Multiclass Classification

Multiclass classification uses \( C \geq 2 \) classes: \( y \in \{ \ldots \} \).

- ID3 and other decision trees are easy to change. e.g. For ID3, change from §3’s binary
  \[ f_{ID3}(S) = \frac{1}{|S|} \sum_{(x,y) \in S} y \]
  to
  \[ f_{ID3}(S,c) = P(y_i = c|x) = \frac{1}{|S|} \sum_{(x,y) \in S \text{ and } \ldots} 1 \]
  for each \( y \in \{1, \ldots, C\} \).

- Logistic regression can be extended via the \textit{softmax function} from §6.

  Recall from §3: In logistic regression for two-class \( y \), we used \( \hat{P}_{w,b}(y = 1|x) = f_{w,b}(x) = \sigma(wx + b) = \frac{1}{1 + e^{-(wx+b)}} \).

  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:
  \[ \sigma(z) = [\sigma(1), \ldots, \sigma(C)], \text{ where } \sigma^{(j)} = \frac{\exp(z^{(j)})}{\sum_{k=1}^{C} \exp(z^{(k)})}, \text{ which maps } \mathbb{R}^C \text{ to } (0,1)^C. \]
  Since \( \sum_{i=1}^{C} \sigma^{(i)} = 1 \), 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 \( wx + b \).
  
  - For multinomial logistic regression, the input to the softmax function \( \sigma() \) is a vector of \( C \) linear functions \( w_i x + b_i \), one for each \( i \in \{1, \ldots, C\} \). Burkov omits describing how \( \{w_i, b_i\} \) are trained.

- \( k \text{NN} \) still returns the \underline{most frequent} 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

  \[ f_{ID3}(S,c) = P(y_i = c|x) = \frac{1}{|S|} \sum_{(x,y) \in S} \delta(y,c), \text{ where } \delta(i,j) = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases} \text{ is the Kronecker delta function.} \]

\( ^2 \) 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 $x$ by choosing the most-certain (non-zero) prediction, where “certainty” is proportional to the distance from $x$ to the decision boundary:
  \[
  d = \frac{w^*x + b^*}{||w^*||}.
  \]

Regarding the distance:

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

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

**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:


**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.

---

\(^3\)See [https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html](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 Gaussian** models the training data with the **multivariate normal distribution** (MND) \( N_D(\mu_D, \Sigma_{D \times D}) \) whose probability density function is

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

where \( \Sigma^{-1} \) is the **inverse** and \( |\Sigma| \) is the **determinant** of covariance matrix \( \Sigma \).

We can interpret this as the probability \( x \) was from the distribution with parameters \( \mu \) indicating the center of the distribution and \( \Sigma \) determining its shape.

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

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](https://www.dropbox.com/s/esprbgjm0wc5afz/Chapter7.pdf?dl=0).

For a more complex shape, we can use a **mixture of Gaussians** requiring one \((\mu, \Sigma)\) pair per Gaussian and parameters that allow combining 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** is similar. It computes \( d(x) \) as the minimum distance from a new \( x \) to each of \( k \) cluster centers. If \( d \) is less than a threshold, \( x \) is in the class.

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

- **One-class SVM** either:
  - separates training examples from \( \square \) by a hyperplane, maximizing the distance from hyperplane to the origin; or
  - makes a (hyper-) \( \square \) boundary around the data by minimizing its volume.

  Burkov omits details. E.g.


**Python**

- from sklearn import mixture:

  - clf = mixture.GaussianMixture(n_components=1) gives a model for estimating \( f_{\mu, \Sigma}(x) \).
  - clf.fit(X) fits the model to array \( X_{N \times D} \).
  - clf.means_ gives \( \mu \) and clf.covariances_ gives \( \Sigma \).
  - clf.score_samples(X) gives the log-likelihood of each sample, so np.exp(clf.score_samples(X)) gives \( f_{\mu, \Sigma}(x) \) for each sample.
  - We can choose a threshold, e.g. from np.quantile(a, q), passing likelihoods as \( a \) and some small threshold \( q \in (0, 1) \).

---

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

5 \( k \)-means clustering is coming in §9.
Multi-Label Classification

*Multi-label classification* is required when several labels apply to a single example $x$.\(^6\) 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 $x$.
- 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.

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

<table>
<thead>
<tr>
<th>Fake class</th>
<th>Medium</th>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>photo</td>
<td>portrait</td>
</tr>
<tr>
<td>2</td>
<td>photo</td>
<td>landscape</td>
</tr>
<tr>
<td>3</td>
<td>_________</td>
<td>other</td>
</tr>
<tr>
<td>4</td>
<td>painting</td>
<td>_________</td>
</tr>
<tr>
<td>5</td>
<td>painting</td>
<td>landscape</td>
</tr>
<tr>
<td>6</td>
<td>painting</td>
<td>other</td>
</tr>
</tbody>
</table>

\(^6\)I am providing no python code for this section.