5: Machine Learning Basics

Saturday, June 9, 2018 5:55 PM

Hyperparameters: determined outside the learning algorithm

ML is form of applied statistics:

↑ emphasis on using computation to estimate functions

↓ emphasis on proving confidence intervals

most deep learning algorithms based on stochastic gradient descent
(optimization algorithm)

(5.1) Learning Algorithms

(5.1.1) The Task: T (what we want to learn to do)

- how the ML system should process an example

example: collection of features - quantitatively measured

$x \in \mathbb{R}^n$ feature vector

Examples of ML tasks:

- Classification: learn $f: \mathbb{R}^n \to \{1, \ldots, k\}$ ($\mathbb{R}^n$: feature vector; k output classes)

  $y = f(x)$

  e.g., object recognition

  variation: output probability distribution

- Classification with missing inputs:

  learn a set of functions, one for each possible set of input features

  e.g. - medical diagnosis: only have results of some tests

- Regression: learn $f: \mathbb{R}^n \to \mathbb{R}$

  e.g. - predict expected claim amount

- Transcription: unstructured → structured data (text)

  e.g., OCR (image → text), speech recognition (sound → text)

- Machine translation e.g., English → Russian
- Structured output e.g., Parsing, image/video segmentation
- Anomaly detection (really, a kind of classification problem)
- Synthesis and sampling e.g., speech synthesis
- Imputation of missing values
- Denoising corrupted example → clean example
- Density estimation or probability mass function estimation

...

(5.1.2) Performance measure: P

classification: accuracy, error rate
(expected 0-1 loss; ↑ correctly classified, ↑ incorrect)

for continuous tasks,

average log-probability

often difficult to choose P.

(5.1.3) Experience: E

unsupervised: experience of dataset, but no given labels
attempt to learn structure, probability distribution

supervised: each example has an associated label or target

difference is not fully defined

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \dots, x_{i-1})$$

decompose feature vector
turns unsupervised problem of modeling $p(x)$ into n supervised learning problems

or: supervised learning problem: learn $p(y \mid x)$
(labels ← feature vector)

$$p(y \mid x) = \frac{p(x, y)}{\sum_{y'} p(x, y')}$$

reinforcement learning: algorithms interact with environment ⇝ feedback loop
instead of fixed dataset

dataset: design matrix – each row is one example, feature vector (+ label if supervised)
assumes same size for each example

(5.1.4) Example: Linear Regression

Input: $x \in \mathbb{R}^n$

Output: $y \in \mathbb{R}$
scalar

w_i is coefficient for x_i

$\hat{y} = w^T x$.
($\hat{y}$: prediction; $w \in \mathbb{R}^n$ is learned vector of parameters (weights))

measure of performance: mean squared error

$$MSE_{test} = \frac{1}{m} \sum_i \left(\hat{y}^{(test)} - y^{(test)}\right)_i^2 = \frac{1}{m} \left\| \hat{y}^{(test)} - y^{(test)} \right\|_2^2$$

($\hat{y}^{(test)}$: prediction; $y^{(test)}$: actual)

simple ML training algorithm: minimize MSE on training data

$$\nabla_w MSE_{train} = 0 \quad \text{solve for gradient} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} \left\| X^{(train)} w - y^{(train)} \right\|_2^2 = 0.$$

$$\nabla_w \left(X^{(train)} w - y^{(train)}\right)^T \left(X^{(train)} w - y^{(train)}\right) = 0$$

$$\Rightarrow \nabla_w \left(w^T X^{(train)T} X^{(train)} w - 2 w^T X^{(train)T} y^{(train)} + y^{(train)T} y^{(train)}\right) = 0$$

$$\Rightarrow 2 X^{(train)T} X^{(train)} w - 2 X^{(train)T} y^{(train)} = 0$$

$$w = \left(X^{(train)T} X^{(train)}\right)^{-1} X^{(train)T} y^{(train)}$$

"normal equations"

extension to affine function:

$\hat{y} = w^T x + b$. bias parameter – without any input, output biased to b.
(not statistical bias)

$\hat{y} = w^\top x + b$. bias parameter — when all input, output biased by b. (not statistical bias)

(note: can just add an extra entry to x that is always 1.)

so, typically call affine - "linear"

(5.2) Capacity, Overfitting, Underfitting

generalization - ability to perform well on previously unobserved inputs

training error: error on training data (can always find a model that makes this 0)

testing/generalization error: this is what separates ML from optimization
expected value of the error on new input

statistical learning theory

assumes data generating process (for both training and test data) follow

i.i.d. assumptions

↑ ↖ **identically distributed** - training & test sets drawn from same probability distribution

independent - each example in dataset is independent from others

shared underlying distribution, p_{data}

with i.i.d. assumption, randomly selected model: expected training error = expected test error

but, with model learned on training data set: training error $\leq$ expected test error

central challenges of ML:

1. make training error small ← avoid underfitting
2. make gap between training & test error small ← avoid overfitting.

capacity - ability to fit a wide variety of functions

high capacity → overfitting memorize training set

low capacity → underfitting

hypothesis space - set of functions a learning algorithm may select

e.g., linear regression: all linear functions

increase capacity → all polynomials of degree 2

$$\hat{y} = b + w_1 x + w_2 x^2$$

note: still linear func. of parameters, so can use closed form lin. reg solution

ML algorithms perform best when capacity is appropriate for true complexity of the task

representational capacity / **effective capacity**

functions that can be represented

functions that can actually be learned

Occam's razor – principle of parsimony
among competing hypotheses that match observations, pick simplest

Vapnik-Chervonenkis (VC) dimension – measure of capacity
of binary classifier: largest m for which there
exists a training set of m different x points that
the classifier can label arbitrarily.

theoretical results in statistical learning theory
bound difference between training error & generalization error
based on model capacity (upper), # of training examples (lower)
bounds rarely useful for deep learning –
- difficult to determine capacity
effective capacity limited by optimization algorithm
little theoretical understanding of general nonconvex optimization used in deep learning

nonparametric models – not limited to learning finite parameter vector
nearest neighbor regression – store training set X, y, lookup closest
$\hat{y} = y_i$ where $i = \arg\min \| X_{i,:} - x \|_2^2$.
achieves minimum training error (if break ties)

wrapping parametric = outer loop that increases polynomial degree for linreg

Bayes error – error incurred by oracle making predictions from true distribution

(5.2.1) No Free Lunch Theorem (Wolpert 1996)

averaged over all possible data-generating distributions, every classification
algorithm has the same error rate on previously unobserved points.
but – real data is not generated arbitrarily

(5.2.2) Regularization

NFL ⇒ must design ML algorithms to perform well on a specific task
idea: give preference to functions with certain properties
(will only prefer others, if they fit data much better)
e.g., weight decay for lin reg:
$J(w) = \text{MSE}_{\text{train}} + \lambda w^T w$ minimize $J(w)$ to have preference for weights

$$J(w) = MSE_{train} + \lambda w^T w$$

minimize $J(w)$ to have preference for weights with smaller L^2 norm

↑ parameter that controls strength of preference

regularizer - penalty added to cost function

regularization - any modification made to a learning algorithm that is intended to reduce generalization error (but not training error).

philosophy of deep learning: a wide range of tasks (such as all the intellectual tasks that people can do) may all be solved effectively using very general-purpose forms of regularization.