Trustworthy Aritificial Intelligence: K-Nearest Neighbors, Linear Regression, Logistic Regression

Presented by

Dr. Narinder Singh Punn,

Assistant Professor,

Dept. of CSE,

ABV-IIITM Gwalior

विश्वजीवनामृतं ज्ञानम्

K-nearest neighbors

- Given *n* data points, a distance function *d* and a new point *x* to classify, select the class of *x* based on the majority vote in the *K* closest points.
  - This requires the definition of a distance function or similarity measure between samples.
  - We also need to determine *K* beforehand.
- From a probabilistic view, KNN tries to approximate the Bayes decision rule on a subset of data.
  - We compute *P(x | y)*, *P(y)* and *P(x)* for some small region around our sample, and the size of that region will be dependent on the distribution of the test sample. How?

K-nearest neighbors

Let z be the new point we want to classify. Let V be the volume of the m dimensional ball $\mathcal{R}$ around z containing the K nearest neighbors for z (where m is the number of features). Also assume that the distribution in R is uniform.

Consider the probability P that a data point chosen at random is in $\mathcal{R}$. On one hand, because there are K points in $\mathcal{R}$ out of a total of N points, $P = \frac{K}{N}$. On the other hand, let $P(x) = q$ be the density at a point $x \in \mathcal{R}$ (q is constant because $\mathcal{R}$ has uniform distribution). Then $P = \int_{x \in \mathcal{R}} P(x)dx = qV$. Hence we see that the marginal probability of z is

$$P(z) = q = \frac{P}{V} = \frac{K}{NV}.$$

Similarly, the conditional probability of z given a class i is

$$P(z \mid y = i) = \frac{K_i}{N_i V}.$$

Finally, we compute the prior of class i:

$$P(y = i) = \frac{N_i}{N}.$$

Using Bayes formula:

$$P(y = i \mid z) = \frac{P(z \mid y = i)P(y = i)}{P(z)} = \frac{K_i}{K}.$$

Using the Bayes decision rule we will choose the class with the highest probability, which corresponds to the class with the highest K_i - the number of samples in K.

Linear Regression

Given an input x we would like to compute an output y as

$$y = wx + \epsilon,$$

where w is a parameter and ϵ represents measurement of noise.

- Our goal is to estimate w from training data of (x_i, y_i) pairs. One way is to find the least square error (LSE)

$$\hat{w}_{LR} = \arg\min_{w} \sum_{i} (y_i - wx_i)^2 \tag{3.1}$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- Which minimizes squared distance between measurements and predicted lines.

LSE Interpretation

$$y_i = wx_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Density: $f(\epsilon_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\epsilon_i^2}{2\sigma^2}\right)$

Since $\epsilon_i = y_i - wx_i$, likelihood for y_i:

$$f(y_i|w, x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - wx_i)^2}{2\sigma^2}\right).$$

For n observations: $L(w) = \prod_{i=1}^{n} f(y_i|w, x_i)$.

$$L(w) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - wx_i)^2}{2\sigma^2}\right).$$

LSE Interpretation

Log-likelihood:

$\log L(w) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - wx_i)^2.$

Maximizing $\log L(w)$ minimizes $\sum_{i=1}^{n}(y_i - wx_i)^2.$

Linear Regression

- LSE has a probabilistic interpretation.
- The solution is

$$\boxed{\hat{w} = \frac{\sum_i x_i y_i}{\sum_i x_i^2}}$$

We take the derivative w.r.t w and set to 0:

$$0 = \frac{\partial}{\partial w} \sum_i (y_i - wx_i)^2 = -2 \sum_i x_i (y_i - wx_i),$$

which yields

$$\sum_i x_i y_i = \sum_i w x_i^2 \Rightarrow \boxed{w = \frac{\sum_i x_i y_i}{\sum_i x_i^2}}$$

Linear Regression

- If the line does not pass through the origin, simply change the model to

$$\boxed{y = w_0 + w_1 x + \epsilon}$$

and following the same process gives

$$\boxed{w_0 = \frac{\sum_i y_i - w_1 x_i}{n}, \quad w_1 = \frac{\sum_i x_i (y_i - w_0)}{\sum_i x_i^2}}$$

