Geoff Hinton lecture notes

RMS Prop (2012)

attempted to fix "prob" w/ Adagrad of inevitable learning rate decay

standard to take $\beta = 0.9$

Idea: $$D_t = \beta D_{t-1} + (1-\beta)\,\text{diag}\left(\nabla f_t(\omega_t)\,\nabla f_t(\omega_t)^T\right)$$

vs

$$D_t = D_{t-1} + \text{diag}\left(\nabla f_t(\omega_t)\,\nabla f_t(\omega_t)^T\right)$$ for Adagrad

Let $g_t = \nabla f_t(\omega_t)$ be our gradient or stochastic gradient estimate at time t, then

$$\text{diag}\left(g_t g_t^T\right) = \begin{bmatrix} (g_t)_1^2 & & \\ & \ddots & \\ & & (g_t)_d^2 \end{bmatrix}$$

e.g.

$$D_3 = .9 D_2 + .1 \operatorname{diag}(g_3 g_3^T)$$

$$= .9(.9 D_1 + .1 \operatorname{diag}(g_2 g_2^T)) + .1 \operatorname{diag}(g_3 g_3^T)$$

$$= (.9)^2 D_1 + (.9)(.1) \operatorname{diag}(g_2 g_2^T) + .1 \operatorname{diag}(g_3 g_3^T)$$

This update prevents inevitable learning rate decay

$$\omega_{t+1} = \omega_t - \alpha_t (D_t + \varepsilon I)^{-1/2} \nabla f_t(\omega_t)$$

Kingma

ADAM (2014) very popular in deep learnings

- tries to fix the diminishing step size problem of Adagrad
- also tries to estimate a better search direction using momentum (exponential average)

$$\mu_t = \beta_1 \mu_{t-1} + (1-\beta_1) \nabla f_t(\omega_t)$$

$$D_t = \beta_2 D_{t-1} + (1-\beta_2) \operatorname{diag}\left(\nabla f_t(\omega_t) \nabla f_t(\omega_t)^T \right)$$

$$\omega_{t+1} = \omega_t - \alpha_t \left(D_t + \varepsilon I \right)^{-1/2} \mu_t$$

General approach to nonlinear ML : Kernel Methods

Instead of models of the form $f(x) = \sum_{i=1}^{d} w_i x_i + w_0$

(linear models) we want to use/fit

nonlinear models $f(x) = ?$

As our working example, consider quadratic model of the form

$$f(x) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i,j=1}^{d} w_{i,j} x_i x_j$$

nonlinear (quadratic) interaction of features

Note the parameter "vector" is $[w_0, \vec{w}, W]$ with $w_0 \in \mathbb{R}$, $\vec{w} \in \mathbb{R}^d$, $W \in \mathbb{R}^{d^2}$

$1 + d + d + \binom{d}{2}$

so we have $\Omega(d^2)$ parameters

note $f(x) = \omega_0 + \langle \omega, x \rangle + x^T W x$

Consider the case of using $f(x)$, a quadratic model, for regression. We use LS loss and ℓ_2 regularization.
Flatten our model parameters into a vector $v \in \mathbb{R}^{1+2d+\binom{d}{2}}$ and write

$$f(x) = v^T \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \\ x_1^2 \\ x_1 x_2 \\ \vdots \\ x_d^2 \end{bmatrix} = v_0 + \sum_{i=1}^{d} v_i x_i + \sum_{i,j=1}^{d} v_{d+(i-1)d+j} \, x_i x_j$$

so $v_0 = \omega_0$

$v_{1:d} = \omega$

$v_{(d+1):\left(1+2d+\binom{d}{2}\right)} \overset{\text{reshape}}{=} W$

$= v^T \phi(x)$

where $\phi : \mathbb{R}^d \to \mathbb{R}^{1+d+\binom{d}{2}}$

$$\phi\left(\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}\right) = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \\ x_1^2 \\ x_1 x_2 \\ \vdots \\ x_d^2 \end{bmatrix}$$

is called our <u>feature map</u>

don't penalize the bias

$\downarrow$ v_0

and we want to solve the ERM problem

$$v_* = \operatorname*{argmin}_v \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \|v_{1:}\|_2^2$$

$$= \operatorname*{argmin}_v \frac{1}{n}\sum_{i=1}^{n}(y_i - v^T\phi(x_i))^2 + \lambda \|v_{1:}\|_2^2$$

