CS 536 : Perceptrons - Complexity Analysis

16:198:536

We continue with the simplified formulation of the linear separator / perceptron problem from the previous notes: given a data set of m points for binary classification, where the 0-th feature component of each data point is taken to be constant, we are looking for a weight vector $\underline{w}$ such that sign($\underline{w}.\underline{x}$) correctly classifies all points in the training set. Given a linear separator specified by $\underline{w}$, it was convenient to define the **margin of** $\underline{w}$, the minimum distance to any training point. In the simplified setup, this could be given by

$$\gamma(\underline{w}) = \min_i \frac{|\underline{w}.\underline{x}^i|}{||\underline{w}||}. \tag{1}$$

Given a data set, it is convenient to define the **maximum margin** as $\gamma^* = \max_{\underline{w}} \gamma(\underline{w})$, taking the maximum over all linear separators $\underline{w}$. Note, if no linear separators exist, the maximum margin is undefined.

Continuing from the previous set of notes, we have the following result:

> Given a data set for binary classification, if a linear separator exists, then the Perceptron Learning Algorithm terminates in $T \leq (1/\gamma^*)^2$ steps. If no linear separator exists, then the algorithm fails to terminate and will loop forever.

This result is somewhat intuitive - the larger the maximum margin, the easier it is to find a linear separator, and the faster the algorithm converges. For data sets with very small maximum margins, any linear separator will be very near weight vectors $\underline{w}$ that are *not* linear separators (i.e., small perturbations in a linear separator will result in misclassifications) so it becomes much harder to identify valid linear separators. If there is no linear separator, the termination condition for the Perceptron Learning Algorithm is never met.

Sample Complexity

Recall the usual result that given a discrete, finite hypothesis space H, in order to guarantee that the true error differs from the training error by no more than ϵ, we need at least

$$m \geq \frac{1}{\epsilon^2} \log |H| \tag{2}$$

training points.

At first glance, this result doesn't seem applicable here, as the set of hypotheses (proposed weight vectors) seems continuous and therefore uncountably infinite. However, note that any weight vector $\underline{w}$ that emerges from the Perceptron Learning Algorithm will have the following form,

$$\underline{w} = y^{i_1}\underline{x}^{i_1} + y^{i_2}\underline{x}^{i_2} + \ldots + y^{i_T}\underline{x}^{i_T}, \tag{3}$$

where each i_t represents the index of the chosen misclassified data point at time t. There are only finitely many possible choices for each misclassified data point at time t (namely, m), and so there are only finitely many options for $\underline{w}$ as well. As such, the number of possible $\underline{w}$ is at most m^T, or observing the previous bound on T we have that

$$|H| \leq m^{\left(\frac{1}{\gamma^*}\right)^2}. \tag{4}$$

One way to interpret this is that the set of viable hypotheses (under the perceptron learning algorithm) is entirely **data-defined**, and in particular depends only on the number of data points and the maximum margin of the data. Note importantly here, the hypothesis is **independent of the underlying dimension!**

From the above relation we have that in any perceptron learning situation,

$$\frac{1}{\epsilon^2}\frac{1}{\gamma^{*2}}\log m \geq \frac{1}{\epsilon^2}\log |H|. \tag{5}$$

Given that this is true, we can meet the criteria for avoiding overfitting as long as $m \geq 1/\epsilon^2 1/\gamma^{*2}\log m$. This is true for all sufficiently large m, but can we provide a more informed bound to measure the sample complexity of perceptron learning? We have the following result:

If we want $|\text{err}(f) - \text{err}_{\text{train}}(f)| \leq \epsilon$, it suffices to take

$$m \geq \frac{1}{\epsilon^2}\frac{1}{\gamma^{*2}}\log^2\left(\frac{1}{\epsilon^2}\frac{1}{\gamma^{*2}}\right). \tag{6}$$

Proof: Let $\delta = \epsilon^2 * \gamma^{*2}$. We want to show that if $m \geq 1/\delta \log(1/\delta)^2$, then $m \geq 1/\delta \log m$. This is equivalent to showing that $\log m/m \leq \delta$. Since $\log m/m$ is a decreasing function (for sufficiently large m), it suffices to show that this result holds at the lower bound, i.e., that

$$\frac{\log\left[\frac{1}{\delta}\log\left(\frac{1}{\delta}\right)^2\right]}{\frac{1}{\delta}\log\left(\frac{1}{\delta}\right)^2} \leq \delta. \tag{7}$$

This simplifies in the following way:

$$\log\left[\frac{1}{\delta}\right] + 2\log\left[\log\frac{1}{\delta}\right] \leq \log\left(\frac{1}{\delta}\right)^2. \tag{8}$$

Taking the substitution $\Delta = \log(1/\delta) > 0$, this becomes

$$\Delta + 2\log\Delta \leq \Delta^2. \tag{9}$$

But this is obviously true for all sufficiently large values of Δ - thus the original holds for all sufficiently small values of δ

Hence perceptrons have the following nice properties:

- A simple learning rule with guaranteed convergence in finite time if a solution exists.
- Complexity of learning and the sample complexity is independent of the underlying dimension.

However, one problem that we run into at this point - as suggested at several points previously - is that if the maximum margin is quite small, γ^* near 0, the convergence of the learning algorithm can take a very long time, as well as requiring a lot of sample data in order to identify an accurate linear separator. Additionally, we lack any guarantees at this point that perceptron learning is efficient, and can be done in polynomial time. Are there any alternatives?

