Machine Learning

Probabilistic KNN.

Mark Girolami

girolami@dcs.gla.ac.uk

Department of Computing Science
University of Glasgow
Probabilistic KNN

• KNN is a remarkably simple algorithm with proven error-rates
Probabilistic KNN

- KNN is a remarkably simple algorithm with proven error-rates
- One drawback is that it is not built on any probabilistic framework
Probabilistic KNN

• KNN is a remarkably simple algorithm with proven error-rates
• One drawback is that it is not built on any probabilistic framework
• No posterior probabilities of class membership
Probabilistic KNN

- KNN is a remarkably simple algorithm with proven error-rates
- One drawback is that it is not built on any probabilistic framework
- No posterior probabilities of class membership
- No way to infer number of neighbours or metric parameters probabilistically
Probabilistic KNN

- KNN is a remarkably simple algorithm with proven error-rates.
- One drawback is that it is not built on any probabilistic framework.
- No posterior probabilities of class membership.
- No way to infer number of neighbours or metric parameters probabilistically.
- Let us try and get around this 'problem'.
Probabilistic KNN

- The first thing which is needed is a likelihood
Probabilistic KNN

• The first thing which is needed is a likelihood
• Consider a finite data sample \( \{(t_1, x_1), \ldots, (t_N, x_N)\} \) where each \( t_n \in \{1, \ldots, C\} \) denotes the class label and \( D \)-dimensional feature vector \( x_n \in \mathbb{R}^D \). The feature space \( \mathbb{R}^D \) has an associated metric with parameters \( \theta \) denoted as \( \mathcal{M}_\theta \).
Probabilistic KNN

• The first thing which is needed is a likelihood

• Consider a finite data sample \( \{(t_1, x_1), \cdots, (t_N, x_N)\} \) where each \( t_n \in \{1, \cdots, C\} \) denotes the class label and \( D \)-dimensional feature vector \( x_n \in \mathbb{R}^D \). The feature space \( \mathbb{R}^D \) has an associated metric with parameters \( \theta \) denoted as \( M_\theta \).

• A likelihood can be formed as

\[
 p(t|X, \beta, k, \theta, M) \approx \prod_{n=1}^{C} \frac{\exp \left\{ \frac{\beta}{k} \sum_{j \sim n|k} \delta_{t_n t_j} \right\}}{\sum_{c=1}^{C} \exp \left\{ \frac{\beta}{k} \sum_{j \sim n|k} \delta_{c t_n} \right\}}
\]
Probabilistic KNN

- The number of nearest neighbours is $k$ and $\beta$ defines a scaling variable. The expression

$$
\sum_{j \sim n|k} \delta_{t_n t_j}
$$

denotes the number of the nearest $k$ neighbours of $x_n$, as measured under the metric $M_\theta$ within $N - 1$ samples from $X$ remaining when $x_n$ is removed which we denote as $X_{-n}$, and have the class label value of $t_n$, whilst each of the terms in the summation of the denominator provides a count of the number of the $k$ neighbours of $x_n$ which have class label equaling $c$. 
Probabilistic KNN

- Likelihood formed by product of terms

\[ p(t_n|x_n, X_{-n}, t_{-n}, \beta, k, \theta, M) \]
Probabilistic KNN

- Likelihood formed by product of terms

\[ p(t_n|x_n, X_{-n}, t_{-n}, \beta, k, \theta, M) \]

- This is a Leave-One-Out (LOO) predictive likelihood, where \( t_{-n} \) denotes the vector \( t \) with the \( n \)'th element removed
Probabilistic KNN

• Likelihood formed by product of terms

\[ p(t_n | x_n, X_{-n}, t_{-n}, \beta, k, \theta, \mathcal{M}) \]

• This is a Leave-One-Out (LOO) predictive likelihood, where \( t_{-n} \) denotes the vector \( t \) with the \( n \)'th element removed

• Approximate joint likelihood provides an overall measure of the LOO predictive likelihood
Probabilistic KNN

• Likelihood formed by product of terms

\[ p(t_n|x_n, X_{-n}, t_{-n}, \beta, k, \theta, M) \]

• This is a Leave-One-Out (LOO) predictive likelihood, where \( t_{-n} \) denotes the vector \( t \) with the \( n \)'th element removed

