risk is $\hat{R}_S(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i);y_i)$

where the dataset is $S = \{(x_i,y_i)\}_{i=1}^{n}$