Perceptrons, SVMs, Neural Networks
ECE 448/ CS 440

Ishan Deshpande

Nov 9
Outline

Supervised Classification

Perceptrons
- Linear Separability
- Training Algorithm
- Multi-class classification

Support Vector Machines
- Picking the best boundary
- Beyond linear boundaries - the Kernel Trick

Neural Networks
- Hidden layers
Learn to *tell apart*

- Given a set of tuples \( \{X_i, Y_i\} \), learn a function \( f \) which tells us \( Y_i \) for a given \( X_i \).
Learn to *tell apart*

- Given a set of tuples \( \{X_i, Y_i\} \), learn a function \( f \) which tells us \( Y_i \) for a given \( X_i \).
- \( X_i \) is the *feature vector*, \( Y_i \) is the *label*, \( f \) is the *classifier*. 
Learn to *tell apart*

- Given a set of tuples \( \{X_i, Y_i\} \), learn a function \( f \) which tells us \( Y_i \) for a given \( X_i \).
- \( X_i \) is the *feature vector*, \( Y_i \) is the *label*, \( f \) is the *classifier*.
- e.g. \( X = \) (vectorized) pixel intensity, \( Y = \) image type
Outline

Supervised Classification

Perceptrons
- Linear Separability
- Training Algorithm
- Multi-class classification

Support Vector Machines
- Picking the best boundary
- Beyond linear boundaries - the Kernel Trick

Neural Networks
- Hidden layers
What’s the classifier
What’s the classifier

- Simply check which side are we on.
What’s the classifier

- Simply check which side we are on.
- Predict $\text{sgn}(w^T x)$, where $w$ is the **normal** to the boundary.
What's a perceptron

\[ x_1 \rightarrow w_1 \rightarrow \text{sgn}(\cdot) \rightarrow y \]

\[ x_2 \rightarrow w_2 \rightarrow \text{sgn}(\cdot) \rightarrow y \]

\[ x_3 \rightarrow w_3 \rightarrow \text{sgn}(\cdot) \rightarrow y \]
Outline

Supervised Classification

Perceptrons
  Linear Separability
  Training Algorithm
  Multi-class classification

Support Vector Machines
  Picking the best boundary
  Beyond linear boundaries - the Kernel Trick

Neural Networks
  Hidden layers
Finding a classifier

- Start with a random guess.
Finding a classifier

- Cycle through the training set. Check prediction $y'$ vs actual label $y$. 
Finding a classifier

- Update line with the rule:

\[ w = w + \alpha (y - y')x \]  \hspace{1cm} (1)
Finding a classifier

Suppose \( \mathbf{w} = (1, -1) \)
Finding a classifier

Consider \((1, 2)\) with \(y = 1\). For this \(y' = -1\). For \(\alpha = 1\) the new \(\mathbf{w}\) is

\[
\mathbf{w} = (1, -1) + 1 \times (1 - (-1)) \times (1, 2) \tag{2}
\]

\[
\mathbf{w} = (3, 3) \tag{3}
\]
Finding a classifier

The boundary is now:

\[ x_1 \]

\[ x_2 \]
Finding a classifier

Consider \((-1, 0.5)\) with \(y = 1\). For this \(y' = -1\). We update again:

\[
\mathbf{w} = (3, 3) + 1 \times (1 - (-1)) \times (-1, 0.5) \quad (4)
\]

\[
\mathbf{w} = (1, 4) \quad (5)
\]
Finding a classifier

The boundary is now:
Finding a classifier

- If the data is indeed linearly separable, it will eventually converge!
Finding a classifier

- If the data is indeed linearly separable, it will eventually converge!
- If the data is NOT linearly separable, training can diverge for a fixed $\alpha$. 

> $\text{If the data is indeed linearly separable, it will eventually converge!}$

> $\text{If the data is NOT linearly separable, training can diverge for a fixed } \alpha.$

\[ y = \text{sgn}(w^T x + b) \] 

\[ \tilde{x} = \begin{cases} x, & \alpha = 1 \\ \{w, b\} & \text{otherwise} \end{cases} \] 

\[ \tilde{w} = \begin{cases} w, & \alpha = 1 \\ \{w, b\} & \text{otherwise} \end{cases} \]
Finding a classifier

- If the data is indeed linearly separable, it will eventually converge!
- If the data is NOT linearly separable, training can diverge for a fixed $\alpha$.
- Solution: Use $\alpha = \frac{1}{t}$ - finds the best separator if data is actually separable, otherwise finds the MMSE solution.
Finding a classifier

- If the data is indeed linearly separable, it will eventually converge!
- If the data is NOT linearly separable, training can diverge for a fixed $\alpha$.
- Solution: Use $\alpha = \frac{1}{t}$ - finds the best separator if data is actually separable, otherwise finds the MMSE solution.
- Include a bias term, i.e. offset of line.

$$y = sgn(w^T x + b)$$  \hspace{1cm} (6)
Finding a classifier

- If the data is indeed linearly separable, it will eventually converge!
- If the data is NOT linearly separable, training can diverge for a fixed $\alpha$.
- Solution: Use $\alpha = \frac{1}{t}$ - finds the best separator if data is actually separable, otherwise finds the MMSE solution.
- Include a bias term, i.e. offset of line.

$$y = \text{sgn}(w^T x + b)$$  \hspace{1cm} (6)

- Use the same training algorithm with $\tilde{x}$ and $\tilde{w}$ as

$$\tilde{x} = \{x, 1\}, \quad \tilde{w} = \{w, b\}$$  \hspace{1cm} (7)
Differentiable Variant

- Instead of $\text{sgn}(.)$, use a differentiable non-linear function, such as the sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$. 

