Shiraz University

گزارش پروژه

دانشجو:

سارا لیمویی، ۹۶۳۲۴۲۱

استاد:

دکتر طاهری

تابستان ۱۴۰۰


بسم الله الرحمن الرحیم

فهرست مطالب

پیش گفتار

در این پروژه قصد داریم تا از یک روش جدید برای بالا بردن معیار k-accuracy در دسته بندی داده ها استفاده کنیم. در دسته بندی داده ها یا classification، روش های مختلفی از جمله نزدیکترین k همسایه یا k nearest neighbor (KNN)، شبکه های عصبی یا neural networks و ... پیاده سازی شده اند.

روش کار KNN معمولی

در این روش به دنبال آن هستیم تا دسته بندی داده های را بر اساس k نزدیکترین همسایه ی آن که بر مبنای معیار تفاوت دو داده و یا شباهت آن ها محاسبه می شوند را بدست بیاوریم به این صورت که فاصله ی هر کدام از داده های موجود را با هر کدام از داده های دیگر محاسبه کرده و به ازای هر کدام، k نزدیکترین داده را به عنوان نزدیکترین نمونه ها بر می گردانیم.

در نهایت اگر مساله به صورت رگرسیون یا regression باشد، بین y های نمونه ها میانگین گرفته و جواب نهایی ای مقدار نمونه ی مورد پرسش بدست می آید و اگر مساله دسته بندی یا classification باشد، مد k نمونه به عنوان دسته ی نمونه ی مورد پرسش در نظر گرفته می شود.

Prototype weighting

تکنیک های بسیاری برای بهبود KNN معمولی ارائه شده اند. از جمله ی این تکنیک ها می توان به وزن دهی مناسب به کلاس ها، protype ها، اشاره کرد. در این روش در واقع به هرکدام از prototype ها در ابتدا وزن اولیه ای داده می شود و در هر مرحله به دنبال آن هستیم تا با تغییر این وزن ها، نمونه های موافق بیشتری در لیست k نزدیکترین نمونه ها قرار بگیرند و در نتیجه ی آن نیز نمونه های مخالف از این لیست خارج شوند.

Feature weighting

در این پروژه می خواهیم با وزن دهی مناسب به ویژگی ها و پیدا کردن بهترین وزن برای هر ویژگی و بهترین k معیار k-accuracy را ماکزیمم کنیم. معیار k-accuracy به این معناست که در k نزدیکترین نمونه ها، حداقل یک نمونه ی موافق که label آن، به عبارت دیگر دسته ای که آن نمونه در آن قرار گرفته است، با label نمونه ی مورد پرسش، query، یکسان باشد. برای پیاده سازی این روش به صورت زیر عمل می کنیم.

روش کار feature weighting

یک تعداد مجموعه ی داده، دیتاست T، که دارای N داده است و هر کدام از داده ها، x، دارای D ویژگی، یا feature، هستند را در نظر میگیریم.

$$\forall x \in T \rightarrow x\ has\ D\ features$$

در واقع به دنبال آن هستیم تا وزن های ویژگی ها را تا زمانی که حداقل یک نمونه ی موافق در k nearest neighbor های نمونه ی مورد پرسش وجود داشته باشد تغییر بدهیم.

ابتدا هر کدام از داده های مجموعه داده را به عنوان یک نمونه ی مورد پرسش جدید در نظر می گیریم و فاصله ی آن را تا تمام داده های دیگر محاسبه می کنیم. برای این منظور معیار فاصله ی دو داده از مجموعه داده ها را بر اساس وزن های ویژگی ها، feature weights، به صورت زیر تعریف می کنیم.

$$d_E(x_1, x_2) = \sum_{j=0}^{D} (x_{1j} - x_{2j})^2 w_j$$

سپس objective-function ای به صورت زیر تعریف می کنیم و نهایتا هنگام استفاده از gradient ascent method به همراه تکنیک leave-one-out به دنبال آن هستیم که مقدار این objective-function را ماکزیمم کنیم.

$$J_w = \frac{1}{N}\sum_{x \in T} acc_{(T-x)}(x)$$

که مقدار J_w نشان دهنده ی درستی معیار k-accuracy است و تابع acc(x) را نیز به صورت زیر تعریف می کنیم.

