Machine Learning for Computational Advertising
L3: Linear Models

Alexander J. Smola

Yahoo! Labs
Santa Clara, CA 95051
alex@smola.org

UC Santa Cruz, April 2009
Overview

L1: Machine learning and probability theory
   Introduction to pattern recognition, classification, regression, novelty detection, probability theory, Bayes rule, density estimation

L2: Instance based learning
   Nearest Neighbor, Kernels density estimation, Watson Nadaraya estimator, crossvalidation

L3: Linear models
   Hebb’s rule, perceptron algorithm, regression, classification, feature maps
L3 Linear Models

Hebb’s rule
- positive feedback
- perceptron convergence rule

Hyperplanes
- Linear separability
- Inseparable sets

Features
- Explicit feature construction
- Implicit features via kernels

Kernels
- Examples
- Kernel perceptron
Biology and Learning

Basic Idea

- Good behavior should be rewarded, bad behavior punished (or not rewarded). This improves the fitness of the system.
- Example: hitting a tiger should be rewarded . . .
- Correlated events should be combined.
- Example: Pavlov’s salivating dog.

Training Mechanisms

- Behavioral modification of individuals (learning): Successful behavior is rewarded (e.g. food).
- Hard-coded behavior in the genes (instinct): The wrongly coded animal dies.
Perceptron

\[ f(x) = w_1 x_1 + \ldots + w_6 x_6 \]
Perceptrons

Weighted combination
- Output of the perceptron is a linear combination of the inputs.
- Rescale output (e.g. by sigmoid or threshold function)

Decision Function
- Results are combined into

\[
\sigma \left( \sum_{i=1}^{n} w_i x_i + b \right) = \sigma(\langle w, x \rangle + b).
\]
Linear Function Classes

Expansion

\[ f(x) = \langle w, x \rangle + b \text{ where } w, x \in \mathbb{R}^m \text{ and } b \in \mathbb{R}. \]

Applications

- Spam filtering (e-mail)
- Echo cancellation (old analog overseas cables)
- Click probability
- Bid value for advanced match
- Machine learning ranking
- Collaborative filtering

Learning

Weights are “plastic” — adapted via the training data.
Linear Separation

\[ f(x) = \langle w, x \rangle + b \]
Perceptron Algorithm

argument: \( X := \{x_1, \ldots, x_m\} \subset \mathcal{X} \) (data)
\( Y := \{y_1, \ldots, y_m\} \subset \{\pm 1\} \) (labels)

function \((w, b) = \text{Perceptron}(X, Y)\)
initialize \(w, b = 0\)
repeat
Pick \((x_i, y_i)\) from data
if \(y_i(w \cdot x_i + b) \leq 0\) then
\(w' = w + y_ix_i\)
\(b' = b + y_i\)
until \(y_i(w \cdot x_i + b) > 0\) for all \(i\)
end
Interpretation

Algorithm
- Do nothing if we classify \((x_i, y_i)\) correctly
- For incorrectly classified observation update \((w, b) + = (y_i x_i, y_i)\).
- Positive reinforcement of observations.

Solution
- Weight vector is linear combination of observations \(x_i\):
  \[
  w \leftarrow w + y_i x_i
  \]
- Classification can be written in terms of dot products:
  \[
  w \cdot x + b = \sum_{j \in E} y_j x_j \cdot x + b
  \]}
Theoretical Analysis

Incremental Algorithm

Already while the perceptron is learning, we can use it.

**Convergence Theorem (Rosenblatt and Novikoff)**

Suppose that there exists a $\rho > 0$, a weight vector $w^*$ satisfying $\|w^*\| = 1$, and a threshold $b^*$ such that

$$y_i (\langle w^*, x_i \rangle + b^*) \geq \rho \quad \text{for all} \quad 1 \leq i \leq m.$$ 

Then the hypothesis maintained by the perceptron algorithm converges to a linear separator after no more than

$$\frac{(b^*^2 + 1)(R^2 + 1)}{\rho^2}$$

updates, where $R = \max_i \|x_i\|$. 
Starting Point

We start from $w_1 = 0$ and $b_1 = 0$.

Step 1: Bound on the increase of alignment

Denote by $w_i$ the value of $w$ at step $i$ (analogously $b_i$).

