Lecture 8. Perceptron and Support Vector Machine

Prof. Alan Yuille

Spring 2014

Outline

1. Perceptron
2. Support Vector Machine

1 Perceptron

The Perceptron Algorithm dates back to the 1950's. It was very influential (i.e. hyped) - and unrealistic claims were made about its effectiveness. But it is very important as a starting point for one style of machine learning.

The Perceptron was criticized (Minksy and Papert) because it was not all to represent all decision rules – i.e. you cannot always separate data into positive and negative by using a plane. But, from the point of view of generalization, it is good that perceptrons cannot learn everything! In technical language, this means that perceptrons have *limited capacity* and this enables good generalization for some types of data.

Perceptrons can only represent a restricted set of decision rules (e.g. separation by hyperplane). This is a limitation and a virtue. If we can find a separating hyperplane, then it is probably not due to chance alignment of the data (provided $n > (d+1)$), and so it is likely to generalize. In Learning Theory (Vapnik) the quantity $(d+1)$ is the VC dimension of perceptrons and is a measure of the *capacity* of perceptrons. There is a hypothesis space of classifiers – the set of all perceptrons in this case – and this hypothesis space has a *capacity* which is $d+1$ for perceptrons. To ensure generalization, you need much more training data than the capacity of the hypothesis space that you are using. We will return to this is later lectures.

An alternative is to use Multilevel Perceptron, see previous lecture.

1.1 Linear Classifiers

A dataset contains N samples: $\{\ (x_\mu, y_\mu) : \mu = 1 \text{ to } N\ \}$, $y_\mu \in \{\pm 1\}$
Can we find a linear classifier that separates the position and negative examples?

E.g., a plane $\vec{a} \cdot \vec{x} = 0$, see figure (**??**), which separates the data with a decision rule $\hat{y}(\vec{x}) = sign(\vec{a} \cdot \vec{x})$
s.t. $\quad \vec{a} \cdot \vec{x}_\mu \geq 0$, if $y_\mu = +1$
s.t. $\quad \vec{a} \cdot \vec{x}_\mu \leq 0$, if $y_\mu = -1$

Plane goes through the origin ($\vec{a} \cdot \vec{0} = 0$) (special case, can be relaxed later).

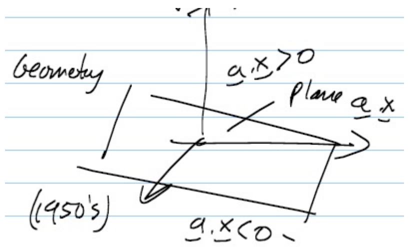

Figure 1: A plane $\vec{a} \cdot \vec{x} =$ separates the space into two regions $\vec{a} \cdot \vec{x} > 0$ and $\vec{a} \cdot \vec{x} < 0$.

1.2 The Perceptron Algorithm

First, we need to replace the negative examples by positive examples: if $y_\mu = -1$, set $\vec{x}_\mu \to -\vec{x}_\mu$, $y_\mu \to -y_\mu$.
Require that sign$(\vec{a} \cdot \vec{x}_\mu) = y_\mu$, which is equivalent to sign$(-\vec{a} \cdot \vec{x}_\mu) = -y_\mu)$.

The perceptron algorithm reduces to finding a plane s.t. $(\vec{a} \cdot \vec{x}_\mu) \geq 0$, for $\mu = 1, ..., N$.

Note: the vector $\vec{a}$ need not be unique. It is better to try to maximize the margin (see later this lecture), which requires finding $\vec{a}$ with $|\vec{a}| = 1$, so that $(\vec{a} \cdot \vec{x}_\mu) \geq m, \ \forall \ \mu = 1, ..., N$ for the maximum value of m.

From a geometrical point of view, we can make the following claim.

Claim: If $\vec{a}$ is a unit vector $|\vec{a}| = 1$, then $\vec{a} \cdot \vec{y}$ is the sign distance of $\vec{y}$ to the plane $\vec{a} \cdot \vec{x} = 0$, see figure (**??**). I.e. $\vec{a} \cdot \vec{y} > 0$, if $\vec{y}$ is above plane; and $\vec{a} \cdot \vec{y} < 0$, if $\vec{y}$ is below plane.