$\sigma(x) = \frac{1}{1+e^{-x}}$
Differentiable Variant

- Instead of $\text{sgn}(\cdot)$, use a differentiable non-linear function, such as the sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$.

- Minimize

$$E(w) = \sum_i (y_i - f(x_i))^2 \quad (8)$$
Differentiable Variant

- Instead of $sgn(.)$, use a differentiable non-linear function, such as the sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$.
- Minimize
  \[
  E(w) = \sum_i (y_i - f(x_i))^2 \tag{8}
  \]
- Update via gradient descent
  \[
  w = w - \alpha \frac{d}{dw} E(w) \tag{9}
  \]
Differentiable Variant

- Instead of $\text{sgn}(.)$, use a differentiable non-linear function, such as the sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$.

- Minimize

$$E(w) = \sum_i (y_i - f(x_i))^2 \quad (8)$$

- Update via gradient descent

$$w = w - \alpha \frac{d}{dw} E(w) \quad (9)$$

- For the sigmoid, this is:

$$w = w - \alpha (y - f(x))f(x)(1 - f(x))x \quad (10)$$
Outline

Supervised Classification

Perceptrons
   Linear Separability
   Training Algorithm
   Multi-class classification

Support Vector Machines
   Picking the best boundary
   Beyond linear boundaries - the Kernel Trick

Neural Networks
   Hidden layers
One vs Others

Source: http://cs231n.github.io/linear-classify/
One vs Others

- One classifier for one class.
One vs Others

- One classifier for one class.
- Predict

\[ c = \arg \max_{c'} w_{c'}^T x \] (11)

If \( c \) is misclassified as \( c' \), update using

\[ w_c = w_c + \alpha x \] (12)

\[ w_{c'} = w_{c'} - \alpha x \] (13)
One vs Others

- One classifier for one class.

- Predict
  \[ c = \arg \max_{c'} \mathbf{w}_{c'}^T \mathbf{x} \]  
  \[ (11) \]

- If \( c \) is misclassified as \( c' \), update using
  \[ \mathbf{w}_c = \mathbf{w}_c + \alpha \mathbf{x} \]  
  \[ (12) \]
  \[ \mathbf{w}_{c'} = \mathbf{w}_{c'} - \alpha \mathbf{x} \]  
  \[ (13) \]
Outline

Supervised Classification

Perceptrons
- Linear Separability
- Training Algorithm
- Multi-class classification

Support Vector Machines
- Picking the best boundary
- Beyond linear boundaries - the Kernel Trick

Neural Networks
- Hidden layers
Which boundary is better?
Which boundary is better?

- Intuitively, pick the one that is equally distant from both classes.

A Tutorial on Support Vector Machines for Pattern Recognition
Which boundary is better?

- Perpendicular distance of support vectors from the boundary is:

\[ \frac{|w^T x + b|}{||w||} \]  \hspace{1cm} (14)
Which boundary is better?

- Perpendicular distance of support vectors from the boundary is:

\[
\frac{|\mathbf{w}^T \mathbf{x} + b|}{||\mathbf{w}||}
\]  \hspace{1cm} (14)

- Suppose we require, for all support vectors, that:

\[
|\mathbf{w}^T \mathbf{x} + b| = 1
\]  \hspace{1cm} (15)
Which boundary is better?

- Perpendicular distance of support vectors from the boundary is:

\[
\frac{|\mathbf{w}^T \mathbf{x} + b|}{||\mathbf{w}||} \quad (14)
\]

- Suppose we require, for all support vectors, that:

\[
|\mathbf{w}^T \mathbf{x} + b| = 1 \quad (15)
\]

- The margin is then \(\frac{2}{||\mathbf{w}||}\)
Which boundary is better?

- Formulated as:

\[
\min_{w,b} \quad \frac{1}{2} w^T w
\]

subject to:

\[
y_i(w^T x_i + b) \geq 1
\]
Which boundary is better?

- Formulated as:
  \[
  \min_{w,b} \frac{1}{2} w^T w
  \]  
  subject to:
  \[
  y_i(w^T x_i + b) \geq 1
  \]  
  Eq. 16 ensures maximum margin, while Eq. 17 ensures correct classification.

\[
\begin{align*}
\min_{w,b} & \quad \frac{1}{2} w^T w \\
\text{subject to:} & \quad y_i(w^T x_i + b) \geq 1
\end{align*}
\]
Which boundary is better?