Alignment: $\langle (w_i, b_i), (w^*, b^*) \rangle$

For error in observation $(x_i, y_i)$ we get

$$\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle$$

$$= \langle [(w_j, b_j) + y_i(x_i, 1)], (w^*, b^*) \rangle$$

$$= \langle (w_j, b_j), (w^*, b^*) \rangle + y_i \langle (x_i, 1) \cdot (w^*, b^*) \rangle$$

$$\geq \langle (w_j, b_j), (w^*, b^*) \rangle + \rho$$

$$\geq j\rho.$$
Proof, Part II

Step 2: Cauchy-Schwartz for the Dot Product

\[
\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle \leq \| (w_{j+1}, b_{j+1}) \| \cdot \| (w^*, b^*) \| = \sqrt{1 + (b^*)^2 \| (w_{j+1}, b_{j+1}) \|}
\]

Step 3: Upper Bound on \( \| (w_j, b_j) \| \)

If we make a mistake we have

\[
\| (w_{j+1}, b_{j+1}) \|^2 = \| (w_j, b_j) + y_i(x_i, 1) \|^2 = \| (w_j, b_j) \|^2 + 2y_i \langle (x_i, 1), (w_j, b_j) \rangle + \| (x_i, 1) \|^2 \leq \| (w_j, b_j) \|^2 + \| (x_i, 1) \|^2 \leq \rho \cdot (R^2 + 1).
\]

Step 4: Combination of first three steps

\[
\rho \leq \sqrt{1 + (b^*)^2 \| (w_{j+1}, b_{j+1}) \|} \leq \sqrt{\rho (R^2 + 1)((b^*)^2 + 1)}
\]

Solving for \( \rho \) proves the theorem.
Solutions of the Perceptron
Learning Algorithm
We perform an update only if we make a mistake.

Convergence Bound
- Bounds the maximum number of mistakes in total. We will make at most \((b^* + 1)(R^1 + 1)/\rho^2\) mistakes in the case where a “correct” solution \(w^*, b^*\) exists.
- This also bounds the expected error (if we know \(\rho, R, \) and \(|b^*|\)).

Dimension Independent
Bound does not depend on the dimensionality of \(X\).

Sample Expansion
We obtain \(w\) as a linear combination of \(x_i\).
Mini Summary

Perceptron
- Separating halfspaces
- Perceptron algorithm
- Convergence theorem
- Only depends on margin, dimension independent

Pseudocode

```python
for i in range(m):
    ytest = numpy.dot(w, x[:,i]) + b
    if ytest * y[i] <= 0:
        w += y[i] * x[:,i]
        b += y[i]
```
Nonlinearity via Preprocessing

Problem
Linear functions are often too simple to provide good estimators.

Idea
- Map to a higher dimensional feature space via $\Phi : x \rightarrow \Phi(x)$ and solve the problem there.
- Replace every $\langle x, x' \rangle$ by $\langle \Phi(x), \Phi(x') \rangle$ in the perceptron algorithm.

Consequence
- We have nonlinear classifiers.
- Solution lies in the choice of features $\Phi(x)$. 
Features
Quadratic features correspond to circles, hyperbolas and ellipsoids as separating surfaces.
Constructing Features

Idea

Construct features manually. E.g. for OCR we could use

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loops</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3 Joints</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4 Joints</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Angles</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ink</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
More Examples

Two Interlocking Spirals

If we transform the data \((x_1, x_2)\) into a radial part \((r = \sqrt{x_1^2 + x_2^2})\) and an angular part \((x_1 = r \cos \phi, x_1 = r \sin \phi)\), the problem becomes much easier to solve (we only have to distinguish different stripes).

Japanese Character Recognition

Break down the images into strokes and recognize it from the latter (there’s a predefined order of them).

Medical Diagnosis

Include physician’s comments, knowledge about unhealthy combinations, features in EEG, . . .

