01/27/2016

Perceptron algorithm continued

- Duality between x-space and w-space:

x-space (primal)	w-space (dual)
hyperplan e: $\{y : w \cdot y = 0\}$	point: w

  - If a point $x \in H$ (hyperplane), then its dual hyperplane x^* contains the dual point H^* (it also preserves orientation).

- If we want to enforce inequality $x \cdot w \geq 0$, that means:
  - x should be in the correct side of $\{y : y \cdot w = 0\}$ in x-space.
  - w should be on the correct side of $\{v : x \cdot v = 0\}$ in w-space.
  - Add image
  - Note: if data is linearly separable there will always be a section where you can put w.

Optimization algorithm 1

Gradient descent on R.

- Given a starting point w, find gradient of R with respect to w; this is the direction of the steepest ascent. Take a step in the opposite direction.

$$\nabla R(w) = \begin{bmatrix} \frac{\partial R}{\partial w_1} \\ \frac{\partial R}{\partial w_2} \\ \vdots \\ \frac{\partial R}{\partial w_d} \end{bmatrix} \quad \text{and} \quad \nabla(z \cdot w) = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_d \end{bmatrix} = z$$

$$\nabla R(w) = \sum_{i \in V} \nabla - y_i x_i \cdot w = \sum_{i \in V} -y_i x_i$$

- At any point w, we walk downhill in direction of steepest descent; $-\nabla R$.

```
Gradient Descent:
    w ← arbitrary non-zero starting point (good choice is any y_i x_i)
    while (R(w) > 0):
        V ← the set of indices i for which y_i x_i · w < 0
        w ← w + ε Σ_{i∈V} y_i x_i
```

- Here ϵ is the step size (aka learning rate chosen empirically).
- Problem: Slow! Each step takes $\mathcal{O}(nd)$ time.

Optimization algorithm 2

Stochastic gradient descent

- Idea: Each step, pick <u>one</u> misclassified x_i; do gradient descent on loss function $L(x_i \cdot w, y_i)$.
- Called perceptron algorithm. Each step takes $\mathcal{O}(d)$ time.

```
Perceptron Algorithm:
    while (some y_i X_i < 0):
            w ← w + ε y_i x_i
```

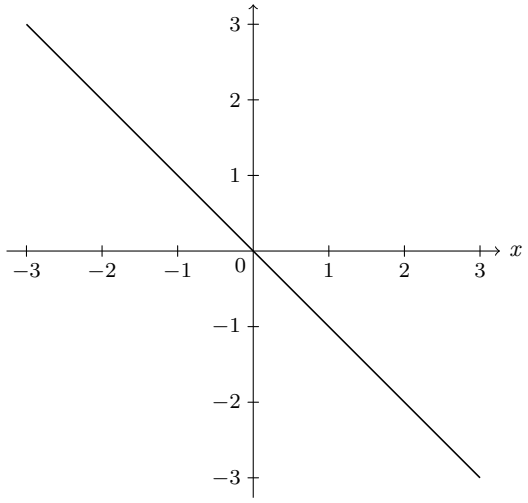
- What if separating hyperplane doesn't pass through the origin?
  - Add a fictitious dimension.
  - Hyperplane: $w \cdot x + \alpha = 0$,

$$\begin{bmatrix} w_1 & w_2 & \dots & w_d & \alpha \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}$$

  - Now we have samples in $\mathbb{R}^{d+1}$, all lying on plane $X_{d+1} = 1$.
- Perceptron Convergence Theorem: If data is linearly separable, perceptron algorithm finds a correct linear classifier in at most $\mathcal{O}(\frac{R^2}{\gamma^2})$ iterations; where $R = \max |X_i|$ is the "radius of data" and γ is the "margin" (length of longest feature vector).

Maximum Margin Classifiers

- The margin of a linear classifier is the distance from the decision boundry to the nearest sample.
- Lets make the margin as big as possible.



- We enforce constraints $y_i(w \cdot x_i + \alpha) \geq 1 \quad \forall i \in [1, n]$.
- If $||w|| = 1$, constraints imply the margin is at least 1.
- But we allow to have any length, so the margin is at least $\frac{1}{||w||}$.

- There is a slab of width $\frac{2}{||w||}$ that contains no samples.
- To maximize the margin, minimize $||w||$.
- Optimization Problem:

$$\text{Find } w \text{ and } \alpha \text{ that maximize } ||w||^2 \text{ subject to } y_i(w \cdot x_i + \alpha) \geq 1 \quad \forall i \in [1, n].$$

- This is called a quadratic program in $d + 1$ dimensions and n constraints.
- It has one unique solution!
- The solution gives us a maximum margin classifier, aka a hard margin support vector machine (SVM).