5.7 Supervised Learning Algorithms

Tuesday, June 12, 2018 9:08 PM

(5.7.1) Probabalistic Supervised Learning

Use maximum likelihood estimation to find best parameters θ for a family of distributions $p(y|x;\theta)$.

Linear regression: $p(y|x;\theta) = \mathcal{N}(y;\theta^T x, I)$

Gaussian distribution over y with mean $\theta^T x$, covariance I

↳ generalize to classification by defining different family of probability distributions (for binary class, just output probability of class 0)

squash output of linear function into interval $(0,1)$ using logistic sigmoid

interpret as probability: $p(y=1|x;\theta) = \sigma(\theta^T x)$.

logistic regression - but, it's really used for classification.

no closed form solution for optimal weights

search for maximizing log-likelihood → use gradient descent to minimize $-\log$ likelihood.

(5.7.2) Support Vector Machines (Boser et al., 1992)

like logistic reg., driven by linear function $w^T x + b$.

unlike logistic reg., does not provide probabilities - only outputs class.

if $w^T x + b$ positive → positive class

$w^T x + b$ negative → negative class

kernel trick - observe, many ML algorithms can be written as dot products:

e.g., $w^T x + b = b + \sum_{i=1}^{m} \alpha_i x^T x^{(i)}$

(α_i: coefficient; $x^{(i)}$: training example)

↳ replace x with output of a feature function $\phi(x)$.

dot product with $k(x, x^{(i)}) = \phi(x) \cdot \phi(x^{(i)})$ — inner product: $\phi(x)^T \phi(x^{(i)})$

dot product with $k(x, x^{(i)}) = \phi(x) \cdot \phi(x^{(i)})$ — inner product. $\phi(x)^T \phi(x^{(i)})$

called a "kernel"

(note: SVM variations use different inner products, not always Σ)

$$f(x) = b + \sum_i \alpha_i k(x, x^{(i)})$$

So, equiv to preprocessing data with $\phi(x)$, then learn linear model

why powerful?

① can learn nonlinear (in x) models using convex optimization
crawings quickly - because ϕ is fixed, only optimize α.
decision function is linear in new space

② $k()$ can often be implemented efficiently (even when $\phi(x)$ is intractable)

Gaussian kernel (most common) a.k.a. radial basis function kernel

$$k(u, v) = \mathcal{N}(u - v; 0, \sigma^2 I)$$

$\mathcal{N}(x; \mu, \Sigma)$ is standard normal density

dot product in infinite dimensional space!

many models can be enhanced using kernel tricks "kernel machines"

main drawback:

cost to evaluate decision function is linear in # of training examples

each example contributes term $\alpha_i k(x, x^{(i)})$ to decision function

(can mitigate by having most $\alpha_i \to 0$) non-zero training examples are "support vectors"

(also: do not generalize well)

(5.7.3) Other Supervised Learning Algorithms

k-nearest neighbors

as # of training $\to \infty$, error (0-1 loss) converges to 2x Bayes error
→ because of breaking ties (equal distance) randomly
Cannot learn which features are discriminative

decision tree

(5.8) Unsupervised Learning Algorithms

(5.8) Unsupervised Learning Algorithms

difference murky - is a feature value provided by "supervisor"
  informally - no need for human annotation

find "best" representation of data
  - preserve (relevant) information, while maximizing some other property (e.g., simplicity)
    - lower dimensions
    - sparse
    - independent representations

(5.8.1) Principal Component Analysis

learn representation with lower dimensionality and no linear correlations (but not independent - would need to remove non-linear correlations also)

(5.8.2) k-means clustering

k centroids $\{\mu^{(1)}, \ldots, \mu^{(k)}\}$

repeat until converges:
  - assign each training example to nearest centroid
  - update centroid $\mu^{(i)}$ to mean of all examples assigned to it

problem is ill-posed: how to measure distance
  without understanding space, no way to ensure meaningful clusters