Suitable Rescaling

If we observe, say the weight and the height of a person, rescale to zero mean and unit variance.
Perceptron on Features

argument: $X := \{x_1, \ldots, x_m\} \subset \mathcal{X}$ (data)
$Y := \{y_1, \ldots, y_m\} \subset \{\pm 1\}$ (labels)

function $(w, b) = \text{Perceptron}(X, Y, \eta)$

initialize $w, b = 0$

repeat
Pick $(x_i, y_i)$ from data
if $y_i(w \cdot \Phi(x_i) + b) \leq 0$ then
  $w' = w + y_i\Phi(x_i)$
  $b' = b + y_i$
until $y_i(w \cdot \Phi(x_i) + b) > 0$ for all $i$
end

Important detail

$w = \sum_j y_j\Phi(x_j)$ and hence $f(x) = \sum_j y_j(\Phi(x_j) \cdot \Phi(x)) + b$
Problems with Constructing Features

Problems

- Need to be an expert in the domain (e.g. OCR).
- Features may not be robust (e.g. postman drops letter).
- Can be expensive to compute.

Solution

- Use shotgun approach.
- Compute many features and hope . . .
- Do this efficiently.
Polynomial Features

Quadratic Features in $\mathbb{R}^2$

$$\Phi(x) := \left( x_1^2, \sqrt{2}x_1x_2, x_2^2 \right)$$

Dot Product

$$\langle \Phi(x), \Phi(x') \rangle = \left\langle \left( x_1^2, \sqrt{2}x_1x_2, x_2^2 \right), \left( x_1'^2, \sqrt{2}x_1'x_2', x_2'^2 \right) \right\rangle$$

$$= \langle x, x' \rangle^2.$$ 

Insight

Trick works for any polynomials of order $d$ via $\langle x, x' \rangle^d$. 

![Color plots](image)
Kernels

Problem
- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to 5005 numbers. For higher order polynomial features much worse.

Solution
Don’t compute the features, try to compute dot products implicitly. For some features this works . . .

Definition
A kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a symmetric function in its arguments for which the following property holds

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

for some feature map $\Phi$. If $k(x, x')$ is much cheaper to compute than $\Phi(x)$ . . .
Polynomial Kernels in $\mathbb{R}^n$

Idea

- We want to extend $k(x, x') = \langle x, x' \rangle^2$ to

$$k(x, x') = (\langle x, x' \rangle + c)^d$$ where $c \geq 0$ and $d \in \mathbb{N}$.

- Prove that such a kernel corresponds to a dot product.

Proof strategy

Simple and straightforward: compute the explicit sum given by the kernel, i.e.

$$k(x, x') = (\langle x, x' \rangle + c)^d = \sum_{i=0}^{m} \binom{d}{i} (\langle x, x' \rangle)^i c^{d-i}$$

Individual terms $(\langle x, x' \rangle)^i$ are dot products for some $\Phi_i(x')$.

Kernel Perceptron

argument: \( X := \{x_1, \ldots, x_m\} \subset \mathcal{X} \) (data)
\( Y := \{y_1, \ldots, y_m\} \subset \{\pm 1\} \) (labels)

function \( f = \text{Perceptron}(X, Y, \eta) \)

initialize \( f = 0 \)
repeat
    Pick \((x_i, y_i)\) from data
    if \( y_i f(x_i) \leq 0 \) then
        \( f(\cdot) \leftarrow f(\cdot) + y_i k(x_i, \cdot) + y_i \)
    until \( y_i f(x_i) > 0 \) for all \( i \)
end

Important detail

\[ w = \sum_j y_j \Phi(x_j) \] and hence \( f(x) = \sum_j y_j k(x_j, x) + b. \)
Are all $k(x, x')$ good Kernels?

**Computability**
We have to be able to compute $k(x, x')$ efficiently (much cheaper than dot products themselves).

**“Nice and Useful” Functions**
The features themselves have to be useful for the learning problem at hand. Quite often this means smooth functions.

**Symmetry**
Obviously $k(x, x') = k(x', x)$ due to the symmetry of the dot product $\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$.