Proof: write $\vec{y} = \lambda \vec{a} + \vec{y}_p$, where $\vec{y}_p$ is the projection of $\vec{y}$ into the plane. By definition $\vec{a} \cdot \vec{y}_p = 0$, hence $\lambda = (\vec{a} \cdot \vec{y})/(\vec{a} \cdot \vec{a}) = (\vec{a} \cdot \vec{y})$, if $|\vec{a}| = 1$.

The perceptron algorithm has the following steps:

```
Initialize: $\vec{a}(0) = 0$
loop over $\mu = 1$ to $N$
If $\vec{x}_\mu$ is misclassified (i.e. $sign(\vec{a} \cdot \vec{x}_\mu) < 0$), set $\vec{a} \to \vec{a} + \vec{x}_\mu$,
Repeat until all samples are classified correctly.
```

Note: instead of changing the signs of the data points to require that sign$(\vec{a} \cdot \vec{x}_\mu) = y_\mu$, the algorithm can be modified to update the weights with negative sign: $\vec{a} \to \vec{a} - \vec{x}_\mu$, which is equivalent.

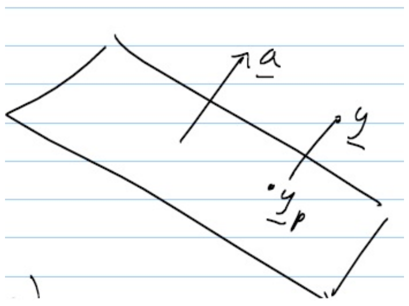

Figure 2: $\vec{y}_p$ is the projection of point $\vec{y}$ onto the plane $\vec{a}\cdot\vec{x}=0$. This means that $\vec{y}-\vec{y}_p \propto \vec{a}$, i.e. the vector joining $\vec{y}$ and $\vec{y}_p$ is parallel to the normal $\vec{a}$ to the plane. If $|\vec{a}|=1$, then the sign projection is $\vec{a}\cdot(\vec{y}-\vec{y}_p)=\vec{a}\cdot\vec{y}$ (since $\vec{y}_p$ lies on the plane and so $\vec{a}\cdot\vec{y}_p=0$). The sign projection is positive $\vec{a}\cdot\vec{y}>0$ if $\vec{y}$ lies above the plane and is negative $\vec{a}\cdot\vec{y}<0$ if $\vec{y}$ lies below the plane.

1.3 Novikov's Theorem (advanced topic)

Novikov's Theorem: The perception algorithm will converge to a solution weight $\vec{a}$ that classifies all the samples correctly (provided this is possible).

Proof.
Let $\hat{\vec{a}}$ be a separating weight $(m>0)$
Let $m=\min_\mu \quad \hat{\vec{a}}\cdot\vec{x}_\mu$
Let $\beta^2=\max_\mu|\vec{x}_\mu|^2$

Suppose that $\vec{x}_t$ is misclassified at time t so $\vec{a}\cdot\vec{x}_t<0$. Then $\vec{a}_{t+1}-(\beta^2/m)\hat{\vec{a}}=\vec{a}_t-(\beta^2/m)\hat{\vec{a}}+\vec{x}_t$.

$\|\vec{a}_{t+1}-\frac{\beta^2}{m}\hat{\vec{a}}\|^2=\|\vec{a}_t-\frac{\beta^2}{m}\hat{\vec{a}}\|^2+\|\vec{x}_t\|^2-2(\vec{a}_t-\frac{\beta^2}{m}\hat{\vec{a}})\cdot\vec{x}_t$

Using $\|\vec{x}_t\|^2\le\beta^2$, $\vec{a}_t\cdot\vec{x}_t<0, -\hat{\vec{a}}\cdot\vec{x}_t<-m$,

it follows that $\|\vec{a}_{t+1}-\frac{\beta^2}{m}\hat{\vec{a}}\|^2\le\|\vec{a}_t-\frac{\beta^2}{m}\hat{\vec{a}}\|^2+\beta^2-2\frac{\beta^2 m}{m}$.

Hence $\|\vec{a}_{t+1}-\frac{\beta^2}{m}\hat{\vec{a}}\|^2\le\|\vec{a}_t-\frac{\beta^2}{m}\hat{\vec{a}}\|^2-\beta^2$.

