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*

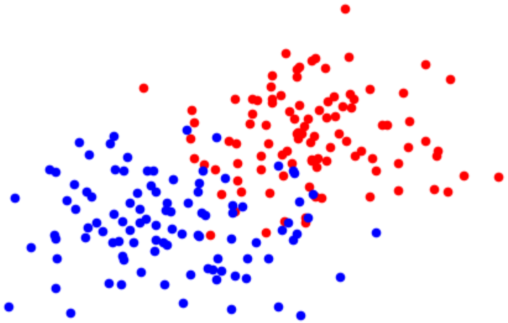
- Given a set of tuples $\{X_i, Y_i\}$, learn a function f which tells us Y_i for a given X_i.
- X_i is the *feature vector*, Y_i is the *label*, f is the *classifier*.

Learn to *tell apart*

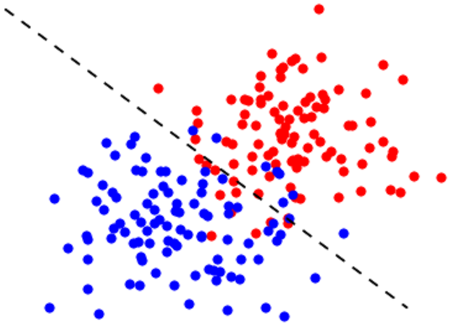
- Given a set of tuples $\{X_i, Y_i\}$, learn a function f which tells us Y_i for a given X_i.
- X_i is the *feature vector*, Y_i is the *label*, f is the *classifier*.
- e.g. $X =$ (vectorized) pixel intensity, $Y =$ image type

Outline

What's the classifier

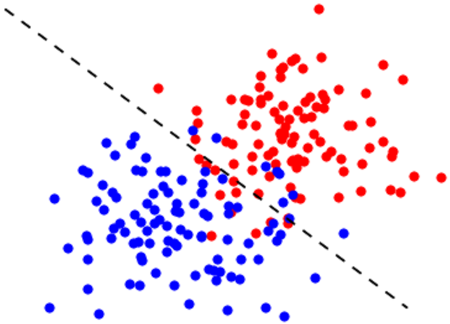


What's the classifier



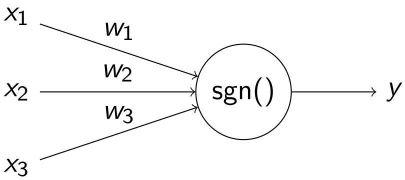
- Simply check which side are we on.

What's the classifier



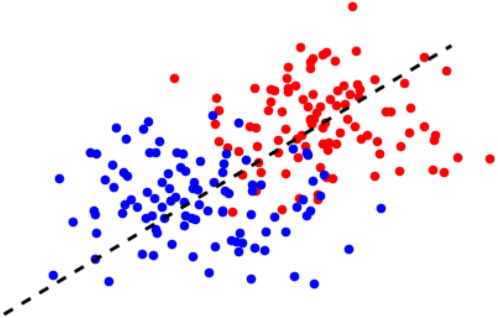
- Simply check which side are we on.
- Predict $sgn(\mathbf{w}^T\mathbf{x})$, where $\mathbf{w}$ is the **normal** to the boundary.

What's a perceptron



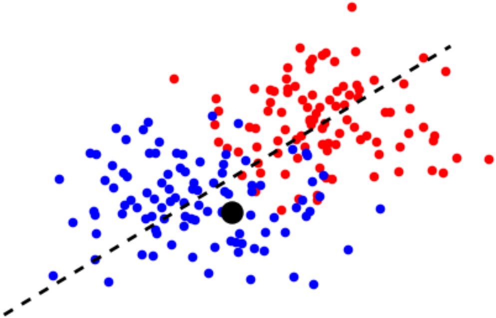
Outline

Finding a classifier



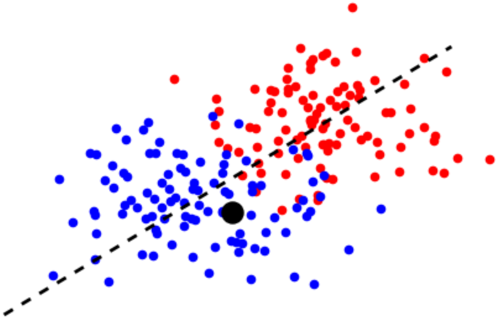
- Start with a random guess.

