Supervised Learning of Classifiers

Carlo Tomasi

Supervised learning is the problem of computing a function from a *feature* (or *input*) *space* X to an *output space* Y from a *training set* T of feature-output pairs:

$$T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\} \quad \text{with} \quad \mathbf{x}_n \in X \quad \text{and} \quad y_n \in Y \ .$$

The learned function f is constrained to be in a *hypothesis space* $\mathcal{H}$.

In the problems considered here, the feature space X is a subset of $\mathbb{R}^D$, and D is called the *dimensionality* of X. If Y is *categorical*, that is, a finite and unordered set,

$$Y = \{c_1, \ldots, c_K\}$$

then the learning problem is called *classification* and the elements of Y are called the *(class) labels*. If $Y = \mathbb{R}$, the problem is called *regression* and the elements of Y are called the *responses*. This note focuses on classifiers.

To measure classification uncertainty for ambiguous cases, the learner first learns a *posterior probability distribution* $p(y|\mathbf{x})$ of the label given the feature $\mathbf{x}$ and then determines the output y corresponding to $\mathbf{x}$ by some separate *decision criterion*. The Maximum a Posteriori (MAP) estimate

$$y = f(\mathbf{x}) = \arg\max_{y \in Y} p(y|\mathbf{x}) \tag{1}$$

is an intuitive choice that can also be justified formally by a Bayesian argument [4]. This probabilistic formulation helps determine the *confidence* in the value of $f(\mathbf{x})$. For instance, if the posteriors of two outputs have similar values, $p(y|\mathbf{x}) \approx p(y'|\mathbf{x})$, we assign low confidence to the result of definition (1). Instead of specifying the hypothesis space $\mathcal{H}$ directly, one specifies the *probability space* $\mathcal{P}$ that $p(y|\mathbf{x})$ is required to belong to, and $\mathcal{H}$ is then determined once the decision criterion is chosen.

To define classification accuracy, it is assumed that all features and labels, including those in the training set, are drawn independently at random from a *joint probability distribution* $p(\mathbf{x}, y)$—an assumption that is not automatically satisfied in practice. The *generalization error* is then the expected misclassification rate:

$$\mathrm{Err}(f) = \mathbb{E}[\mathcal{L}(y, f(\mathbf{x}))]$$

where the *loss function* $\mathcal{L}(y, y')$ measures the cost incurred for outputting y' when the correct label is y. We use the misclassification loss

$$\mathcal{L}(y, y') = I(y \neq y') = \begin{cases} 1 & \text{if } y \neq y' \\ 0 & \text{if } y = y' \end{cases}$$

so that

$$\mathrm{Err}(f) = \mathbb{E}[I(y \neq y')] = \mathbb{P}(y \neq y') \ .$$

The *training error*

$$\overline{\mathrm{err}}(f,T) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y_n, f(\mathbf{x}_n)) \tag{2}$$

is an empirical estimate of the generalization error, and typically a poor one: One can make the function f fit the training data arbitrarily well by allowing a sufficiently large hypothesis space $\mathcal{H}$ (or probability space $\mathcal{P}$), with the result that f will learn idiosyncrasies of the training set and generalize poorly to other inputs. For instance, a car classifier may learn that cars appear on a gray road against a blue sky, and then anything in such an environment might look like a car to the classifier.

Good generalization requires to construct a suitably restricted hypothesis space $\mathcal{H}$ by constraining the set $\mathcal{P}$ of posterior probability distributions the learner is allowed to choose from.

1 Classifier Complexity

A classifier typically implements a member of a set of classification rules of varying complexity.[1] For instance, a classification tree is a tree of questions (called "*split rules*") that admit binary answers and depend on certain parameters. Answering all the questions about an input $\mathbf{x}$ leads to the leaf that contains[2] $p(y|\mathbf{x})$. A particular tree is one of many possible trees of varying depth, and the deeper the tree the more complex the hyper-surface it implies in feature space. Once the depth is chosen, the complexity of the classification rule is set, and training is then used to determine the split rules at the nodes. If there is a choice among different types of split rules, then the rule type is also a factor in determining the complexity of the classification rule. As another example, AdaBoost [2] is a classification method that uses a collection of weak classifiers (classifiers that are just somewhat better than a random guess at the answer) to build a strong classifier. The number of weak classifiers is a factor in determining the classifier's complexity.