**Dot Product in Feature Space**
Is there always a $\Phi$ such that $k$ really is a dot product?

**Mercer’s theorem**
$k$ needs to correspond to a positive integral operator . . .
Some Good Kernels

Examples of kernels $k(x, x')$

- Linear: $\langle x, x' \rangle$
- Laplacian RBF: $\exp\left(-\lambda \| x - x' \|\right)$
- Gaussian RBF: $\exp\left(-\lambda \| x - x' \|^2\right)$
- Polynomial: $(\langle x, x' \rangle + c)^d, \ c \geq 0, \ d \in \mathbb{N}$
- B-Spline: $B_{2n+1}(x - x')$
- Cond. Expectation: $E_c[p(x|c)p(x'|c)]$

Simple trick for checking Mercer’s condition
Compute the Fourier transform of the kernel and check that it is nonnegative.
Linear Kernel

\[ k(x, y) \text{ for } x = 1 \]

The graph shows a linear kernel where the value of the kernel function increases linearly with the distance in the feature space.
Laplacian Kernel

\[ k(x,y) \text{ for } x=1 \]
Gaussian Kernel

![Gaussian Kernel Graph](image)
Polynomial (Order 3)
$B_3$-Spline Kernel
Mini Summary

Features

- Prior knowledge, expert knowledge
- Shotgun approach (polynomial features)
- Kernel trick $k(x, x') = \langle \phi(x), \phi(x') \rangle$
- Mercer's theorem

Applications

- Kernel Perceptron
- Nonlinear algorithm automatically by query-replace

Examples of Kernels

- Gaussian RBF
- Polynomial kernels
Risk Minimization

General Problem
Find $f(x) = \langle w, x \rangle + b$ such that loss $l(y, f(x))$ is minimized.

Learning as optimization
Minimize the average risk on training set via

$$\min_{(w,b)} \sum_{i=1}^{m} l(y_i, \langle w, x_i \rangle + b) + \frac{\lambda}{2} \|w\|^2$$

Here $\frac{\lambda}{2} \|w\|^2$ is a regularizer penalizing how steep the function $f$ is.
Applications

Regression — Least Mean Squares
- Bid estimation: $y$ is bid, $x = (\text{keyword, query})$
- Collaborative filtering: $y$ is rating, $x = (\text{product, user})$

$$l(y, f(x)) = \frac{1}{2}(y - f(x))^2$$

Classification — Logistic Regression
- Click probability estimation: $y$ is click/no click, $x$ is ad
- Spam filtering: $y$ is spam/no spam, $x$ is webpage

$$l(y, f(x)) = -\log p(y|x) = \log(1 + e^{-yf(x)})$$

equivalently $p(y|x) = \frac{1}{1 + e^{-yf(x)}}$
Applications

Classification — Hinge loss

\[ l(y, f(x)) = \max(0, -yf(x)) \]

This will give us the Perceptron algorithm

Classification — Soft margin loss

\[ l(y, f(x)) = \max(0, 1 - yf(x)) \]

This will give us the Support Vector Machines loss. We want to ensure that we classify with confidence: loss only vanishes if \( yf(x) \geq 1 \).

Regression — Absolute Value Loss

Want to penalize absolute deviation from observation

\[ l(y, f(x)) = |y - f(x)| \]
Stochastic Gradient Descent

argument: \( X := \{ x_1, \ldots, x_m \} \subset \mathcal{X} \) (data)
\( Y := \{ y_1, \ldots, y_m \} \subset \mathcal{Y} \) (labels)

function \( (w, b) = \text{StochasticGradientDescent}(X, Y) \)
initialize \( w, b = 0 \)
initialize counter \( n = n_0 \), scale \( \eta_0 \)
repeat
  Pick \((x, y)\) from data
  Compute prediction \( f(x_i) \)
  Compute learning rate \( \eta = \frac{\eta_0}{\sqrt{n}} \)
  \[
  (w, b) = (1 - \lambda \eta)(w, b) - \eta(x, 1) \cdot \partial_{f(x)} l(y, f(x_i))
  \]
  Increment \( n = n + 1 \)
until all data read
end
Key Theorem

Convergence
Under general conditions stochastic gradient descent converges at rate $O(n^{-\frac{1}{2}})$ to the optimal solution.

Practical hack
Set regularization $\lambda = 0$ if we have lots of data.
Summary

Hebb’s rule
- positive feedback
- perceptron convergence rule, kernel perceptron

Features
- Explicit feature construction
- Implicit features via kernels

Kernels
- Examples
- Mercer’s theorem

Risk Minimization
- Stochastic gradient descent algorithm
- Simple to implement
- Fast convergence