Finding a classifier



- Cycle through the training set. Check prediction y' vs actual label y.

Finding a classifier

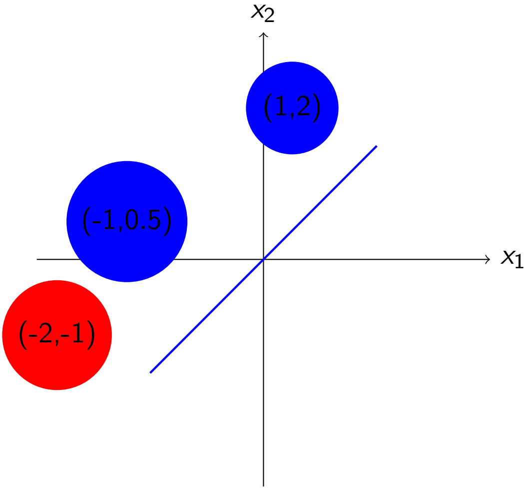


- Update line with the rule:

$$\mathbf{w} = \mathbf{w} + \alpha(y - y')\mathbf{x} \tag{1}$$

Finding a classifier

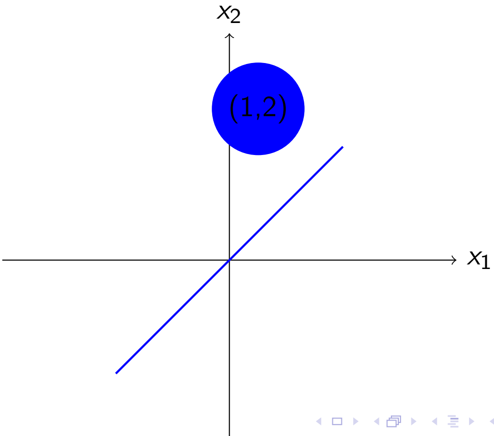
Suppose $\mathbf{w} = (1, -1)$



Finding a classifier

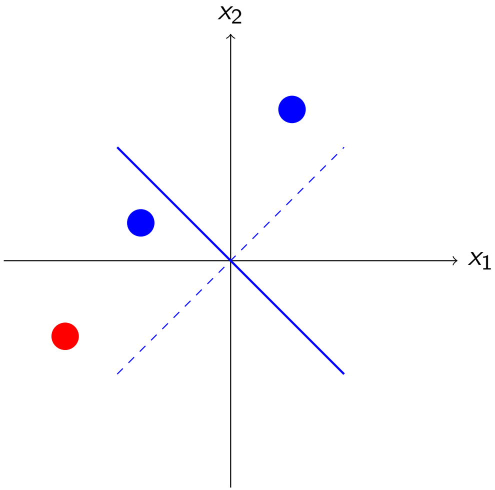
Consider $(1,2)$ with $y = 1$. For this $y' = -1$. For $\alpha = 1$ the new $\mathbf{w}$ is

$$\mathbf{w} = (1,-1) + 1 \times (1 - (-1)) \times (1,2) \quad (2)$$
$$\mathbf{w} = (3,3) \quad (3)$$



Finding a classifier

The boundary is now:

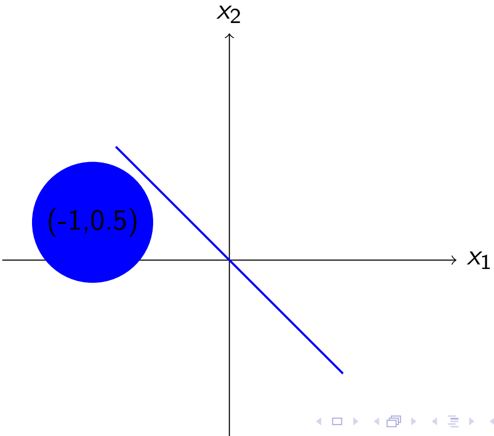


Finding a classifier

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

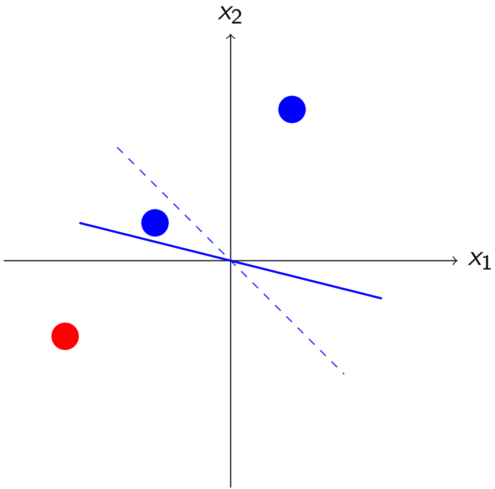
$$\mathbf{w} = (3,3) + 1 \times (1 - (-1)) \times (-1, 0.5) \tag{4}$$

$$\mathbf{w} = (1,4) \tag{5}$$



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 α.

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 α.
- 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 α.
- 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.

$$y = sgn(\mathbf{w}^T\mathbf{x} + b) \tag{6}$$

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 α.
- 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.

$$y = sgn(\mathbf{w}^T\mathbf{x} + b) \tag{6}$$

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

$$\tilde{\mathbf{x}} = \{\mathbf{x}, 1\}, \quad \tilde{\mathbf{w}} = \{\mathbf{w}, b\} \tag{7}$$

Differentiable Variant

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

Differentiable Variant

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

$$E(\mathbf{w}) = \sum_i (y_i - f(\mathbf{x}_i))^2 \tag{8}$$

Differentiable Variant

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

$$E(\mathbf{w}) = \sum_i (y_i - f(\mathbf{x}_i))^2 \tag{8}$$

- Update via gradient descent

$$\mathbf{w} = \mathbf{w} - \alpha \frac{d}{d\mathbf{w}} E(\mathbf{w}) \tag{9}$$

Differentiable Variant

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

$$E(\mathbf{w}) = \sum_i (y_i - f(\mathbf{x}_i))^2 \quad (8)$$

- Update via gradient descent

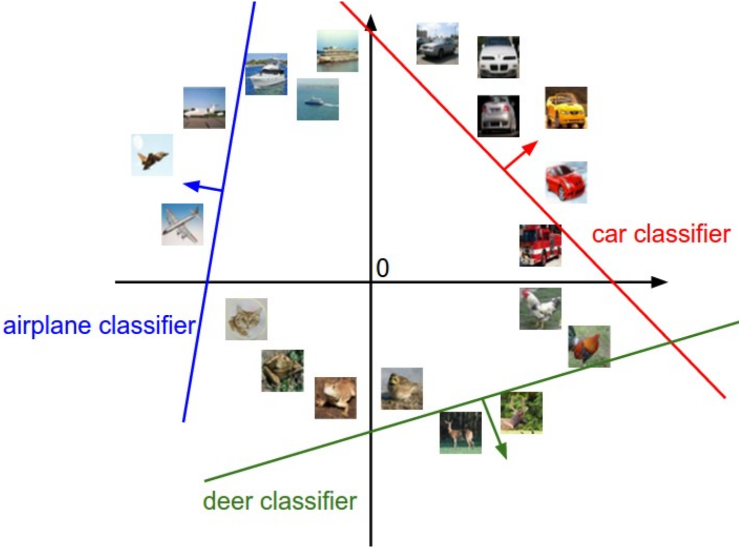
$$\mathbf{w} = \mathbf{w} - \alpha \frac{d}{d\mathbf{w}} E(\mathbf{w}) \quad (9)$$

- For the sigmoid, this is:

$$\mathbf{w} = \mathbf{w} - \alpha(y - f(\mathbf{x}))f(\mathbf{x})(1 - f(\mathbf{x}))\mathbf{x} \quad (10)$$

