Maximizing the Margin

Pascal Fua
IC-CVLab
Logistic Regression is Better than the Perceptron

But ....
Outliers Can Cause Problems

- Logistic regression tries to minimize the error-rate at training time.
- Can result in poor classification rates at test time.

—> Sometimes, we should accept to misclassify a few training samples.
The orthogonal distance between the decision boundary and the nearest sample is called the **margin**.
Maximizing the Margin

- The larger the margin, the better!
- The logistic regression does not guarantee the largest.

How do we maximize it?
Reminder: Signed Distance

\[ \mathbf{x} = [1, x_1, \ldots, x_N] \]

\( h=0 \): Point is on the decision boundary.
\( h>0 \): Point on one side.
\( h<0 \): Point on the other side.

\[ \tilde{\mathbf{w}} = [w_0, w_1, \ldots, w_n] \]

\[ \tilde{\mathbf{w}} = [w_0 | \mathbf{w}] \text{ with } \sum_{i=1}^{N} w_i^2 = 1 \]

Hyperplane: \( \mathbf{x} \in \mathbb{R}^N, \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} = 0 \), with \( \tilde{\mathbf{x}} = [1 | \mathbf{x}] \).

Signed distance: \( \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} \), with \( \tilde{\mathbf{w}} = [w_0 | \mathbf{w}] \) and \( ||\mathbf{w}|| = 1 \).
Binary Classification in N Dimensions

Hyperplane: $x \in \mathbb{R}^N$, $\tilde{w} \cdot \tilde{x} = 0$, with $\tilde{x} = [1 \mid x]$.

Signed distance: $\tilde{w} \cdot \tilde{x}$, with $\tilde{w} = [w_0 \mid w]$ and $\|w\| = 1$.

Problem statement: Find $\tilde{w}$ such that
- for all or most positive samples $\tilde{w} \cdot \tilde{x} > 0$,
- for all or most negative samples $\tilde{w} \cdot \tilde{x} < 0$. 
Reformulating the Signed Distance Again

\[ w = [w_1, \ldots, w_N] \]

\( h = 0 \): Point is on the decision boundary.
\( h > 0 \): Point on one side.
\( h < 0 \): Point on the other side.

\[ \tilde{x} = [1, x_1, \ldots, x_N] \]
\[ \tilde{w} = [w_0, w_1, \ldots, w_N] \text{ with } \sum_{i=1}^{N} w_i^2 = 1 \]

Hyperplane: \( x \in \mathbb{R}^N, \tilde{w} \cdot \tilde{x} = 0, \text{ with } \tilde{x} = [1 \mid x] \).

Signed distance: \( \tilde{w} \cdot \tilde{x} \), with \( \tilde{w} = [1 \mid w] \) and \( ||w|| = 1 \).
Reformulated Signed Distance

\[ \mathbf{\tilde{x}} = [1, x_1, \ldots, x_N] \]

\[ \mathbf{w} = [w_1, \ldots, w_n] \]

\( h = 0 \): Point is on the decision boundary.
\( h > 0 \): Point on one side.
\( h < 0 \): Point on the other side.

\[ \mathbf{\tilde{w}} = [w_0 | \mathbf{w}] \in \mathbb{R}^{N+1} \]

\[ \mathbf{\tilde{w}}' = \frac{\mathbf{\tilde{w}}}{||\mathbf{w}||} = \left[ \frac{w_0}{||\mathbf{w}||} | \mathbf{\tilde{w}} \right] \]

Hyperplane: \( \mathbf{x} \in \mathbb{R}^N, \mathbf{\tilde{w}} \cdot \mathbf{\tilde{x}} = 0 \), with \( \mathbf{\tilde{x}} = [1 | \mathbf{x}] \).

Signed distance: \( \mathbf{\tilde{w}}' \cdot \mathbf{\tilde{x}} = \frac{\mathbf{\tilde{w}} \cdot \mathbf{\tilde{x}}}{||\mathbf{w}||}, \forall \mathbf{\tilde{w}} \in \mathbb{R}^{N+1} \).
Maximum Margin Classifier

- Given a training set \( \{(\mathbf{x}_n, t_n)\}_{1 \leq n \leq N} \) with \( t_n \in \{-1, 1\} \) and solution such that all the points are correctly classified, we have
  \[
  \forall n, \quad t_n (\tilde{\mathbf{w}}_n \cdot \tilde{\mathbf{x}}_n) \geq 0 .
  \]

- We can write the unsigned distance to the decision boundary as
  \[
  d_n = t_n \frac{(\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}_n)}{||\mathbf{w}||}
  \]

  \( \rightarrow \) A maximum margin classifier aims to maximize this distance for the point closest to the boundary, that is, maximize the minimum such distance.

  \[
  \tilde{\mathbf{w}}^* = \arg\max_{\tilde{\mathbf{w}}} \min_n \left( \frac{t_n \cdot (\tilde{\mathbf{w}} \cdot \mathbf{x}_n)}{||\mathbf{w}||} \right)
  \]
Maximum Margin Classifier

\[ \tilde{w}^* = \arg\max_{\tilde{w}} \min_n \left( \frac{t_n \cdot (\tilde{w} \cdot x_n)}{\| w \|} \right) \]