A Linear Programming Approach

Consider the following alternative specification for a linear separator: we want to identify a weight vector $\underline{w}$ such that for $y^i = 1, \underline{w}.\underline{x}^i > 0$, and for $y^i = -1, \underline{w}.\underline{x}^i < 0$. We can unify these two conditions in the following way - that we want to find a weight vector $\underline{w}$ such that

$$\forall i : y^i\left(\underline{w}.\underline{x}^i\right) > 0. \tag{10}$$

Note that each of the above can be expanded as

$$\forall i : (y^i x_0^i) * w_0 + (y^i x_1^i) * w_1 + \ldots + (y^i x_k^i) * w_k > 0, \tag{11}$$

i.e., the above represents a system of linear inequalities in terms of the unknown variables $\{w_j\}$. Solving for an assignment for these variables to satisfy each of these inequalities is exactly the domain of **Linear Programming**. Typical LP problems take them for

$$\max_{\underline{x}} \underline{r}.\underline{x} \text{ such that } A\underline{x} \geq \underline{b}, \tag{12}$$

where $\underline{r}, \underline{b}$ are vectors of real values specified by the problem, and A represents some matrix of coefficients on the unknown variables in $\underline{x}$. Linear programs have been studied extensively and a variety of algorithms exist for generating a solution, given $\underline{r}, \underline{b}, A$, in polynomial time. Note that in this case, we are only specifying the constraints on the unknown $\underline{w}$ - we have yet to specify an objective function to optimize. In fact, any (convex) objective function will do - the mechanics of linear programming algorithms can still be applied to the problem of generating *feasible points* that satisfy the system of inequalities. The specific objective function simply determines *which* of the feasible points will be returned by the algorithm.

Utilizing linear programming techniques in this case, we can apply polynomial time *interior point* algorithms or related algorithms to generate feasible solution weights (if such weights exist), and we can do so in such a way that requires at least

$$m \geq \frac{1}{\epsilon^2} k \tag{13}$$

training points to achieve ϵ-generalizability.

The question that concerns us now is this: given the option of the perceptron learning algorithm approach, with potentially non-polynomial time complexity for learning and a sample complexity of $1/\epsilon^2 1/\gamma^2 \log(1/\epsilon^2 1/\gamma^2)^2$ (independent of the ambient dimension k) and the linear programming approach, with polynomial time complexity but sample complexity of k/ϵ^2 - which approach is better? In various regimes (large k, large γ, or small k, small γ, etc) how do we decide which approach to take - especially since γ may not be known! Or could we combine these approaches and achieve the best of both worlds?

An LP Specification

As indicated - to specify the perceptron as a linear programming problem, the most important aspect is the feasibility - correct classification of the training data requires that all the linear constraints be satisfied. A specification of the LP might be therefore given like

$$\begin{aligned} &\min_{\underline{w}} w_0 \\ &(s.t.) \quad \forall i y^i (\underline{w}.\underline{x}^i) > 0. \end{aligned} \tag{14}$$

The choice of w_0 as objective function here is arbitrary.

One thing that can complicate implementing this solution in an immediate raw way is the fact that the inequality constraints are strict - most LP techniques rely on the constraints being specified in a non-strict way, $A\underline{x} \geq \underline{b}$. This non-strict constraint makes the set of feasible solutions closed, if not compact, which makes it easier to approximate solutions with sequences that converge to the solution. To illustrate why this is potentially an issue - note that if all the inequalities were not strict, i.e., $y^i(\underline{w}.\underline{x}^i) \geq 0$, then $\underline{w} = 0$ would be a feasible solution to the problem - but this would not be a valid binary classifier in this framework.

We would like to be able to reframe this LP in a more traditional way, so that classical LP solutions could be applied. In order to do so, note the following - if $\underline{w}$ is a feasible solution, i.e., satisfies all the constraints, then $\rho * \underline{w}$ is also a

feasible solution for any $\rho > 0$. That is, any solution $\underline{w}$ may be scaled up by a positive factor, and will remain a valid classifier on the training data. This freedom allows us to rescale the problem in the following way: for any feasible solution $\underline{w}$, we can define the 'slack' δ of that solution as

$$\delta = \min_i y^i(\underline{w}.\underline{x}^i) > 0. \tag{15}$$

In this case, defining $\underline{w}' = \underline{w}/\delta$, we see that $\underline{w}'$ satisfies

$$\forall i : y^i(\underline{w}'.\underline{x}^i) = \frac{1}{\delta} y^i(\underline{w}.\underline{x}^i) \geq \frac{1}{\delta}\delta = 1, \tag{16}$$

or $y^i(\underline{w}'.\underline{x}^i) \geq 1$.

Hence we have the following rescaling:

> If there are any solutions $\underline{w}$ such that $y^i(\underline{w}.\underline{x}^i) > 0$ for all i, then there are solutions $\underline{w}$ such that $y^i(\underline{w}.\underline{x}^i) \geq 1$ for all i.

Hence we can give a restatement of the initial LP in a more 'traditional' form:

$$\begin{aligned} &\min_{\underline{w}} \; w_0 \\ &(s.t.) \quad \forall i y^i(\underline{w}.\underline{x}^i) \geq 1. \end{aligned} \tag{17}$$

Note - the solution to this LP may be distinct from the solution to the previous, but in either case the solution will be a valid linear separator on the training data. The distinction is simply that this version may be solved with classical linear programming solutions (frequently available in numerical analysis libraries).