So, each time we update a weight, we reduce the quality $\|\vec{a}_t-\frac{\beta^2}{m}\hat{\vec{a}}\|^2$ by a fixed amount β^2. But $\|\vec{a}_0-\frac{\beta^2}{m}\hat{\vec{a}}\|^2$ is bounded above by $\frac{\beta^4}{m^2}\|\hat{\vec{a}}\|^2$. So, we can update the weight at most $\frac{\beta^2}{m^2}|\hat{\vec{a}}^2|$ times. This guarantees convergence.

2 Support Vector Machine

2.1 Linear Separation: Margins and Duality

Support Vector Machine (SVM) is a modern approach to linear separation. Suppose you have
Data: $\{(\vec{x}_\mu, y_\mu) : \mu = 1 \text{ to } N\}, \qquad y_\mu \in \{-1, 1\}$
Hyperplane: $< \vec{x} : \vec{x} \cdot \vec{a} + b = 0 > \qquad |\vec{a}| = 1$
The signed distance of a point $\vec{x}$ to the plane is $\vec{a} \cdot \vec{x} + b$.
If we project the line $\vec{x}(\vec{\lambda}) = \vec{x} + \lambda\vec{a}$, it hits plane when $\vec{a} \cdot (\vec{x} + \lambda\vec{a}) = -b$. Follows that $\lambda = -(\vec{a} \cdot \vec{x} + b)/|\vec{a}|^2$, and if $|\vec{a}| = 1$, then $\lambda = -(\vec{a} \cdot \vec{x} + b)$.
In SVM we seek a classifier with biggest margin:

$$\max_{\vec{a}, b, |\vec{a}|=1} C \quad \text{s.t.} \;\; y_\mu(\vec{x}_\mu \cdot \vec{a} + b) \geq C, \;\forall \mu \geq 1 \text{ to } N$$

I.e, the positive examples are at least distance C above the plane, and negative examples are at least C below the plane, see figure (**??**).

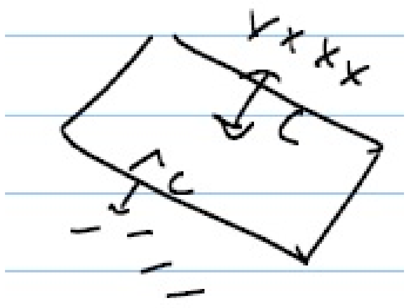

Figure 3: Want positive examples to be above the plane by more than C, and negative examples below the plane by more than C.

Having a large margin is good for generalization because there is less chance of an accidental alignment.
Perfect separation is not always possible. Let's allow for some data points to be misclassified. We define the *slack variables* $\{z_1, ..., z_n\}$ allow data points to move in direction $\vec{a}$, so that they are on the right side of the margin, see figure (**??**).
Criterion:

$$\max_{a, b, |\vec{a}|=1} C \;\; \text{s.t.} \;\; y_\mu(\vec{x}_\mu \cdot \vec{a} + b) \geq C(1 - z_\mu), \forall \mu \in \{1, N\} \quad \text{s.t.} \;\; z_\mu \geq 0, \forall \mu$$

.
Alternately, $y_\mu\{(\vec{x}_\mu + C z_\mu \vec{a}) \cdot \vec{a} + b\} \geq C$, which is like moving $\vec{x}_\mu$ to $\vec{x}_\mu + z_\mu \vec{a}$.

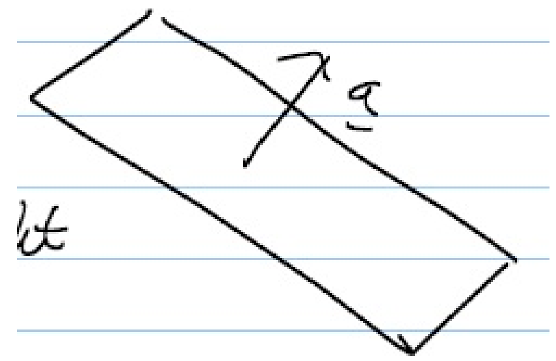

Figure 4: Datapoints can use slack variables to move in the direction of the normal $\vec{a}$ to the plane so that they lie on the right side of the margin. But the datapoints must pay a penalty of C times the size of the slack variable.