Define feature matrix $\Phi = \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_n)^T \end{bmatrix} \in \mathbb{R}^{n \times D}$

$\longrightarrow D = 1 + d + \binom{d}{2}$

then we have

$$v_* = \operatorname{argmin} \frac{1}{n} \| y - \Phi v \|_2^2 + \lambda \| v_{1:} \|_2^2$$

$$v_* = (\Phi^T \Phi + n\lambda I)^{-1} \Phi^T y$$ ← (assuming we regularized w/ $\lambda \|v\|_2^2$)

$D \times D = \mathcal{O}(d^2 \times d^2)$

Now we can predict

$$f(x_{new}) = v_*^T \phi(x_{new})$$

drawback: using Φ and solving the linear system requires inverting a $D \times D$ matrix which is infeasible if $d > 10K$

Σ "$d^2 \times d^2$"

remedy: I only care about prediction; we can do this much faster.

$$f(x_{new}) = v_*^T \phi(x_{new}) = \phi(x_{new})^T \left(\Phi^T\Phi + n\lambda I\right)^{-1} \Phi^T y$$

observe (from the SVD) that for any matrix M

$$(M^TM + n\lambda I)^{-1} M^T = M^T (MM^T + n\lambda I)^{-1}$$

$$\Rightarrow f(x_{new}) = \phi(x_{new})^T \Phi^T \underbrace{\left(\Phi\Phi^T + n\lambda I\right)^{-1}}_{n \times n} y$$

Implies prediction is cheaper than the naive approach when $n \ll D$

Let's introduce the <u>kernel function</u> $K(x, y)$ (lowercase kappa)

where $x, y \in \mathbb{R}^d$

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

associated w/ the feature map ϕ

Further introduce the <u>kernel matrix</u> on our training data

$$[K]_{ij} = K(x_i, x_j) \quad \text{where} \quad K \in \mathbb{R}^{n \times n}$$

(K: uppercase letter kappa)

Now note that with this notation,

$$\phi(x_{new})^T \Phi^T = \phi(x_{new})^T \left[\phi(x_1) \,\vdots\, \cdots \,\vdots\, \phi(x_n) \right]$$

$$= \left[K(x_{new}, x_1) \,\vdots\, \cdots \,\vdots\, K(x_{new}, x_n) \right]$$

and

$$(\Phi\Phi^T + n\lambda I)^{-1} y$$

$$(\Phi\Phi^T)_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = K(x_i, x_j)$$

$$\Leftrightarrow \Phi\Phi^T = K$$

so

$$f(x_{new}) = \left[K(x_{new}, x_1) \,|\, \cdots \,|\, K(x_{new}, x_n) \right] (K + n\lambda I)^{-1} y$$

Letting $\alpha = (K + n\lambda I)^{-1} y$ be precomputed, we can predict by forming

$$f(x_{new}) = \sum_{i=1}^{n} K(x_{new}, x_i)\alpha_i$$

we call f a kernel estimator

Comparison

for linear models $f(x_{new}) = \langle \alpha, \overbrace{x}^{\mathbb{R}^d}_{new} \rangle$

for nonlinear models, our feature map ϕ induces a kernel K and our predictions take the form

$$f(x_{new}) = \langle \begin{bmatrix} K(x_{new}, x_1) \\ \vdots \\ K(x_{new}, x_n) \end{bmatrix}, \alpha \rangle$$

$\mathbb{R}^n \longrightarrow$

We presented this result for a specific loss function and model class
(least squares) (quadratic)

In general we can use the kernel trick idea to kernelize machine learning problems for a large class of hypothesis spaces w/ specific regularizers

Kernel Methods

(RKHS)

$\mathcal{H}$ needs to a reproducing kernel Hilbert space and the regularizer $R(f) = \|f\|_{\mathcal{H}}^2$ where $\|\cdot\|_{\mathcal{H}}$ is the Hilbert space norm.

Then the ERM

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2$$

has a solution of the form

$$f^*(x) = \sum_{i=1}^{n} K(x, x_i) \alpha_i$$

where K is the kernel associated with $\mathcal{H}$ (K & $\mathcal{H}$ determine each other)