• Unfortunately, this is a difficult optimization problem to solve.
• We will convert it into an equivalent, but easier to solve, problem.
The signed distance is invariant to a scaling of $\tilde{w}$:

$$\tilde{w} \rightarrow \lambda \tilde{w} : d_n = t_n \frac{(\lambda \tilde{w} \cdot \tilde{x}_n)}{||\lambda w||} = t_n \frac{(\tilde{w} \cdot \tilde{x}_n)}{||w||}.$$  

We can choose $\lambda$ so that for the point $m$ closest to the boundary, we have

$$t_m \cdot (\tilde{w} \cdot x_m) = 1.$$  

For all points we therefore have

$$t_n \cdot (\tilde{w} \cdot x_n) \geq 1,$$

and the equality holds for at least one point.
Linear Support Vector Machine

\[ \forall n, \quad t_n(\tilde{w} \cdot x_n) \geq 1 \]

\[ \exists n \quad t_n(\tilde{w} \cdot x_n) = 1 \]

\[ \Rightarrow \min_n d_n = \min_n \frac{t_n(\tilde{w} \cdot x_n)}{||w||} = \frac{1}{||w||} \]

• To maximize the margin, we only need to maximize \( 1/||w|| \).

• This is equivalent to minimizing \( \frac{1}{2} ||w||^2 \).

• We can find max margin classifier as

\[ w^* = \arg\min_w \frac{1}{2} ||w||^2 \text{ subject to } \forall n, \quad t_n \cdot (\tilde{w} \cdot x_n) \geq 1 \]

• This is a quadratic program, which is a convex problem.

\[ \Rightarrow \text{ It can be solved to optimality.} \]
LR vs Linear SVM

- The LR decision boundary can come close to some of the training examples.
- The SVM tries to prevent that.
From Perceptron and LR to Linear SVM

Perceptron

Logistic Regression

Are we done yet?

No!

Linear SVM
Maximum Margin Classifier

- Given a training set \( \{(x_n, t_n)_{1 \leq n \leq N}\} \) with \( t_n \in \{-1, 1\} \) and solution such that all the points are correctly classified, we have

\[
\forall n, \quad t_n(\tilde{w} \cdot \tilde{x}_n) \geq 1.
\]

- We can write the **unsigned** distance to the decision boundary as

\[
d_n = t_n \frac{(\tilde{w} \cdot \tilde{x}_n)}{||w||},
\]

\( \tilde{w}^* = \arg\max_{\tilde{w}} \min_n \left( \frac{t_n \cdot (\tilde{w} \cdot x_n)}{||w||} \right) \)

Rarely achievable in practice.
Overlapping Classes

The data rarely looks like this. It generally looks like that.

→→ Must account for the fact that not all training samples can be correctly classified!
Relaxing the Constraints

• The original problem

\[ w^* = \arg\min_w \frac{1}{2} ||w||^2 \text{ subject to } \forall n, \quad t_n \cdot (\tilde{w} \cdot x_n) \geq 1, \]

cannot be satisfied.

• We must allow some of the constraints to violated, but as few as possible.
Slack Variables

- We introduce an additional slack variable $\xi_n$ for each sample.
- We rewrite the constraints as $t_n \cdot (\tilde{w} \cdot x_n) \geq 1 - \xi_n$.
- $\xi_i \geq 0$ weakens the original constraints.

If $0 < \xi_n \leq 1$, sample $n$ lies inside the margin, but is still correctly classified.

If $\xi_n \geq 1$, then sample $i$ is misclassified.
Naive Formulation

\[ w^* = \text{argmin}_w \frac{1}{2} ||w||^2 \]

subject to \( \forall n, \quad t_n \cdot (\tilde{w} \cdot x_n) \geq 1 - \xi_n \text{ and } \xi_n \geq 0 \)

- This would simply allow the model to violate all the original constraints at no cost.
- This would result in a useless classifier.
**Improved Formulation**

\[ w^* = \arg\min_{(w, \{\xi_n\})} \frac{1}{2} ||w||^2 + C \sum_{n=1}^{N} \xi_n, \]

subject to \( \forall n, \quad t_n \cdot (\tilde{w} \cdot x_n) \geq 1 - \xi_n \) and \( \xi_n \geq 0. \)

- \( C \) is constant that controls how costly constraint violations are.
- The problem is still convex.

Large margin but potential misclassifications.

Smaller margin but fewer misclassifications.
Choosing the C Parameter

C=1:
• Large margin.
• Many training samples misclassified.

C=100:
• Small margin.
• Few training samples misclassified.

Which is best?
• It depends.
• Must use cross-validation, as we did for k-Means.
Linear SVM Trade Off

- The points can be linearly separated but the margin is still very small.
- At test time the two circles will be misclassified.

- The margin is much larger but one training example is misclassified.
- At test time the two circles will be classified correctly.

→ Tradeoff between the number of mistakes on the training data and the margin.
Support Vector Machines

- Logistic Regression: 63.5%
- Decision Trees: 49.9%
- Random Forests: 46.3%
- Neural Networks: 37.6%
- Bayesian Techniques: 30.6%
- Ensemble Methods: 28.5%
- SVMs: 26.7%
- Gradient Boosted Machines: 23.9%
- CNNs: 18.9%
- RNNs: 12.3%
- Other: 8.3%
- Evolutionary Approaches: 5.5%
- HMMs: 5.4%
- Markov Logic Networks: 4.9%
- GANs: 2.8%