But, we must pay a penalty for using slack variables. E.g, a penalty $\sum_{\mu=1}^{N} z_\mu$. If $z_\mu = 0$, then the data point is correctly classified and is past the margin. If $z_\mu > 0$, then the data point is on the wrong side of the margin, and so had to be moved.

2.2 The Max-Margin Criterion

Here the task is to estimate several quantities simultaneously:
(1) The plane $\vec{a}, b$
(2) The margin C
(3) The slack variables $\{z_\mu\}$

We need a criterion that maximizes the margin and minimizes the amount of slack variables used. We absorb C into $\vec{a}$ by $\vec{a} \to a/c$ and remove the constraint $|a| = 1$. Hence, $C = \frac{1}{|\vec{a}|}$.

The Max-Margin Criterion:
$\min \frac{1}{2} \sum \vec{a} \cdot \vec{a} + \gamma \sum_\mu z_\mu$
s.t. $y_\mu(\vec{x}_\mu \cdot \vec{a} + b) \geq 1 - z_\mu, \quad \forall\ \mu z_\mu \geq 0$

First, we need to solve the *Quadratic Primal Problem* using Lagrange multipliers:

$L_p(\vec{a}, b, z; \alpha, \tau) = \frac{1}{2}\vec{a} \cdot \vec{a} + \gamma \sum_\mu z_\mu - \sum_\mu \alpha_\mu \{y_\mu(\vec{x}_\mu \cdot \vec{a} + b) - (1 - z_\mu)\} - \sum_\mu \tau_\mu z_\mu$

The $\{\alpha_\mu\}$ and $\{\tau_\mu\}$ are Lagrange parameters needed to enforce the inequality constraints. We require that $\alpha_\mu \geq 0, \tau_\mu \geq 0, \forall \mu$.

The function $L_p(\vec{a}, b, z; \alpha, \tau)$ should be *minimized* with respect to the *primal variables* $\vec{a}, z$ and *maximized* with respect to the dual variables α, τ. Note this means that if the constraints are satisfied then we need to set the corresponding lagrange parameter to be zero (to maximize). For example, if $y_\mu(\vec{x}_\mu \cdot \vec{a} + b) - (1 - z_\mu) > 0$ for some μ then we set $\alpha_\mu = 0$

because the term $-\alpha_\mu\{y_\mu(\vec{x}_\mu \cdot \vec{a} + b) - (1 - z_\mu)\}$ is non-positive, and so the maximum value occurs when $\alpha_\mu = 0$. But if the constraint is not satisfied – e.g., $y_\mu(\vec{x}_\mu \cdot \vec{a} + b) - (1 - z_\mu) < 0$ – then the lagrange parameter will be positive. So there is a relationship between the lagrange parameters which are positive (non-zero) and the constraints which are satisfied. This will have important consequences.

2.3 Support Vectors

L_p is a function of the primal variable $\vec{a}, b, \{z_\mu\}$ and the Lagrange parameters $\{\alpha_\mu, \tau_\mu\}$. There is no analytic solution for these variables, but we can use analytic techniques to get some understanding of their properties.
$\dfrac{\partial L_p}{\partial \vec{a}} = 0 \Rightarrow \hat{\vec{a}} = \sum_\mu \alpha_\mu y_\mu \vec{x}_\mu$
$\dfrac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_\mu \alpha_\mu y_\mu = 0$
$\dfrac{\partial L_p}{\partial z_\mu} = 0 \Rightarrow \alpha_\mu = \gamma - \hat{\tau_\mu}, \forall \mu$

The classifier is: $sign < \hat{\vec{a}} \cdot \vec{x} + \hat{b} >= sign < \sum_\mu \alpha_\mu y_\mu \vec{x}_\mu \cdot \vec{x} + b >$, by using the equation $\frac{\partial L_p}{\partial \vec{a}} = 0$.

Given that the solution depends only on the vectors $\vec{x}_\mu$ for which $\alpha_\mu \neq 0$, we call them *support vectors*.

The constraints are $y_\mu(\vec{x}_\mu \cdot \tilde{\vec{a}} + \tilde{b}) \geq 1 - \hat{z_\mu}$, $\hat{z_\mu} \geq 0$, and $\hat{\tau_\mu} \geq 0$.

By theory of Quadratic Programming, $\alpha_\mu > 0$, only if either:
(i) $z_\mu > 0$, i.e, slack variable is used.
(ii) $z_\mu, \text{but} y_\mu(\vec{x}_\mu \cdot \tilde{\vec{a}} + \tilde{b}) = 1$, i.e. data point is on the margin.

