CS142: Machine Learning Spring 2017

Lecture 9

Instructor: Pedro Felzenszwalb *Scribes: Dan Xiang, Tyler Dae Devlin*

Approaches to Classification

Classification models may be divided into three different categories: models that are *generative*, models that are *discriminative*, and models that make use of a *discriminant function*. Here we briefly introduce what these terms mean.

Generative models 生成模型

A generative approach to classification entails modeling the full joint distribution $p(x, y) = p(x|y)p(y)$ of the product space of inputs and outputs. If we have the joint distribution in hand, then we know everything that we could ever possibly want to know about the uncertainty associated with the problem. In particular, we can marginalize to find $p(x)$ or $p(y)$, and we can condition to find $p(y|x)$ or $p(x|y)$. Furthermore, we can sample from $p(x, y)$ to *generate* new data.

Discriminative models 判别模型

In contrast to generative models, discriminative models aim to directly estimate the posterior class probabilities $p(y|x)$. In other words, discriminative methods infer decision regions and decision boundaries from data without making reference to the distribution on inputs, $p(x)$.

Discriminant functions

A discriminant function maps inputs x to classes y. In contrast to generative and discriminative methods, classification techniques that use a discriminant function make no attempt whatsoever to model the underlying probability distributions. Instead, a training data set is used to build a discriminant function with low in-sample error, with the hope that this function performs well out-of-sample as well. Linear discriminant functions are the subject of this lecture.

Linear Classifiers

When faced with a new learning problem, often the simplest and best place to start is with models that rely on linearity in some way. Just as we started this course with linear models for regression, we now turn to linear models for classification.

Let $X = \mathbb{R}^D$ be the input space and $C : X \to \{-1, +1\}$ be a binary classifier, defined by

$$C(x) = \begin{cases} +1 & w^T x \geq b \\ -1 & w^T x < b, \end{cases}$$

where $w = (w_1, \ldots, w_D) \in \mathbb{R}^D$ can be thought of as a vector of weights and $b \in \mathbb{R}$ (sometimes called the *bias* term when it appears on the other side of the inequality) defines the location of the decision boundary.

In words, C takes a linear combination of the input features and checks to see whether the result exceeds a certain threshold. This classifier is very similar to the simple linear regression model that we considered before, except that the output is constrained to be ± 1. The boundary induced by this classifier is a $D - 1$-dimensional hyperplane perpendicular to w. The distance from the origin to the hyperplane depends on $\|w\|$ and b.

Nonlinear classification

Just as we were able to fit nonlinear real-valued functions using linear regression in combination with a set of basis functions, we can redefine the classifier $C : X \to \{-1, +1\}$ by

$$C(x) = \begin{cases} +1 & w^T \phi(x) \geq b \\ -1 & w^T \phi(x) < b \end{cases}$$

for some non-linear feature map $\phi : X \to X'$.

Suppose we have a data set consisting of points in $\mathbb{R}^2$ where all the positive $(+1)$ examples fall within some ball centered around a point $\mu \in \mathbb{R}^2$, and all negative (-1) examples are outside this region. You should convince yourself that no line in $\mathbb{R}^2$ will perform well in this scenario. We want to define our classifier so

that $C(x) = +1$ whenever $\|x - \mu\| \leq r$. Now observe

$$\begin{aligned}\|(x-\mu)\|^2 &= (x-\mu)^T(x-\mu) \\ &= x^Tx + \mu^T\mu - 2x^T\mu \\ &= \begin{bmatrix} x \\ x^Tx \\ 1 \end{bmatrix}^T \cdot \begin{bmatrix} -2\mu \\ 1 \\ \mu^T\mu \end{bmatrix},\end{aligned}$$

where the last two vectors are both in $\mathbb{R}^{D+2}$ ($x^Tx \in \mathbb{R}$ and $1 \in \mathbb{R}$ are appended to the vector $x \in \mathbb{R}^D$). Thus, we can write

$$\|(x-\mu)\|^2 = w^T\phi(x),$$

where $\phi : \mathbb{R}^D \to \mathbb{R}^{D+2}$ and w are defined by

$$\phi(x) = \begin{bmatrix} x \\ x^Tx \\ 1 \end{bmatrix}, \quad w = \begin{bmatrix} -2\mu \\ 1 \\ \mu^T\mu \end{bmatrix}.$$

So it is in fact possible to obtain nonlinear decision boundaries by using an appropriate feature transform ϕ.