• Approximate joint likelihood provides an overall measure of the LOO predictive likelihood

• Should exhibit some resilience to overfitting due to the LOO nature of the approximate likelihood
Posterior inference will follow by obtaining the parameter posterior distribution $p(\beta, k, \theta | t, X, M)$.
Probabilistic KNN

• Posterior inference will follow by obtaining the parameter posterior distribution \( p(\beta, k, \theta | t, X, M) \)

• Predictions of the target class label \( t_* \) of a new datum \( x_* \) are made by posterior averaging such that \( p(t_* | x_*, t, X, M) \) equals

\[
\sum_k \int p(t_* | x_*, t, X, \beta, k, \theta, M) p(\beta, k, \theta | t, X, M) d\beta d\theta
\]
• Posterior inference will follow by obtaining the parameter posterior distribution $p(\beta, k, \theta|t, X, M)$

• Predictions of the target class label $t_*$ of a new datum $x_*$ are made by posterior averaging such that $p(t_*|x_*, t, X, M)$ equals

$$\sum_k \int p(t_*|x_*, t, X, \beta, k, \theta, M)p(\beta, k, \theta|t, X, M)d\beta d\theta$$

• Posterior takes an intractable form so MCMC procedure is proposed so that the following Monte-Carlo estimate is employed

$$\hat{p}(t_*|x_*, t, X, M) = \frac{1}{N_s} \sum_{s=1}^{N_s} p(t_*|x_*, t, X, \beta^{(s)}, k^{(s)}, \theta^{(s)}, M)$$
Probabilistic KNN

- Posterior sampling algorithm simple Metropolis algorithm
Probabilistic KNN

• Posterior sampling algorithm simple Metropolis algorithm
• Assume priors on \( k \) and \( \beta \) are uniform over all possible values (integer & real)
Probabilistic KNN

- Posterior sampling algorithm simple Metropolis algorithm
- Assume priors on $k$ and $\beta$ are uniform over all possible values (integer & real)
- Proposal distribution for $\beta_{new}$ is Gaussian i.e. $\mathcal{N}(\beta^{(i)}, h)$
Probabilistic KNN

- Posterior sampling algorithm simple Metropolis algorithm
- Assume priors on $k$ and $\beta$ are uniform over all possible values (integer & real)
- Proposal distribution for $\beta_{new}$ is Gaussian i.e. $\mathcal{N}(\beta^{(i)}, h)$
- Proposal distribution for $k$ is uniform between Min & Max values

$$index \sim U(0, k_{step} + 1)$$
$$k_{new} = k_{old} + k_{inc}(index);$$
Probabilistic KNN

- Need to accept this new move using Metropolis ratio
• Need to accept this new move using Metropolis ratio

\[
\min \left\{ 1, \frac{p(t|X, \beta_{\text{new}}, k_{\text{new}}, \theta_{\text{new}}, \mathcal{M})}{p(t|X, \beta, k, \theta, \mathcal{M})} \right\}
\]
Probabilistic KNN

• Need to accept this new move using Metropolis ratio

\[
\min \left\{ 1, \frac{p(t|X, \beta_{\text{new}}, k_{\text{new}}, \theta_{\text{new}}, M)}{p(t|X, \beta, k, \theta, M)} \right\}
\]

• Builds up a Markov Chain whose stationary distribution is \( p(\beta, k, \theta|t, X, M) \)
Probabilistic KNN

- Need to accept this new move using Metropolis ratio

\[
\min \left\{ 1, \frac{p(t|X, \beta_{\text{new}}, k_{\text{new}}, \theta_{\text{new}}, M)}{p(t|X, \beta, k, \theta, M)} \right\}
\]

- Builds up a Markov Chain whose stationary distribution is \( p(\beta, k, \theta|t, X, M) \)

- Very simple algorithm to implement - Matlab and C implementations available
Probabilistic KNN

- Trace of Metropolis Sampler for $\beta$ & $k$
Figure 1: The top graph shows a histogram of the marginal posterior for $K$ on the synthetic Ripley dataset and the bottom shows the 10CV error against the value of $K$. 
Figure 2: The percentage test error obtained with training sets of varying size from 25 to 250 data points. For each sub-sample size, 50 random subsets were sampled and each of these used to obtain a KNN and PKNN classifier which were then used to make predictions on the 1000 independent test points. The mean percentage performance and associated standard error obtained for each training set are shown in the above figure for each classifier.
## Probabilistic KNN