The classifier depends only on the support vectors, the other data points do not matter, see figure (**??**)

This is intuitively reasonable - the classifier must pay close attention to the data that is difficult to classify - the data near the boundary.
This differs from the probabilistic approach & when we learn probability models for each class and then use the Bayes classifier (but we can derive it as an approximation to probabilistic methods – see end of this lecture). Note that the support vectors include all the datapoints which have positive slack variables – i.e. which are on the wrong sides of the margin.

2.4 Dual and Relation to Primal

We can solve the problem more easily in the dual formulation – this is a function of Lagrange multipliers only.

$L_p = \sum_\mu \alpha_\mu - \frac{1}{2} \sum_{\mu,\nu} \alpha_\mu \alpha_\nu y_\mu y_\nu \vec{x}_\mu \vec{x}_\nu$ s.t $0 \leq \alpha_\mu \leq \tau, \sum_\mu \alpha_\mu y_\mu = 0$

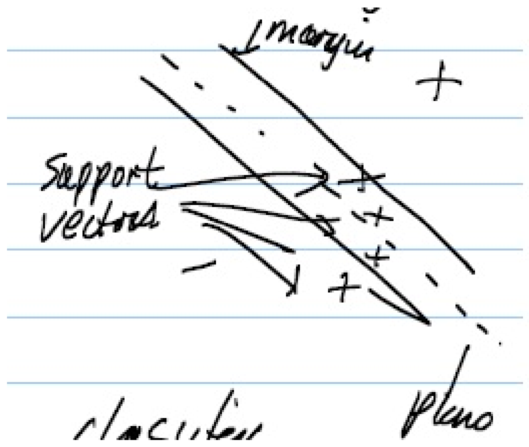

Figure 5: The classifier depends only on the support vectors.

There are standard packages to solve this. (Although they get slow if you have a very large amount of data). Knowing $\{\hat{\alpha}_\mu\}$, will give us the solution $\hat{\vec{\alpha}} = \sum_\mu \hat{\alpha}_\mu y_\mu \vec{x}_\mu$, (only a little more work needed to get $\hat{b}$).

Now we show how to obtain the dual formulation from the primal. The method we use is only correct if the primal is a convex function (but it is a positive quadratic function with linear constraints, which is convex).

Start with the dual formulation L_p. Rewrite it as
$L_p = -\frac{1}{2}\vec{a}\cdot\vec{a} + \sum_\mu \alpha_\mu + \vec{a}\cdot(\vec{a} - \sum_\mu \alpha_\mu y_\mu \vec{x}_\mu) + \sum_\mu z_\mu(\gamma - \tau_\mu - \alpha_\mu) - b\sum_\mu \alpha_\mu y_\mu$.

Extremize w.r.t. $\vec{a}, b, \{z_\mu\}$. The result is:
$\hat{\vec{a}} = \sum_\mu \alpha_\mu y_\mu \vec{x}_\mu, \sum_\mu \alpha_\mu y_\mu = 0, \gamma - \tau_\mu - \alpha_\mu = 0$
Substituting back into L_p gives:
$L_p = -\frac{1}{2}\sum_{\mu,\nu} \alpha_\mu \alpha_\nu y_\mu y_\nu \vec{x}_\mu \vec{x}_\nu + \sum_\mu \alpha_\mu$,
which has to be maximized w.r.t. $\{\alpha_\mu\}$.

2.5 Reformulation of the Perceptron

The Perceptron can be reformulated in the following way. By the theory, the weight hypothesis will always be of form: $\vec{a} = \sum_\mu \alpha_\mu y_\mu \vec{x}_\mu$.

The Perceptron update rule is: If data $\vec{x}_\mu$ is misclassified (i.e, $y_\mu(\vec{a}\cdot\vec{x}_\mu + b) \le 0$), then set
$\vec{\alpha}_\mu \to \vec{\alpha}_\mu + 1$
$b \to b + y_\mu K^2$,
where K is the radius of the smallest ball containing the data.