Multivariate and general linear regression

- If we have several inputs, this becomes a multivariate regression problem:

$$\boxed{y = w_0 + w_1 x_1 + \ldots + w_k x_k + \epsilon}$$

- However, not all functions can be approximated using the input values directly. In some cases we would like to use polynomial or other terms based on the input data.
- *As long as the coefficients are linear, the equation is still a linear regression problem*. For instance,

$$\boxed{y = w_0 x_1 + w_1 x_1^2 + \ldots + w_k x_k^2 + \epsilon}$$

Multivariate and general linear regression

- Typical non-linear basis functions include:
  - Polynomial $\phi_j(x) = x^j$
  - Gaussian $\phi_j(x) = \frac{(x-\mu_j)^2}{2\sigma_j^2}$
  - Sigmoid $\phi_j(x) = \frac{1}{1+\exp(-s_j x)}$
- Using this new notation, we formulate the general linear regression problem:

$$\boxed{y = \sum_j w_j \phi_j(x)}$$

Multivariate and general linear regression

- Now assume the general case where we where have *n* data points and each data point has *k* features. Again using LSE to find the optimal solution

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})$$

feature j of $x^{(i)}$ is denoted $x_j^{(i)}$

$$\Phi = \begin{pmatrix} \phi_0(x^{(1)}) & \phi_1(x^{(1)}) & \ldots & \phi_k(x^{(1)}) \\ \phi_0(x^{(2)}) & \phi_1(x^{(2)}) & \ldots & \phi_k(x^{(2)}) \\ \vdots & \vdots & \ldots & \vdots \\ \phi_0(x^{(n)}) & \phi_1(x^{(n)}) & \ldots & \phi_k(x^{(n)}) \end{pmatrix} = \begin{pmatrix} \text{—} \ \phi(x^{(1)})^T \ \text{—} \\ \text{—} \ \phi(x^{(2)})^T \ \text{—} \\ \ldots \\ \text{—} \ \phi(x^{(n)})^T \ \text{—} \end{pmatrix}, \tag{3.4}$$

$$\boxed{w = (\Phi^T\Phi)^{-1}\Phi^T y}$$

Multivariate and general linear regression

Our goal is to minimize the following loss function:

$$J(w) = \sum_i (y^{(i)} - \sum_j w_j \phi_j(x^{(i)}))^2 = \sum_i (y^{(i)} - w^T \phi(x^{(i)}))^2,$$

where w and $\phi(x^{(i)})$ are vectors of dimension $k+1$ and $y^{(i)}$ is a scalar.
Setting the derivative w.r.t w to 0:

$$0 = \frac{\partial}{\partial w} \sum_i (y^{(i)} - w^T \phi(x^{(i)}))^2 = 2 \sum_i (y^{(i)} - w^T \phi(x^{(i)})) \phi(x^{(i)})^T,$$

which yields

$$\sum_i y^{(i)} \phi(x^{(i)})^T = w^T \sum_i \phi(x^{(i)}) \phi(x^{(i)})^T.$$

Hence, defining Φ as in (3.4) would give us

$$(\Phi^T \Phi) w = \Phi^T y \Rightarrow \boxed{w = (\Phi^T \Phi)^{-1} \Phi^T y}$$

General Linear Regression

Input: Given n input data $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ where $x^{(i)}$ is $1 \times m$ and $y^{(i)}$ is scalar, as well as m basis functions $\{\phi_j\}_{j=1}^{m}$, we find

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{n} (y^{(i)} - w^T \phi(x^{(i)}))^2$$

by the following procedure:

1. Compute Φ as in (3.4).
2. Output $\hat{w} = (\Phi^T \Phi)^{-1} \Phi^T y$.

Ridge Regression

- What if $\Phi^T\Phi$ is not invertible?
- Recall that full rank matrices are invertible, and that

$$\begin{aligned}\text{rank}(\Phi^T\Phi) &= \text{the number of non-zero eigenvalues of } \Phi^T\Phi \\ &\leq \min(n,k) \text{ since } \Phi \text{ is } n \times k\end{aligned}$$

- In other words, $\Phi^T\Phi$s not invertible if *n < k*, i.e., there are more features than data point. More specifically, we have n equations and *k > n* unknowns - this is an undetermined system of linear equations with many feasible solutions.
- One way, for example, is Ridge Regression - using L2 norm as penalty to bias the solution to "small" values of *w* (so that small changes in input don't translate to large changes in output)

Ridge Regression

$$
\begin{aligned}
\hat{w}_{Ridge} &= \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i w)^2 + \lambda \|w\|_2^2 \\
&= \arg\min_{w} (\Phi w - y)^T (\Phi w - y) + \lambda \|w\|_2^2, \quad \lambda \geq 0 \\
&= (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y.
\end{aligned}
$$

- This addition ensures that $\Phi^T\Phi + \lambda I$ becomes full rank, even if $\Phi^T\Phi$ was not. A full-rank square matrix is always invertible.

Lasso Regression

- We could also use Lasso Regression (L1 penalty)

$$\boxed{\hat{w}_{Lasso} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i w)^2 + \lambda \|w\|_1}$$

- There is no closed form solution. Multiple subgradients value possible.

$$\partial||w||_1/\partial w_j = \begin{cases} 1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j < 0 \\ [-1, 1] & \text{if } w_j = 0 \end{cases}$$

- Iterative method is needed.