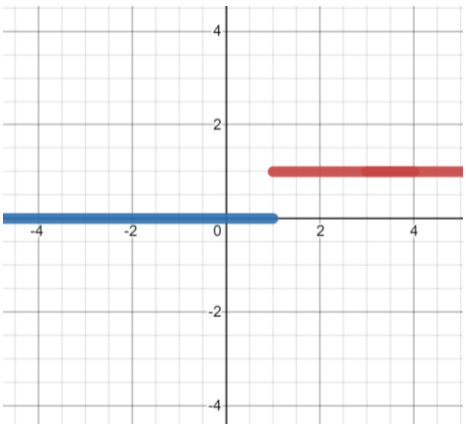
$$acc(x) = step(\frac{R_w(x)}{d_E(x, x^=)})$$

که در آن $R_w(x)$ نشان دهنده ی شعاع نمونه ی x تا k اُمین نزدیکترین نمونه و یک عدد ثابت می باشد و ن نیز در ابتدا تعیین می شود. شعاع نمونه ی x در هر مرحله از اول به کمک فرمول زیر محاسبه ی شود.

$$R_w(x) = d_E(x, N_k(x))$$

$x^=$ نیز نزدیکترین نمونه ی موافق با x می باشد که label آن ها یکسان است و فاصله ی $d_E(x, x^=)$ نی به کمک فرمول x-y محاسبه می شود. تابع step نیز به صورت زیر تعریف می شود.

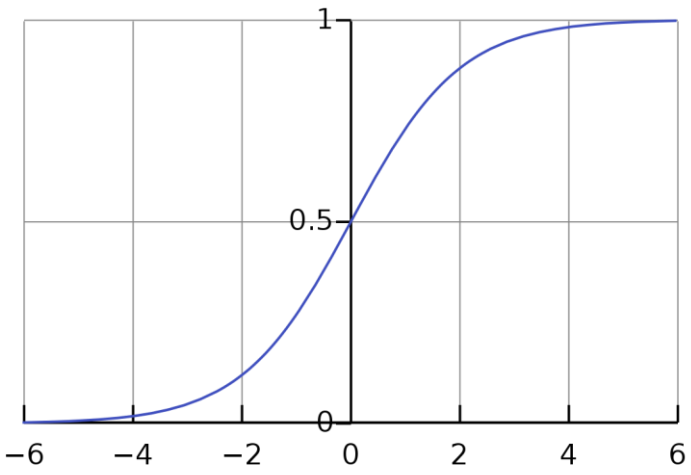
$$step(z) = \begin{cases} 1 & if\ z \geq 1 \\ 0 & if\ z < 1 \end{cases}$$



تابع step

از آن جا که برای ماکزیمم کردن تابع accuracy از تکنیک gradient ascent استفاده می کنیم، تابع objective function-acc(x) باید نسب به w_i مشتق پذیر باشد و به همین دلیل تابع step را با تابع sigmoid که تعریف آن در ادامه آمده است جایگزین می کنیم.

$$\varphi_\beta(z) = \frac{1}{1 + e^{\beta(1-z)}}$$



تابع sigmoid(x)

در نهایت دو فرمول زیر برای محاسبه ی accuracy شبکه داریم:

$$acc(x) = \varphi_\beta(r_w(x))$$

$$r_w(x) = \frac{R_w(x)}{d_E(x, x^=)}$$

همان طور که گفته شد برای تغییر وزن های ویژگی ها در هر مرحله، از تکنیک gradient ascent استفاده می کنیم. به همین دلیل باید از تابع J_w نسبت به هر کدام از w_i ها مشتق بگیریم. در ادامه مشتق J_w نسبت به w_i را محاسبه کرده ایم.

$$\varphi_\beta{}'(r_x) = \frac{\beta\, e^{\beta(1-r_x)}}{(1 + e^{\beta(1-r_x)})^2} = \beta\varphi_\beta(r_x)(1 - \varphi_\beta(r_x))$$

$$\frac{\partial J_w}{\partial w_i} = \sum_{x \in T} \frac{\partial \varphi_\beta}{\partial r_w(x)} \times \frac{\partial r_w(x)}{\partial w_i} = \sum_{x \in T} \varphi_\beta{}'(r_x) \times \frac{-R_w(x) \times \sum_{j=0}^{D}(x - x^=)}{d_E(x, x^=)^2}$$

$$= \sum_{x \in T} - \varphi_\beta{}'(r_x)\, r_w(x)\, \frac{\sum_{j=0}^{D}(x - x^=)}{d_E(x, x^=)}$$

برای تغییر ضرایب و وزن های ویژگی ها، w_i ها، به صورت زیر عمل می کنیم. α ضریب یادگیری شبکه یا learning rate می باشد که معمولا مقدار 0.1 یا 0.01 دارد.

$$w_i^{new} = w_i^{old} + \alpha \times \frac{\partial J_w}{\partial w_i}$$

منابع

[۱] A Generalized Weighted Distance k-Nearest Neighbor for Multi-label Problems, A Generalized Weighted Distance k-Nearest Neighbor for Multi-label Problems | Request PDF (researchgate.net)

[۲] Learning weighted metrics to minimize nearest-neighbor classification error, Learning weighted metrics to minimize nearest-neighbor classification error (researchgate.net)

[۳] Feature weighting to tackle label dependencies in multi-label stacking nearest neighbor, Feature weighting to tackle label dependencies in multi-label stacking nearest neighbor | SpringerLink