04 - Linear Classification

Ethan Graham

20th June 2023

Line Parametrization

We can either parametrize a line with like this: $u, v \in \mathbb{R}^2\, s.t.\, au + bv + c = 0,\, and\, a^2 + b^2 = 1$ Or like this: $\mathbf{n} = \begin{pmatrix} a \\ b \end{pmatrix}$ with point (u_0, v_0)

We can define the **signed distance** of a point to a line

$h = \mathbf{n} \cdot (u_1 - u_0, v_1 - v_0) = au_1 + bv_1 + c$

if $h > 0$ then it is on one side, if $h < 0$ then is on the other.

If we let $\tilde{x} = (1, x_1, ..., x_D)$ and $\tilde{w} = (w_0, w_1, ..., w_D)$ with $w_1^2 + ... + w_D^2 = 1$ we get $h = \mathbf{\tilde{w}} \cdot \mathbf{\tilde{x}}$

A hyperplane is defined by all points that satisfy $h = \mathbf{\tilde{w}} \cdot \mathbf{\tilde{x}} = \mathbf{0}$

Perceptron

Assuming two classes, one positive and one negative. We want to find $\mathbf{\tilde{w}}$ such that - $\mathbf{\tilde{w}} \cdot \mathbf{\tilde{x}} > \mathbf{0}$ for all/most positive samples - $\mathbf{\tilde{w}} \cdot \mathbf{\tilde{x}} < \mathbf{0}$ for all/most negative samples

I.e. minimize $E(\mathbf{w}) = -\sum_{n=1}^{N} sign(\mathbf{\tilde{w}} \cdot \mathbf{\tilde{x}_n}) t_n$ Which we do with the following procedure: - set $\mathbf{\tilde{w}}$ to 0 - iteratively pick a random index n a. if $\mathbf{\tilde{x}_n}$ is correctly classified, do nothing b. otherwise $\mathbf{\tilde{w}_{t+1}} = \mathbf{\tilde{w}_t} + t_n \mathbf{\tilde{x}_n}$

At test time, $y(\mathbf{x}; \mathbf{\tilde{w}}) = sign(\mathbf{\tilde{w}} \cdot (1|\mathbf{x}))$

Convergence theorem

If $\exists \gamma > 0$ and $w\star$ with $\|w\star\| = 1$ such that $\forall n$, $t_n(\mathbf{w} \star \cdot \mathbf{x_n}) > \gamma$, then the perceptron algorithm makes at most $\frac{R^2}{\gamma^2}$ errors where $R = max_n \|x_n\|$. Here γ is the margin.

Problem with the Perceptron and Logistic regression

Since we initialize randomly, there are infinitely many different possible solutions/convergences. The perceptron has no way of favouring one over the other.

What we decide to do is replace the step function with a **smooth curve** like a sigmoid. We now get the prediction $y(\mathbf{x}; \tilde{\mathbf{w}}) = \sigma(\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}})$

And out cost function becomes the **cross-entropy** $E(\tilde{\mathbf{w}}) = -\sum_n \{t_n ln(y_n) + (1-t_n) ln(1-y_n)\}$ Where $y_n = y(\mathbf{x_n}; \tilde{\mathbf{w}})$. This is a convex function, and therefore its gradient is easy to compute *easy to optimize*

Given the definiton of the sigmoid function, we get $y(\mathbf{x_n}; \tilde{\mathbf{w}}) = \frac{1}{1+exp(-\tilde{\mathbf{w}} \cdot \mathbf{x})}$ And this result can be interpreted as the **probability that that x belongs to one class or the other**. Logistic regression finds what's called the **maximum likelihood solution** under the assumption that the noise is Gaussian (*i.e. the noise follows a Normal distribution*)

We can extend the definition very easily to k classes by making multiple classifiers. These classifiers form regions.

The multiclass entropy is just the sum of entropies over all k classifiers. This multiclass entropy is still convex and easy to optimize.

Problems with Logistic Regression

Logistic regression tries to minimize the error-rate at training time. This can result in poor results during testing.