Outline

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

$$c = argmax_{c'} \quad \mathbf{w}_{c'}^T \mathbf{x} \tag{11}$$

One vs Others

- One classifier for one class.
- Predict

$$c = argmax_{c'} \quad \mathbf{w}_{c'}^T \mathbf{x} \tag{11}$$

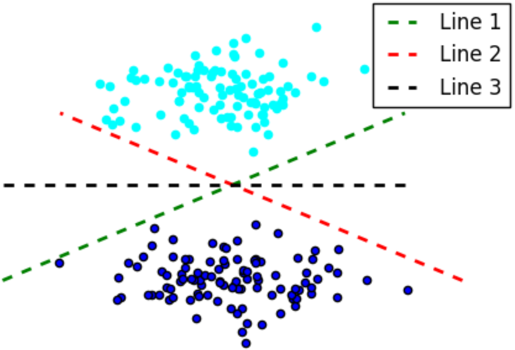
- If c is misclassified as c', update using

$$\mathbf{w}_c = \mathbf{w}_c + \alpha \mathbf{x} \tag{12}$$

$$\mathbf{w}_{c'} = \mathbf{w}_{c'} - \alpha \mathbf{x} \tag{13}$$

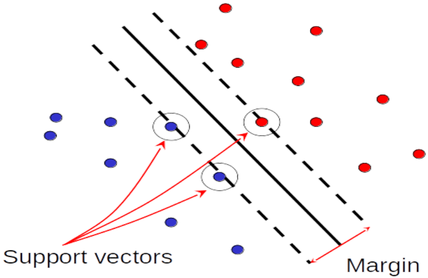
Outline

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:

$$\frac{\left|\mathbf{w}^T\mathbf{x} + b\right|}{\|\mathbf{w}\|} \tag{14}$$

Which boundary is better?

- Perpendicular distance of support vectors from the boundary is:

$$\frac{\left|\mathbf{w}^T\mathbf{x} + b\right|}{\|\mathbf{w}\|} \tag{14}$$

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

$$\left|\mathbf{w}^T\mathbf{x} + b\right| = 1 \tag{15}$$

Which boundary is better?

- Perpendicular distance of support vectors from the boundary is:

$$\frac{\left|\mathbf{w}^T\mathbf{x} + b\right|}{\|\mathbf{w}\|} \tag{14}$$

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

$$\left|\mathbf{w}^T\mathbf{x} + b\right| = 1 \tag{15}$$

- The margin is then $\frac{2}{\|\mathbf{w}\|}$

Which boundary is better?

- Formulated as:

$$min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{16}$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \tag{17}$$

Which boundary is better?

- Formulated as:

$$min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{16}$$

  subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \tag{17}$$

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

Which boundary is better?

- Formulated as:

$$min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{16}$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \tag{17}$$

- Eq. 16 ensures maximum margin, while Eq. 17 ensures correct classification.
- The classifier is of the form:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \tag{18}$$

and

$$y = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b \tag{19}$$

where α_i are learned weights.

Which boundary is better?

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

$$min_{\mathbf{w},b} \;\; \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \quad (20)$$

Which boundary is better?

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

$$min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \quad (20)$$

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

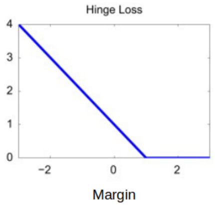
Which boundary is better?

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

$$min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \quad (20)$$

- C allows you to give weight to one over the other.
- Here we judge classification accuracy with the 'hinge' loss

$$max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \quad (21)$$



Outline

Supervised Classification

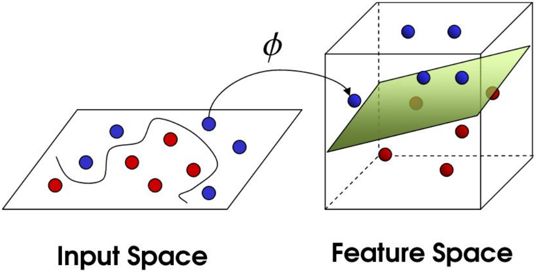
Perceptrons
- Linear Separability
- Training Algorithm
- Multi-class classification