<table>
<thead>
<tr>
<th>Data</th>
<th>KNN</th>
<th>PKNN</th>
<th>P-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>29.91 ± 9.22</td>
<td>26.67 ± 8.81</td>
<td>0.517</td>
</tr>
<tr>
<td>Iris</td>
<td>5.33 ± 5.25</td>
<td>4.00 ± 5.62</td>
<td>0.537</td>
</tr>
<tr>
<td>Crabs</td>
<td>15.00 ± 8.82</td>
<td>19.50 ± 6.85</td>
<td>0.240</td>
</tr>
<tr>
<td>Pima</td>
<td>27.00 ± 8.88</td>
<td>24.00 ± 14.68</td>
<td>0.645</td>
</tr>
<tr>
<td>Soybean</td>
<td>14.50 ± 16.74</td>
<td>4.50 ± 9.56</td>
<td>0.155</td>
</tr>
<tr>
<td>Wine</td>
<td>3.922 ± 3.77</td>
<td>3.37 ± 2.89</td>
<td>0.805</td>
</tr>
<tr>
<td>Balance</td>
<td>11.52 ± 2.99</td>
<td>10.23 ± 3.02</td>
<td>0.324</td>
</tr>
<tr>
<td>Heart</td>
<td>15.18 ± 5.91</td>
<td>15.18 ± 4.43</td>
<td>1.000</td>
</tr>
<tr>
<td>Liver</td>
<td>33.60 ± 6.98</td>
<td>36.26 ± 12.93</td>
<td>0.705</td>
</tr>
<tr>
<td>Diabetes</td>
<td>25.91 ± 7.15</td>
<td>25.25 ± 8.11</td>
<td>0.970</td>
</tr>
<tr>
<td>Vehicle</td>
<td>36.28 ± 5.16</td>
<td>37.22 ± 4.53</td>
<td>0.732</td>
</tr>
</tbody>
</table>
## Probabilistic KNN

<table>
<thead>
<tr>
<th>Data</th>
<th>KNN</th>
<th>PKNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>39.55</td>
<td>243.52</td>
</tr>
<tr>
<td>Iris</td>
<td>7.58</td>
<td>91.8</td>
</tr>
<tr>
<td>Crabs</td>
<td>21.99</td>
<td>156.30</td>
</tr>
<tr>
<td>Pima</td>
<td>24.10</td>
<td>103.60</td>
</tr>
<tr>
<td>Soybean</td>
<td>1.16</td>
<td>38.38</td>
</tr>
<tr>
<td>Wine</td>
<td>27.9</td>
<td>144.90</td>
</tr>
<tr>
<td>Balance</td>
<td>609.86</td>
<td>555.72</td>
</tr>
<tr>
<td>Heart</td>
<td>96.11</td>
<td>145.22</td>
</tr>
<tr>
<td>Liver</td>
<td>116.71</td>
<td>189.73</td>
</tr>
<tr>
<td>Diabetes</td>
<td>1643.09</td>
<td>567.03</td>
</tr>
<tr>
<td>Vehicle</td>
<td>4226.69</td>
<td>1063.13</td>
</tr>
</tbody>
</table>

Table 1: The running times (seconds) for KNN with no...
Probabilistic KNN

- PKNN is a fully Bayesian method for KNN classification
Probabilistic KNN

- PKNN is a fully Bayesian method for KNN classification
- Requires MCMC therefore slow
Probabilistic KNN

- PKNN is a fully Bayesian method for KNN classification
- Requires MCMC therefore slow
- Possible to learn metric though this is computationally demanding
Probabilistic KNN

- PKNN is a fully Bayesian method for KNN classification
- Requires MCMC therefore slow
- Possible to learn metric though this is computationally demanding
- Predictive probabilities more useful in certain applications - e.g. clinical prediction
Probabilistic KNN

- PKNN is a fully Bayesian method for KNN classification
- Requires MCMC therefore slow
- Possible to learn metric though this is computationally demanding
- Predictive probabilities more useful in certain applications - e.g. clinical prediction
- On 0-1 loss no statistically significant difference with CV & KNN