7 Problems and Solutions (Part 2 of 3)

Multiclass Classification

Multiclass classification uses $C \geq 2$ classes: $y \in \{$_______________$\}$.

- ID3 and other decision trees are easy to change. e.g. For ID3, change from §3's binary

  $$f_{ID3}(S) = \frac{1}{|S|} \sum_{(\mathbf{x},y) \in S} y$$

  to

  $$f_{ID3}(S,c) = P(y_i = c|\mathbf{x}) = \frac{1}{|S|} \sum_{(\mathbf{x},y) \in S \text{ and } \underline{\qquad\qquad}} 1$$

  for each $y \in \{1, \ldots, C\}$.[1]

- Logisitic regression can be extended via the *softmax function* from §6:[2]

  Recall from §3: In logistic regression for two-class y, we used $\hat{P}_{\mathbf{w},b}(y = 1|\mathbf{x}) = f_{\mathbf{w},b}(\mathbf{x}) = \sigma(\mathbf{wx} + b) = \frac{1}{1+e^{-(\mathbf{wx}+b)}}$. Here we used the standard logistic function, $\sigma(t) = \frac{1}{1+e^{-t}}$, which maps $t \in \mathbb{R}$ to $(0, 1)$.

  The softmax function generalizes to $C > 2$ dimensions as follows:

  $\sigma(\mathbf{z}) = [\sigma^{(1)}, \ldots, \sigma^{(C)}]$, where $\sigma^{(j)} = \dfrac{\exp\left(z^{(j)}\right)}{\sum_{k=1}^{C} \exp\left(z^{(k)}\right)}$, which maps $\mathbb{R}^C$ to $(0, 1)^C$. Since $\sum_{i=1}^{C} \sigma^{(i)} =$ _________, we can interpret $[\sigma^{(1)}, \ldots, \sigma^{(C)}]$ as a vector of probabilities, with $\sigma^{(i)} = \hat{P}(y = i)$, for $i \in \{1, \ldots, C\}$.

  - For simple logistic regression, the input to the logistic function $\sigma()$ is the linear function $\mathbf{wx} + b$.
  - For multinomial logistic regression, the input to the softmax function $\sigma()$ is a vector of C linear functions $\mathbf{w}_i\mathbf{x} + b_i$, one for each $i \in \{1, \ldots, C\}$. Burkov omits describing how $\{\mathbf{w}_i, b_i\}$ are trained.

- kNN still returns the ____ among the k nearest neighbors; now "most frequent" is across $C \geq 2$ classes, not just $C = 2$.

---

[1] I think Burkov made a typo by writing "y" where I wrote "1" in the second sum. e.g. For $c = 1$, his probability is the proportion of examples with $y = 1$; but for $c = 2$, his probability is twice the proportion of examples with $y = 2$, which is 2 when all the examples in S have $y = 2$. Another way to write the required proportion is $f_{ID3}(S,c) = P(y_i = c|\mathbf{x}) = \frac{1}{|S|} \sum_{(\mathbf{x},y) \in S} \delta(y, c)$, where $\delta(i,j) = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases}$ is the *Kronecker delta* function.

I wrote $f_{ID3}(S,c)$ where Burkov wrote $f_{ID3}(S)$ because his notation is ambiguous without specifying c.

[2] We did not study §6: Neural Nets and Deep Learning, a STAT 453 topic.

- SVM is naturally binary. It, and most other binary classifiers, can be extended by *one vs. rest*, which solves a C-class problem via C binary classifiers.

  e.g. For three classes, $y \in \{1, 2, 3\}$ (so $C = 3$):

  - __________________________ times, changing labels other than 1 to 0 in the first copy, labels other than 2 to 0 in the second, and labels other than 3 to 0 in the third.
  - Train three binary classifiers for 1 and 0, 2 and 0, and 3 and 0.
  - Classify a new $\mathbf{x}$ by choosing the most-certain (non-zero) prediction, where "certainty" is proportional to the __________________________ from $\mathbf{x}$ to the decision boundary,[3] $d = \dfrac{\mathbf{w}^*\mathbf{x} + b^*}{||\mathbf{w}^*||}$. Regarding the distance:

    As in §1, the hyperplane $\mathbf{wx} + b = 0$ has normal vector $\mathbf{w}$ and points on the normal line through some point $\mathbf{z}$ are given by $\mathbf{z} + \mathbf{w}t$ for $t \in \mathbb{R}$. The intersection is when $\mathbf{w}(\mathbf{z} + \mathbf{w}t) + b = 0 \implies t = -\frac{\mathbf{wz}+b}{||\mathbf{w}||^2}$. The distance from $\mathbf{z}$ to the intersection point is $||\mathbf{z} - (\mathbf{z} + \mathbf{w}t)|| = ||\mathbf{w}t|| = ||\mathbf{w}|| \cdot |t| = \frac{|\mathbf{wz}+b|}{||\mathbf{w}||}$. The signed distance is $\frac{\mathbf{wz}+b}{||\mathbf{w}||}$. (Well, I need to add "*" to all my $\mathbf{w}$'s and finally change my $\mathbf{z}$ to Burkov's $\mathbf{x}$.)