Perceptron Algorithm

Now that we have given some evidence that linear classifiers can produce decision boundaries of varying complexity, we turn to the problem of actually finding the weight vector w that defines a classifier.

We first make a slight modification to our original classifier by defining $\phi(x) = \begin{bmatrix} 1 \\ x \end{bmatrix}$, and letting the classifier C be

$$C(x) = \begin{cases} +1 & w^T\phi(x) \geq 0 \\ -1 & w^T\phi(x) < 0, \end{cases}$$

where we now have $w = (w_0, w_1, \ldots, w_D) \in \mathbb{R}^{D+1}$. We have effectively incorporated the threshold b into the weight vector.

Given a training data set that is linearly separable in the ϕ-transformed feature space, the *perceptron learning algorithm* (which is now over 50 years old) outputs a set of weights that perfectly classify the training data. The algorithm uses a simple iterative method as follows.

Algorithm 1 Perceptron Learning Algorithm

Precondition: $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is a linearly separable set of labeled examples with $x_i \in \mathbb{R}^D$ and $y_i \in \{-1, +1\}$.

```
function PLA(T)
    Initialize w ← 0
    while ∃ (x_i, y_i) ∈ T such that y_i ≠ sign(w^T x_i) do
        Pick a random (x_j, y_j) ∈ T
        if y_j(w^T x_j) < 0, meaning that x_j is misclassified then
            Update w ← w + y_j x_j
    return w
```

The perceptron learning algorithm cycles through the points in the training set. Every time it encounters a misclassified point (x_j, y_j), it updates the weights w so as to be "less wrong" with respect to (x_j, y_j). Note, however, that after a weight update, it is possible for points other than (x_j, y_j) that were previously correctly classified to become misclassified. The following theorem states that, in spite of this last observation, the perceptron works (given linearly separable data).

Linear Separation Theorem

Idea: If the training examples are linearly separable with a nonzero margin, the algorithm converges to a linear separator.

Theorem: Suppose that for each $(x_i, y_i) \in T$, we have $\|x_i\| \leq R \in \mathbb{R}$ a constant, and that there exists a $\bar{w} \in \mathbb{R}^D$ with margin γ, i.e. $\gamma \in \mathbb{R}^+$ such that $\|\bar{w}\| = 1$ and $y_i(\bar{w}^T x_i) \geq \gamma$ for all $i \in \{1, \ldots, n\}$. Then the number of mistakes that are made before the algorithm terminates is at most R^2/γ^2.

Proof. Let w^k denote the state of the weight vector just before the kth update. For example, $w^1 = 0$, since before the 1st update occurs, the weights are still initialized to all be 0. Then we have

$$\begin{aligned}
(w^{k+1})^T \bar{w} &= (w^k + y_i x_i)^T \bar{w} && \text{(by the weight update rule)} \\
&= (w^k)^T \bar{w} + (y_i x_i)^T \bar{w} && \text{(distribute)} \\
&\geq (w^k)^T \bar{w} + \gamma && \text{(by the assumption that } y_i(\bar{w}^T x_i) \geq \gamma).
\end{aligned}$$

By induction on k, we obtain

$$(w^{k+1})^T \bar{w} \geq k\gamma.$$

The Cauchy-Schwarz inequality along with this last result give

$$\|w^{k+1}\| = \|w^{k+1}\| \cdot \|\bar{w}\| \geq (w^{k+1})^T \bar{w} \geq k\gamma. \tag{1}$$

The squared norm is then

$$\begin{aligned}\|w^{k+1}\|^2 &= \|w^k + y_i x_i\|^2 \\ &= (w^k + y_i x_i)^T (w^k + y_i x_i) \\ &= (w^k)^T (w^k) + y_i^2 x_i^T x_i + 2(w^k y_i x_i) \\ &\leq \|w^k\|^2 + R^2.\end{aligned}$$

Another induction on k shows

$$\|w^{k+1}\|^2 \leq kR^2. \tag{2}$$

Combining (1) and (2), we have

$$\boxed{k \leq \frac{R^2}{\gamma^2}}$$

□