- Formulated as:

\[
\min_{w, b} \quad \frac{1}{2} w^T w
\]  
subject to:

\[
y_i(w^T x_i + b) \geq 1
\]

- Eq. 16 ensures maximum margin, while Eq. 17 ensures correct classification.

- The classifier is of the form:

\[
w = \sum_i \alpha_i y_i x_i
\]

and

\[
y = \sum_i \alpha_i y_i x_i^T x + b
\]

where \(\alpha_i\) are learned weights.
Which boundary is better?

- This can be relaxed if the data is not actually separable

\[
\min_{w,b} \frac{1}{2} w^T w + C \sum_{i} \max(0, 1 - y_i (w^T x_i + b)) \quad (20)
\]
Which boundary is better?

- This can be relaxed if the data is not actually separable

\[
\min_{w, b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i (w^T x_i + b))
\] (20)

- \( C \) allows you to give weight to one over the other.
Which boundary is better?

- This can be relaxed if the data is not actually separable

\[
\min_{w,b} \frac{1}{2}w^Tw + C \sum_i \max(0, 1 - y_i(w^Tx_i + b)) \tag{20}
\]

- \(C\) allows you to give weight to one over the other.

- Here we judge classification accuracy with the ‘hinge’ loss

\[
\max(0, 1 - y_i(w^Tx_i + b)) \tag{21}
\]
Outline

Supervised Classification

Perceptrons
  Linear Separability
  Training Algorithm
  Multi-class classification

Support Vector Machines
  Picking the best boundary
  Beyond linear boundaries - the Kernel Trick

Neural Networks
  Hidden layers
What if the data is not linearly separable?

- Try mapping it to a space where it is! Use a $\phi$ such that

![Input Space to Feature Space Transformation](image)
What if the data is not linearly separable?

- Consider points in concentric circles. Map $x_i$ to $x_i^2$. 
What if the data is not linearly separable?

The kernel trick

- Eq. 19 will be rewritten as:

\[ y = \sum_i \alpha_i y_i \phi(x_i)^T \phi(x) + b \]  

(22)
What if the data is not linearly separable?

The kernel trick

- Eq. 19 will be rewritten as:

\[
y = \sum_{i} \alpha_i y_i \phi(x_i)^T \phi(x) + b
\]

- Instead of explicitly defining \( \phi \), we can also define a \( K(x, x') \) such that

\[
y = \sum_{i} \alpha_i y_i K(x_i, x) + b
\]

Note: \( K \) must satisfy Mercer’s conditions. Examples include polynomial kernels \((1 + x^T x')^d\), Gaussian kernels \( \exp(-\frac{1}{2}(x-x')^T(x-x')) \)
Outline

Supervised Classification

Perceptrons
  Linear Separability
  Training Algorithm
  Multi-class classification

Support Vector Machines
  Picking the best boundary
  Beyond linear boundaries - the Kernel Trick

Neural Networks
  Hidden layers
Stacks of perceptrons can learn non-linear functions. e.g. Consider a simple 1-d scenario.
Combine several perceptron units

\[ x_1 > 2? \]

- Perceptrons
- SVMs, Neural Networks
- Supervised Classification
- Perceptrons
  - Linear Separability
  - Training Algorithm
  - Multi-class classification
- Support Vector Machines
  - Picking the best boundary
  - Beyond linear boundaries - the Kernel Trick
- Neural Networks
  - Hidden layers
Combine several perceptron units

\[ x_1 > 2? \]

\[ y_1 \]

\[ x_1 \]

\[ y_2 \]

\[ x_1 < -2? \]
Combine several perceptron units

\[ x_1 > 2? \]

\[ y_1 \]

\[ y \]

\[ x_1 \]

\[ y_2 \]

\[ x_1 < -2? \]
Multi-layer perceptron

\( w_{ij} \) represents weight to perceptron \( i \) in the hidden layer from input \( j \), and \( w_i \) represents weight of perceptron \( i \) in the hidden layer to the output. Then:

\[
y_i = \text{sgn}(\sum_j w_{ij}x_j) \tag{24}
\]

\[
y = \text{sgn}(\sum_i w_i \times y_i) \tag{25}
\]
How do we train this monster?

- Use differentiable perceptrons.
How do we train this monster?

- Use differentiable perceptrons.
- Minimize

\[ E(w) = \sum_i (y_i - f(x_i))^2 \]  \hspace{1cm} (26)

using gradient descent.
How do we train this monster?

- Use differentiable perceptrons.
- Minimize

\[ E(w) = \sum_i (y_i - f(x_i))^2 \]  \hspace{1cm} (26)

using gradient descent.

- Use chain rule to recursively compute gradients from output layer to input - pass information backwards.

\[ \frac{d}{dw_{11}} E(w) = \left( \frac{d}{dy_1} E(w) \right) \frac{dy_1}{dw_{11}} \]  \hspace{1cm} (27)
How powerful is this hidden layer?
http://playground.tensorflow.org/