18. Linear Predictors (part 1: binary classification)

Monday, February 19, 2024 4:28 PM

ch 9: §9.1 Binary classificatn
§9.2 (linear) regression
§9.3 logistic regression

Starting Part Two of SS, "From Theory to Applications"

Usefulness of linear predictors

- used in practice (or their generalizations like SVM)
- can be supercharged via kernel methods
- easy to understand, interpret and train
- used for classification and regression

we really should say "affine"

Def Linear classifier space L_D

$L_d = \{ h_{\omega,b} : \omega \in \mathbb{R}^d, b \in \mathbb{R} \}$, $h_{\omega,b}(x) = \langle \omega, x \rangle + b$ is affine

(often work in $d+1$ dim and take $b=0$ wlog, i.e. $x_{(d+1)} := 1$, $\omega_{(d+1)} = b$

Later, we'll take $\mathcal{H} = \phi \circ L_d$ where ϕ is identity for regression
$\phi = \text{sign}$ for classification

§9.1 Binary Classification and Halfspaces

$HS_d := \phi \circ L_d$, $\phi = \text{sign}$.

We've seen $\text{VCdim}(HS_d) = d+1 < \infty$ so it's PAC learnable

(need $m = \Omega\left(\frac{d \cdot \log(1/\varepsilon) + \log(1/\delta)}{\varepsilon} \right)$ samples)

$m = \Omega(g(m))$ means $g(m) = O(m)$

... and ERM is a PAC learner!

How to solve ERM for HS_d ?

If data isn't separable (ie. not realizable) it can be very hard. In practice, use a surrogate loss function that's at least continuous (unlike 0-1 loss), eg. hinge (SVM) or logistic

So for now assume data is separable
and wlog look at homogeneous ($b=0$) case

Mostly of theoretical and historical interest

So assume $\exists\, \omega_{\text{oracle}}$ st. $y_i = \text{sign}(\langle \omega_{\text{oracle}}, x_i \rangle)$

Find ω st.

$$y_i = \text{sign}(\langle \omega, x_i \rangle) \iff y_i \cdot \langle \omega, x_i \rangle > 0$$

18a. Linear Predictors (part 1: binary classification)

Monday, February 19, 2024 4:44 PM

Idea: solve this via linear programming $\left(\min_{w} \; c^T w \quad \text{s.t. } A\cdot w = b, \; C\cdot w \geq d \right)$

which are efficiently solvable in both theory and practice

sort of... depends on your model of computation.

But... our constraint was $\ldots > 0$ not $\ldots \geq 0$. In optimization, that's a big deal!

Fix: positive scaling of w won't affect final output

so $\forall w$ that correctly classifies the data, i.e. $y_i \langle w, x_i \rangle > 0$

$\exists$ a scaled version $\tilde{w} = c \cdot w$ $(c > 0)$ st.

$$y_i \cdot \langle \tilde{w}, x_i \rangle \geq 1$$

i.e. $c = \dfrac{1}{\min_i \langle w, x_i \rangle}$

Alternative (historical) method: Rosenblatt's "Perceptron" for ERM, 1958

"Batch perceptron": Data input $((x_i, y_i))_{i=1}^m$ $x_i \in \mathbb{R}^d$, $y_i = \pm 1$

$w^{(0)} = 0 \in \mathbb{R}^d$

For $t = 1, 2, \ldots$

If $\exists\, i \in [m]$ s.t. $y_i \langle w^{(t)}, x_i \rangle \leq 0$ i.e. misclassified

$w^{(t+1)} = w^{(t)} + y_i \cdot x_i$ pushes in right direction on x_i

Else

Done! 0 training loss

Note $\underbrace{y_i \langle w^{(t+1)}, x_i \rangle}_{\text{NEW}} = \underbrace{y_i \langle w^{(t)}, x_i \rangle}_{\text{OLD}} + \underbrace{\|x_i\|^2}_{\text{pushes it positive.}}$

Thm Perceptron converges (if data separable)

One way to prove: exercise 14.3, this is subgradient descent on

$$f(w) = \max_{i \in [m]} \left(1 - y_i \langle w, x_i \rangle \right)$$

max preserves convexity

no gradient if argmax not unique.

Def subgradient If f is convex, $f: \mathbb{R}^d \to \mathbb{R}$, then $g \in \mathbb{R}^d$ is a subgradient of f at w if $\forall w'$, $f(w') \geq f(w) + \langle g, w' - w \rangle$

Sometimes we go in depth on convexity here...