The Perceptron

The perceptron implements a binary classifier $f : \mathbb{R}^D \mapsto \{+1, -1\}$ with a linear decision surface through the origin:

$$f(x) = \mathrm{step}(\boldsymbol{\theta}^\top \boldsymbol{x}). \tag{1}$$

where

$$\mathrm{step}(z) = \begin{cases} 1 & \text{if} \;\; z \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

Using the zero-one loss

$$L(y, f(x)) = \begin{cases} 0 & \text{if} \;\; y = f(x) \\ 1 & \text{otherwise,} \end{cases}$$

the empirical risk of the perceptron on training data $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ is just the number of misclassified examples:

$$R_{\mathrm{emp}}(\boldsymbol{\theta}) = \sum_{i \in (1,2,\ldots,N) \,:\, y_i \neq \mathrm{step}\left(\boldsymbol{\theta}^T \boldsymbol{x}_i\right)} 1.$$

The problem with this is that $R_{\mathrm{emp}}(\theta)$ is not differentiable in θ, so we cannot do gradient descent to learn θ.

To circumvent this, we use the modified empirical loss

$$R_{\mathrm{emp}}(\boldsymbol{\theta}) = \sum_{i \in (1,2,\ldots,N) \,:\, y_i \neq \mathrm{step}\left(\boldsymbol{\theta}^T \boldsymbol{x}_i\right)} -y_i \left(\boldsymbol{\theta}^T \boldsymbol{x}_i\right). \tag{2}$$

This just says that correctly classified examples don't incur any loss at all, while incorrectly classified examples contribute $\left| \, \boldsymbol{\theta}^T x_i \, \right|$, which is some sort of measure of confidence in the (incorrect) labeling. [1]

We can now use gradient descent to learn $\boldsymbol{\theta}$. Starting from an arbitrary $\boldsymbol{\theta}^{(0)}$, we update our parameter vector according to

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} R \big|_{\boldsymbol{\theta}^{(t)}} \,,$$

where η, called the learning rate, is a parameter of our choosing. The gradient of (2) is again a sum over the misclassified examples:

$$\nabla_{\boldsymbol{\theta}} R_{\mathrm{emp}}(\boldsymbol{\theta}) = \sum_{i \in (1,2,\ldots,N) \,:\, y_i \neq \mathrm{step}\left(\boldsymbol{\theta}^T \boldsymbol{x}_i\right)} -y_i \boldsymbol{x}_i.$$

[1] A slightly more principled way to look at this is to derive this modified risk from the hinge loss $L(y, \theta^T x) = \max\left(\, 0, -y\left(\boldsymbol{\theta}^T \boldsymbol{x}\right) \right)$.

If we let $M \subset S$ be the set of training examples misclassified by $\boldsymbol{\theta}^{(t)}$, the update rule can be written very simply as

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \eta \sum_{(x_i, y_i) \in M} y_i \boldsymbol{x}_i.$$

One issue that remains is how to implement a bias term generalizing to linear classifiers that do not necessarily cross the origin:

$$f(x) = \mathrm{step}(\theta_0 + \boldsymbol{\theta}^\top \boldsymbol{x}). \tag{3}$$

The simplest solution to this is to append a constant (0'th) element 1 to each input vector and incorporate θ_0 in $\boldsymbol{\theta}$. This reduces (3) to the original (1) except that the dimensionality of all the vectors has increased by one.

On-line perceptron (not examinable)

What we described above is the batch perceptron. The perceptron has a more prominent role in the world of online learning [1]. In online learning there is no distinction between the training set and testing set. The input is a continuous stream of examples, and the algorithm has to make a prediction immediately after x_i arrives. Before the next example arrives, the true label y_i is presented, and the algorithm can update its internal parameters to reflect what it has learnt from its success or failiure in predicting y_i.

The online perceptron is about as simple as a learning algorithm gets:

```
w=0
for i=1 to m
  predict y=step(w*x_i)
  if (y=-1 and y_i=1) w=w+x_i
  if (y=1 and y_i=-1) w=w-x_i
end
```

(note that `w` and `x_i` are vectors and `*` is the dot product). Remarkably, it is still a powerful learning algorithm. It is possible to prove that, provided the data lies withing a ball of radius R centered on the origin and is separable with margin γ (i.e. there exists a separating hyperplane with normal vector $\boldsymbol{w}$ such that $|\,\boldsymbol{w} \cdot x_i\,| / \|\, w \,\| \geq \gamma$ for all examples), the online perceptron will make no more than $\lceil M/\gamma^2 \rceil$ errors, regardless of the number of examples.

References

[1] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.