For other classes of algorithms, the complexity is tied more subtly to algorithm parameters. For instance, the complexity of a neural net classifier is determined (quite obviously) by the number of its neurons, but also by the number of iterations used in the optimization algorithm that fits the neuron parameters to the training data: When many iterations are used, fitting often becomes nearly perfect, and this means that the algorithm has found a hyper-surface of sufficient complexity in feature space to separate the inputs as indicated by the labels. The empirical error is now zero (or very small), but is usually an overly optimistic estimate of the generalization error: the neural network has *overfit* the data. So the number of optimization iterations is a factor in the complexity of the resulting classification rule.

In the examples above, the split sets of a classification tree, weak classifier parameters in AdaBoost, and neuron parameters in a neural net are not tied to the complexity of the classification rule, but are rather what determines the specific rule implemented by a classifier of given complexity. So there is, at least intuitively, a distinction between algorithm parameters that determine the complexity of the underlying classification model, and algorithm parameters that do not, and rather help fit a model of fixed complexity to the available training data. Let us call the first type of parameters the *hyper-parameters*[3], and the second type just the *parameters*.

[1] The terms "expressiveness" or "capacity" are also sometimes used instead of "complexity."

[2] More on classification trees in an upcoming note.

[3] Sometimes the hyper-parameters are called "regularization parameters."

2 Training, Validation, and Testing

When training a classifier, one needs to adjust both parameters and hyper-parameters, and there is a tension between the requirements for the two tasks: Parameters are set so as to fit the data as well as possible—the training error decreases—while hyper-parameters are set so as to constrain the size of the hypothesis space as much as possible, for better generalization. Thus, adjusting hyper-parameters tends to *increase* the training error out of hope for a better generalization error. Because of this tension, two different sets of data are typically used to optimize parameters and hyper-parameters: Parameters are optimized on a *training set* and hyper-parameters are optimized on a *validation set*. Finally, the performance of the algorithm is estimated on a *test set*, which has no part whatsoever in determining either parameters or hyper-parameters. Once cannot use the validation set for performance evaluation, because that set was used to determine some aspect of the system (its hyper-parameters), and would therefore yield a biased estimate of the generalization error.

A popular rule of thumb is to split the available data (feature-label pairs) randomly and independently into about 70% for training and the rest divided somehow into validation and testing, but numbers vary. A typical algorithm design method is to propose a multitude of hyper-parameter settings, train the classifier with each setting on the training data, and pick the hyper-parameter values that perform best on the validation data. Parameter overfitting is thereby avoided, because the validation data are not used to optimize the classifier's parameters, and the classifier's performance on the validation set is an estimate of its generalization performance.

Sometimes, no explicit distinction is made between training data and validation data. In that case, hyper-parameters are estimated by *cross-validation*, two particularly popular and effective variants of which are K-fold and leave-one-out cross-validation. In *K-fold cross-validation*, the training data are split into K equal-sized sets at random[4]. One then picks a particular setting of the hyper-parameters, and for every $k = 1, \ldots, K$ trains the classifier on all training data points except those in set k. The classifier is then tested on set k, and its performance (number of correct or incorrect outputs) is recorded. This procedure is repeated for different settings of the hyper-parameters, and the settings that yield the lowest *cross-validation error*—the average classification error over all K folds—are chosen. *Leave-one-out cross-validation* is K-fold cross-validation with $K = N$.

The cross-validation error is not an unbiased estimate of the generalization error, because the data sets used for training and validation are the same. However, since training and validation sets are distinct within each fold, the cross-validation error is a better estimate of the generalization error than the training error.