General Linear Regression

- In general, we can phrase the problem as finding

$$\boxed{\hat{w} = \arg\min_{w}(\Phi w - y)^T(\Phi w - y) + \lambda \mathrm{pen}(w)}$$

Logistic Regression

- We know that regression is for predicting real-valued output Y, while classification is for predicting discrete-valued Y.
- But is there a way to connect regression to classification?
- Can we predict the probability of a class label?
- The answer is generally yes, but we have to keep in mind the constraint that the probability value should lie in [0, 1].
- In essence, logistic regression means applying the logistic function to a linear function of the data. However, note that it is still a linear classier.

$$\boxed{\sigma(z) = \frac{1}{1+\exp(-z)}}$$

Logistic Regression

Assume the following functional form for $P(Y \mid X)$:

$$P(Y = 1 \mid X) = \frac{1}{1 + \exp(-(w_0 + \sum_i w_i X_i))}, \tag{4.1}$$

$$P(Y = 0 \mid X) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}. \tag{4.2}$$

Logistic Regression as a Linear Classifier

Note that $P(Y = 1 \mid X)$ can be rewritten as

$$P(Y = 1 \mid X) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}.$$

We would assign label 1 if $P(Y = 1 \mid X) > P(Y = 0 \mid X)$, which is equivalent to

$$\exp(w_0 + \sum_i w_i X_i) > 1 \Leftrightarrow w_0 + \sum_i w_i X_i > 0.$$

Similarly, we would assign label 0 if $P(Y = 1 \mid X) < P(Y = 0 \mid X)$, which is equivalent to

$$\exp(w_0 + \sum_i w_i X_i) < 1 \Leftrightarrow w_0 + \sum_i w_i X_i < 0.$$

In other words, the decision boundary is the line $w_0 + \sum_i w_i X_i$, which is linear.

Training Logistic Regression

- Given training data $\{(x_i, y_i)\}_{i=1}^n$ where the input has *d* features, we want to learn the parameters $w_0, w_1, \ldots, w_d$
- We can do so by Maximum Conditional Likelihood Estimation (MCLE):

$$\hat{w}_{MCLE} = \arg\max_w \prod_{i=1}^n P(y^{(i)} \mid x^{(i)}, w). \tag{4.3}$$

- Discriminative philosophy: *don't waste effort learning P(X), focus on P(Y | X) - that's all that matters for classification!*

Training Logistic Regression

- Using (4.1) and (4.2), we can then compute the log-likelihood:

$$
\begin{aligned}
l(w) &= \ln\left(\prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}, w)\right) \\
&= \sum_{i=1}^{n}\left[y^{(i)}(w_0 + \sum_{j=1}^{d} w_i x_j^{(i)}) - \ln(1 + \exp(w_0 + \sum_{j=1}^{d} w_i x_j^{(i)}))\right].
\end{aligned} \tag{4.4}
$$

Assume the following functional form for $P(Y \mid X)$:

$$
P(Y = 1 \mid X) = \frac{1}{1 + \exp(-(w_0 + \sum_i w_i X_i))}, \tag{4.1}
$$

$$
P(Y = 0 \mid X) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}. \tag{4.2}
$$

Training Logistic Regression

- There is no closed-form solution to maximize *l(w)*

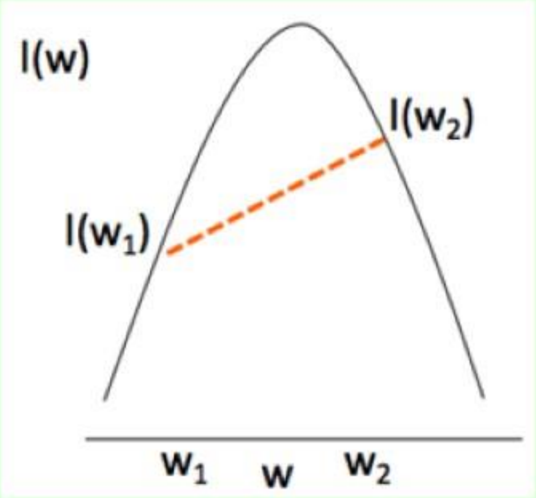
$$\frac{\partial l(w)}{\partial w_k} = \sum_{i=1}^{n} \left[y^{(i)} x_k^{(i)} - \frac{\exp\left(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)}\right)}{1 + \exp\left(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)}\right)} x_k^{(i)} \right]$$

$$\frac{\partial l(w)}{\partial w_k} = \sum_{i=1}^{n} \left[y^{(i)} x_k^{(i)} - \text{sigmoid}\left(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)} \right) x_k^{(i)} \right]$$

Concave Function

- However, it is a concave function.