  Most classification algorithms either are convertible to multiclass or ____________________ with which we can use this one-vs.-rest strategy.

  e.g. Draw three sets of 2D points (1s, 2s, and 3s) and classify a new point.

Python

- Here is an encouraging note from the user guide linked below: "All classifiers in scikit-learn do multiclass classification ______________________. You don't need to use the `sklearn.multiclass` module unless you want to experiment with different multiclass strategies."

To learn more:

- User guide: `https://scikit-learn.org/stable/modules/multiclass.html`
- Reference manual:

  `https://scikit-learn.org/stable/modules/classes.html?highlight=multiclass#module-sklearn.multiclass`

[3] See `https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html` and its `decision_function(X)`. Note that `svm.LinearSVC()` is required for one-vs.-rest behavior; `svm.SVC()` uses a "one-vs.one" method we did not discuss.

One-Class Classification

One-class classification identifies one class for which we have training data from everything else.

e.g. An IT department managing its computer network wants to detect anomalous traffic.

- *One-class Gaussian* models the training data with the *multivariate normal distribution* (MND) $N_D(\boldsymbol{\mu}_D, \boldsymbol{\Sigma}_{D\times D})$ whose probability density function is

  $$f_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}}$$

  where $\boldsymbol{\Sigma}^{-1}$ is the *inverse* and $|\boldsymbol{\Sigma}|$ is the *determinant* of covariance matrix $\boldsymbol{\Sigma}$.[4]

  We can interpret this as the probability $\mathbf{x}$ was from the distribution with parameters $\boldsymbol{\mu}$ indicating the center of the distribution and $\boldsymbol{\Sigma}$ determining its shape.

  A new input $\mathbf{x}$ is in the one class if $f_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{x})$ is above an experimentally-decided ______________.

  e.g. Draw a $D = 1$ $N(\mu, \sigma)$ curve over a few data points and indicate an outlier.

  e.g. For $D = 2$ examples, see Burkov's Figure 7.2 on p. 80. It is also p. 6 of

  `https://www.dropbox.com/s/esprbgjm0wc5afz/Chapter7.pdf?dl=0`.

  For a more complex shape, we can use a *mixture of Gaussians* requiring one $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ pair per Gaussian and parameters that allow combinging the Gaussians into one pdf. See §9 for an application of such a mixture to the problem of density estimation soon.

- *One-class k-means*[5] is similar. It computes $d(\mathbf{x})$ as the minimum distance from a new $\mathbf{x}$ to each of k cluster centers. If d is less than a threshold, $\mathbf{x}$ is in the class.

- *One-class kNN* is similar. Burkov omits details.

- *One-class SVM* either:

  - separates training examples from __________________ by a hyperplane, maximizing the distance from hyperplane to the origin; or
  - makes a (hyper-)______________ boundary around the data by minimizing its volume.

  Burkov omits details. e.g.

  `https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html`

---

[4] Recall the $1D$ $N(\mu, \sigma)$, whose pdf is $f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ with mean μ and standard deviation σ.

[5] k-means clustering is coming in §9.

Python

- `from sklearn import mixture`:
  - `clf = mixture.GaussianMixture(n_components=1)` gives a model for estimating $f_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{x})$.
  - `clf.fit(X)` fits the model to array $\mathtt{X}_{N\times D}$.
  - `clf.means_` gives $\boldsymbol{\mu}$ and `clf.covariances_` gives $\boldsymbol{\Sigma}$.
  - `clf.score_samples(X)` gives the log-likelihood of each sample, so `np.exp(clf.score_samples(X))` gives $f_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{x})$ for each sample.
  - Choose a threshold, e.g. `np.quantile(a, q)` with `a`=likelihoods and small `q` $\in (0, 1)$.

To learn more:

- User guide: `https://scikit-learn.org/stable/modules/outlier_detection.html`

  One-class Gaussian: `https://scikit-learn.org/stable/modules/mixture.html`

- Reference manual:

  `https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html`

  `https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html`

  Examples:

  `https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_pdf.html`

  `https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html`

Multi-Label Classification

Multi-label classification is required when several labels apply to a single example $\mathbf{x}$.[6]

e.g. A picture of a road in forested mountains has three labels: "conifer," "mountain," "road."

- Transform each labeled example into several examples, each with one of the several original labels. Now we have a multiclass classification problem that can be solved with the ____________________ strategy. Add a threshold hyperparameter, chosen using validation data, and the label for each class scoring above the threshold is assigned to $\mathbf{x}$.

- Other natural multiclass algorithms (decision tree, logistic regression) give a score for each class, so again each class above the threshold is assigned.

- Where the number of values each label can take is small, we can convert a multi-label problem to a ________________ problem.

[6] I am providing no python code for this section.

e.g. For images with two types of labels, medium $\in$ {photo, painting} and style $\in$ {portrait, landscape, other}, create a new fake class for each combination:

Fake class	Medium	Style
1	photo	portrait
2	photo	landscape
3	______________	other
4	painting	______________
5	painting	landscape
6	painting	other