3 The State of the Art of Image Classification

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition that pits image recognition and object detection algorithms against each other on a standardized image database that contains 1.4 million images divided into 1,000 object classes. Designing the database and the various aspects of the competition is a significant research effort in its own right [5]. The competition includes three tasks:

Image Classification: The database maintainers label each image manually with the presence of *one* instance out of 1000 object categories, even if more objects are present in the image. These annotations are called the *ground truth*. Competing classification algorithms return a *list* of up to five objects for

[4]This K has nothing to do with the number of classes. The letter K is used for cross-validation in the literature.

each test image, and the list is deemed correct if it includes the ground-truth label. Algorithms are trained on a subset of 1.2 million images for which the ground-truth annotations are made public, and about 50,000 images with annotations are made public for validation. The competition organizers withhold a set of 100,000 images to test the algorithms submitted by the contestants, and measure the *error rate* as the fraction of incorrect outputs.[5]

Single-Object Localization: In addition to image classification, algorithms also provide an axis-aligned rectangular bounding box around *one* instance of the recognized object class. The box is correct if the area of the intersection of computed and true box is at least 1/2 of the area of their union. As above, a list of up to five object-box pairs is returned.

Object Detection: Same as single-object localization, but *every* instance of the object class is to be found exactly once. Missing, duplicate, and false detections are penalized.

Winners of the competition for each task are announced and invited to contribute insights at either the International Conference on Computer Vision (ICCV) or the European Conference on Computer Vision (ECCV) in alternate years.

Classification, detection, and localization go hand-in-hand: To classify an image one must first detect and localize an object that is deemed to be "of interest," and a correctly detected and well localized object is of course easier to classify correctly. Conversely, a detector often works by running a classifier at all or many windows in the image. This note only looks at performance figures in image classification, with the *caveat* that the best systems often perform well in more than one category, because of these interdependencies.

The classification task is very difficult for at least the following reasons:

- Images are "natural," that is, they are not contrived for the databased but rather downloaded from photo sharing sites like Flickr. Because of this, the objects of interest are on arbitrary backgrounds, and may appear in very different sizes, viewpoints, and lighting conditions in different images, and may be only partially visible.
- There are 1,000 categories, and distinctions between categories are often very subtle. For instance, Siberian husky and Eskimo dog are different categories, but dogs of the two breeds look very similar.
- At the same time, variations of appearance within one category can be significant (how many lamps can you think of?)
- What part of the image is of interest to the labeler may be based on very high-level semantic considerations that a machine learning algorithm may not have access to. For instance, a picture of a group of people examining a fishing rod was labeled as "reel."

Because of these difficulties, the image classification error was 28.2 percent in 2010, even after many years of research and experimentation with smaller databases. The winner of the 2014 challenge, on the other hand, achieved a 6.7% error rate [6]. This dramatic improvement was achieved through the use of deep convolutional nets. As a reference point, the winning architecture returns the ensemble prediction of seven networks. Each network is 27 layers deep, has about 6.8 million parameters each, and performs 1.5 billion add/multiplies to classify each image. With only slight oversimplification, one can say that the change from 17% to 6.7% from a previous system [3] was achieved by about trebling the number of layers and at the same time reducing the number of parameters tenfold for better generalization. This change is the

[5] Other measures of error have been proposed in the past and are given as secondary measures in current challenges as well.

outcome of interesting theoretical [1] and practical insights [6] about the tradeoff between expressiveness and generalization.

References

[1] S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. In *Proceedings of the 31st International Conference on Machine Learning*, pages 584–592, 2014.

[2] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning*, pages 148–156, 1996.

[3] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1106–1114, 2012.

[4] K. P. Murphy. *Machine Learning—A Probabilistic Perspective*. MIT Press, 2012.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. Technical Report 1409.4842 [cs.CV], arXiv, 2014.