A function $l(w)$ is called *concave* if the line joining two points $l(w_1), l(w_2)$ on the function does not lie above the function on the interval $[w_1, w_2]$.



Equivalently, a function $l(w)$ is *concave* on $[w_1, w_2]$ if

$$l(tx_1 + (1-t)x_2) \geq tl(x_1) + (1-t)l(x_2)$$

for all $x_1, x_2 \in [w_1, w_2]$ and $t \in [0, 1]$. If the sign is reversed, l is a *convex* function.

Concave and Convex Function

- Concave: A function is concave if the line segment between any two points on the graph of the function lies on or below the graph.
- Convex: A function is convex if the line segment between any two points on the graph of the function lies on or above the graph.

- So what about straight line?

  Its both concave and convex

Proving l(w) is Concave

We first note the following lemmas:

1. If f is convex then $-f$ is concave and vice versa.
2. A linear combination of n convex (concave) functions $f_1, f_2, \ldots, f_n$ with nonnegative coefficients is convex (concave).
3. Another property of twice differentiable convex function is that the second derivative is nonnegative. Using this property, we can see that $f(x) = \log(1 + \exp x)$ is convex.
4. If f and g are both convex, twice differentiable and g is non-decreasing, then $g \circ f$ is convex.

Proving l(w) is Concave

For convenience we denote $x_0^{(i)} = 1$, so that $w_0 + \sum_{i=j}^{d} w_i x_j^{(i)} = w^T x^{(i)}$.
Now we rewrite $l(w)$ as follows:

$$
\begin{aligned}
l(w) &= \sum_{i=1}^{n} y^{(i)} w^T x^{(i)} - \log(1 + \exp(w^T x^{(i)})) \\
&= \sum_{i=1}^{n} y^{(i)} w^T x^{(i)} - \sum_{i=1}^{n} \log(1 + \exp(w^T x^{(i)})) \\
&= \sum_{i=1}^{n} y^{(i)} f_i(w) - \sum_{i=1}^{n} g(f_i(w)),
\end{aligned}
$$

where $f_i(w) = w^T x^{(i)}$ and $g(z) = \log(1 + \exp z)$.
$f_i(w)$ is of the form $Ax + b$ where $A = x^{(i)}$ and $b = 0$, which means it's affine (i.e., both concave and convex). We also know that $g(z)$ is convex, and it's easy to see g is non-decreasing. This means $g(f_i(w))$ is convex, or equivalently, $-g(f_i(w))$ is concave.
To sum up, we can express $l(w)$ as

$$
l(w) = \underbrace{\sum_{i=1}^{n} y^{(i)} f_i(w)}_{\text{concave}} + \underbrace{\sum_{i=1}^{n} -g(f_i(w))}_{\text{concave}},
$$

hence $l(w)$ is concave.

Gradient Ascent Algorithm

- As such, it can be optimized by the Gradient Ascent Algorithm.

Initialize: Pick w at random.

Gradient:

$$\nabla_w E(w) = \left(\frac{\partial E(w)}{\partial w_0}, \frac{\partial E(w)}{\partial w_1}, \ldots, \frac{\partial E(w)}{\partial w_d} \right).$$

Update:

$$\Delta w = \eta \nabla_w E(w)$$

$$w_t^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial E(w)}{\partial w_i},$$

where $\eta > 0$ is the learning rate.

Training Logistic Regression

Initialize: Pick w at random and a learning rate η.
Update:

- Set an $\epsilon > 0$ and denote

$$\hat{P}(y^{(i)} = 1 \mid x^{(i)}, w^{(t)}) = \frac{\exp(w_0^{(t)} + \sum_{j=1}^{d} w_j^{(t)} x_j^{(i)})}{1 + \exp(w_0^{(t)} + \sum_{j=1}^{d} w_j^{(t)} x_j^{(i)})}.$$

- Iterate until $|w_0^{(t+1)} - w_0^{(t)}| < \epsilon$:

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_{i=1}^{n} \left[y^{(i)} - \hat{P}(y^{(i)} = 1 \mid x^{(i)}, w^{(t)}) \right].$$

- For $k = 1, \ldots, d$, iterate until $|w_k^{(t+1)} - w_k^{(t)}| < \epsilon$:

$$w_k^{(t+1)} \leftarrow w_k^{(t)} + \eta \sum_{i=1}^{n} x_j^{(i)} \left[y^{(i)} - \hat{P}(y^{(i)} = 1 \mid x^{(i)}, w^{(t)}) \right].$$

THANK YOU