2.6 Max-Margin from Empirical Risk

Suppose we look at the primal function L_p. Consider the constraint $y_\mu(\vec{x}_\mu \cdot \vec{a} + b) - 1 > 0$. If this constraint is satisfied, then it is best to set the slack variable $z_\mu = 0$ (because otherwise we pay a penalty γ for it). If the constraint is not satisfied, then we set the slack variable to be $z_\mu = 1 - y_\mu(\vec{x}_\mu \cdot \vec{a})$ because this is the smallest value of the slack variable which satisfies the constraint. We can summarize this by paying a *Hinge Loss* penalty $\max\{0, 1 - y_\mu(\vec{x}_\mu \cdot \vec{a} + b)\}$ – if the constraint is satisfied, then the maximum is 0 but, if not, the maximum is $1 - y_\mu(\vec{x}_\mu \cdot \vec{a})$ which is minimum value of the slack variable (to make the constraint satisfied).

This gives an energy function:
$L(\vec{a}, b) = \frac{1}{2}\vec{a} \cdot \vec{a} + \gamma \sum_\mu \max\{0, 1 - y_\mu(\vec{x}_\mu \cdot \vec{a} + b)\}$.
Note that the hinge loss is a convex loss function, see figure (**??**).

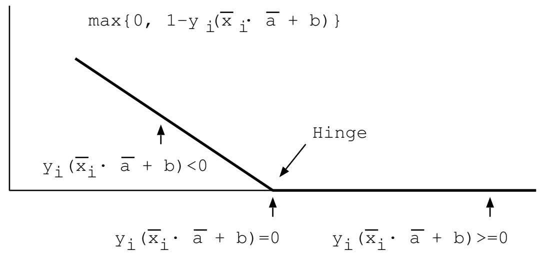

Figure 6: The Hinge loss function.

So, we can re-express the max-margin criterion as the sum of the empirical risk (with hinge loss function) plus a term $\frac{1}{2}|\vec{a}|^2$, multiplied by a constant $\frac{1}{\gamma N}$:
$\frac{L_p}{\gamma N} = \frac{1}{2\gamma N}|\vec{a}|^2 + \frac{1}{N}\sum_{\mu=1}^{N} \max\{0, 1 - y_\mu(\vec{x}_\mu \cdot \vec{a} + b)\}$.
The first term is a *regularizer*. It penalizes decision rules $\hat{y}(\vec{x}) = \text{sign}(\vec{x} \cdot \vec{a} + b)$ which have large $|\vec{a}|$. This is done in order to help generalization. If we only tried to minimize the loss function, we may overfit the data, because the space of possible decision rules is very big (all values of $\vec{a}$ and b). If we penalize those rules with big $|\vec{a}|$, then we restrict our set of rules and are more likely to generalize to new data.

2.7 Online Learning

We can do online learning (update with new data) using the cost function
$L(\vec{a}, b) = \frac{1}{2}\vec{a} \cdot \vec{a} + \gamma \sum_\mu \max\{0, 1 - y_\mu(\vec{x}_\mu \cdot \vec{a} + b)\}$.
Consider the second term for one datapoint $\vec{x}_\mu, y_\mu$:
$\frac{\partial}{\partial \vec{a}} \max\{0, 1 - y_\mu(\vec{x}_\mu \cdot \vec{a} + b)\} = -y_\mu \vec{x}_\mu, \quad \text{if} \quad y_\mu(\vec{x}_\mu \cdot \vec{a} + b) < 1,$
$= 0$ otherwise.

The online learning consists of:
Selecting the data $(\vec{x}_\mu, y_\mu)$ at random.
Computing $\frac{\partial}{\partial \vec{a}} \max\{0, 1 - y_\mu(\vec{x}_\mu \cdot \vec{a} + b)\}$.
If $_\mu(\vec{x}_\mu \cdot \vec{a} + b) < 1$, updating $\vec{a}^t \to \vec{a}^t - \frac{1}{2N}\vec{a}^t - \gamma\{-y_\mu \vec{x}_\mu\}$,
or if $_\mu(\vec{x}_\mu \cdot \vec{a} + b) > 1$, updating $\vec{a}^t \to \vec{a}^t - \frac{1}{2N}\vec{a}^t - \gamma\{0\}$.

This is almost exactly the 1950's perceptron algorithm. The $-y_\mu$ term is like converting negative examples to positive ones. The $\frac{1}{2N}\vec{a}^t$ is from the regularizer.