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

Neural Networks
- Hidden layers

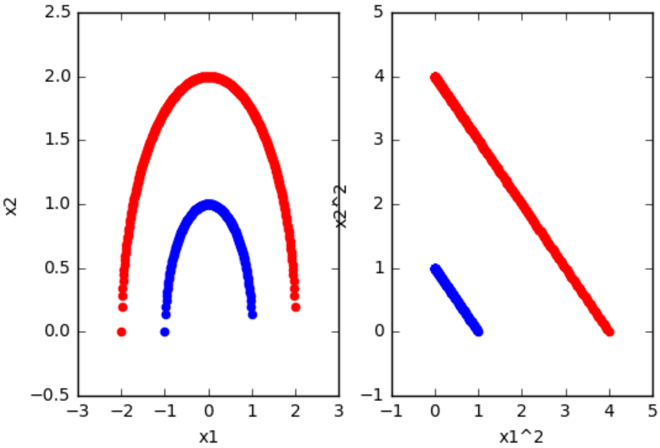
What if the data is not linearly separable?

- Try mapping it to a space where it is! Use a ϕ such that



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:

$$y = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \tag{22}$$

What if the data is not linearly separable?
The kernel trick

- Eq. 19 will be rewritten as:

$$y = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \tag{22}$$

- Instead of explicitly defining ϕ, we can also define a $K(\mathbf{x}, \mathbf{x'})$ such that

$$y = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{23}$$

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

Outline

Supervised Classification

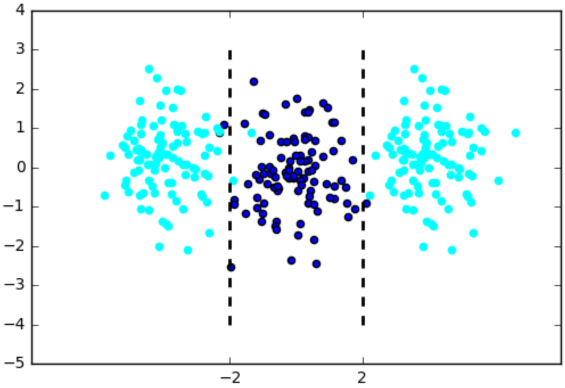
Perceptrons
- Linear Separability
- Training Algorithm
- Multi-class classification

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

Neural Networks
- Hidden layers

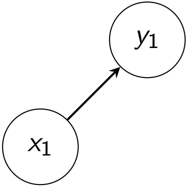
Can we learn this transformation?

- Stacks of perceptrons can learn non-linear functions. e.g. Consider a simple 1-d scenario

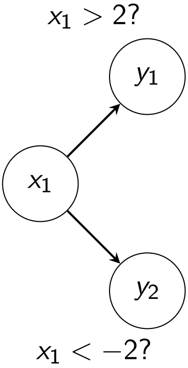


Combine several perceptron units

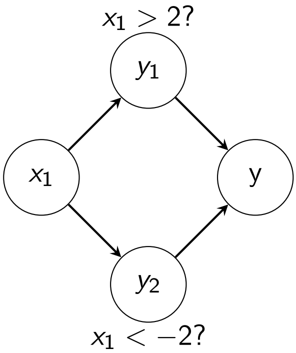
$x_1 > 2?$



Combine several perceptron units



Combine several perceptron units

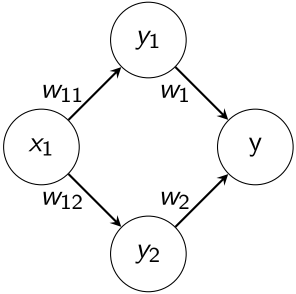


Multi-layer perceptron

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:

$$y_i = sgn(\sum_j w_{ij} x_j) \quad (24)$$

$$y = sgn(\sum_i w_i \times y_i) \quad (25)$$



How do we train this monster?

- Use differentiable perceptrons.

How do we train this monster?

- Use differentiable perceptrons.
- Minimize

$$E(\mathbf{w}) = \sum_i (y_i - f(\mathbf{x}_i))^2 \tag{26}$$

using gradient descent.

How do we train this monster?

- Use differentiable perceptrons.
- Minimize

$$E(\mathbf{w}) = \sum_i (y_i - f(\mathbf{x}_i))^2 \tag{26}$$

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

$$\frac{d}{dw_{11}} E(\mathbf{w}) = (\frac{d}{dy_1} E(\mathbf{w})) \frac{dy_1}{dw_{11}} \tag{27}$$

How powerful is this hidden layer?

http://playground.tensorflow.org/