Perceptron, Kernels, and SVM

CSE 546 Recitation
November 5, 2013

Grading Update

- Midterms: likely by Monday
  - Expected average is 60%
- HW 2: after midterms are graded
- Project proposals: mostly or all graded (everyone gets full credit)
  - Check your dropbox for comments
- HW 3 scheduled to be released tomorrow, due in two weeks

Perceptron Basics

- Online algorithm
- Linear classifier
- Learns set of weights
- $w^{t+1} \leftarrow w^t + y^{t+1} x^{t+1} \mathbb{I}(sign(x^{t+1} \cdot w^t) \neq y^{t+1})$
- Always converges on linearly separable data

What does perceptron optimize?

- Perceptron appears to work, but is it solving an optimization problem like every other algorithm?
- $yw \cdot x < 0$ Is equivalent to making a mistake
- Hinge loss penalizes mistakes by

$$\ell(w, x, y) = 0 \text{ if } yw \cdot x \geq 0$$
$$\ell(w, x, y) = -yw \cdot x \text{ if } yw \cdot x < 0$$

Hinge Loss

$$\min \frac{1}{N} \sum_{j=1}^{N} l(w, x^j, y^j) = \frac{1}{N} \sum \left(-y^j w \cdot x^j\right)_+$$

- Gradient descent update rule:

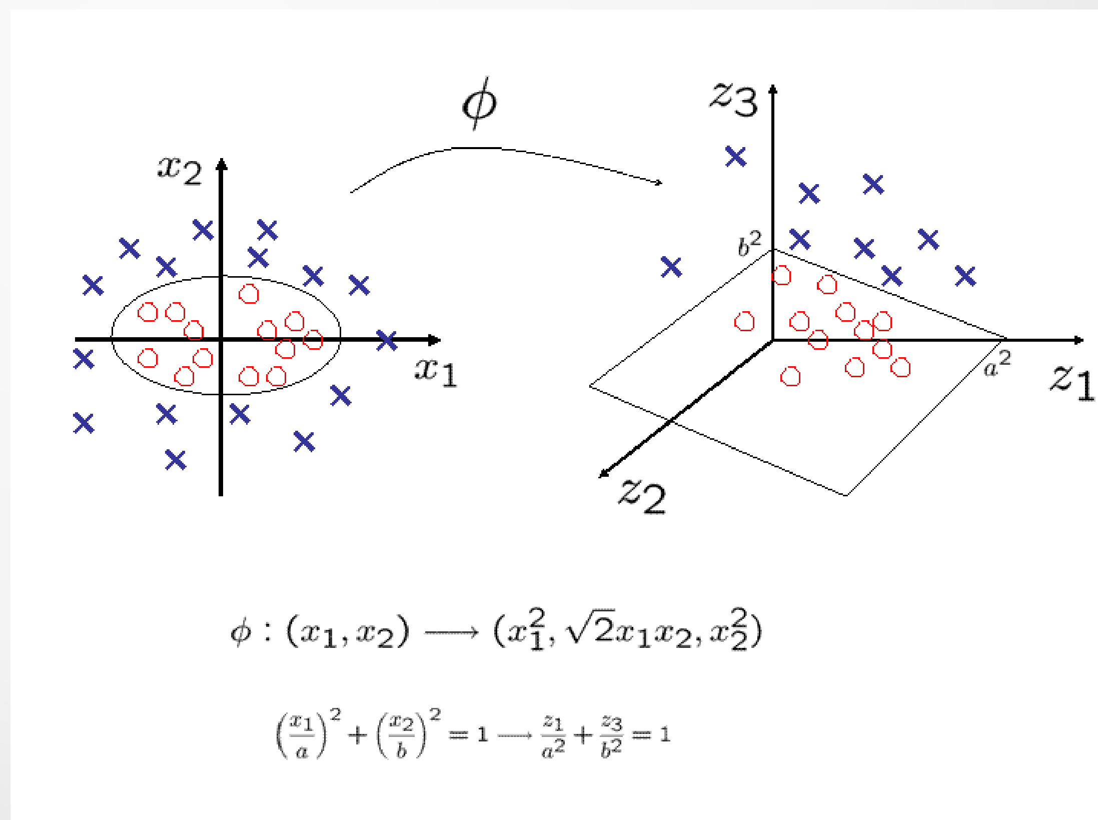
$$w^{t+1} \leftarrow w^t + \eta \frac{1}{N} \sum_{i=1}^{N} y^i x^i \mathbb{I}(y^i w^t \cdot x^i \leq 0)$$

- Stochastic gradient descent update rule = perceptron:

$$w^{t+1} \leftarrow w^t + y^{t+1} x^{t+1} \mathbb{I}(y^{t+1} w^t \cdot x^{t+1} \leq 0)$$

Feature Maps

- What if data aren't linearly separable?
- Sometimes if we map features to new spaces, we can put the data in a form more amenable to an algorithm, e.g. linearly separable
- The maps could have extremely high or even infinite dimension, so is there a shortcut to represent them?
  - Don't want to store every $\phi(x)$ or do computation in high dimensions



Kernel Trick

- Kernels (aka kernel functions) represent dot products of mapped features in same dimension as original features
  - Apply to algorithms that only depend on dot product
- $k(u,v) = \phi(u) \cdot \phi(v)$
  - Lower dimension for computation
  - Don't have to store $\phi(x)$ explicitly
- Choose mappings that have kernels, since not all do
  - e.g. $\phi((x_1, x_2)) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$

$$\phi(x) \cdot \phi(y) = x_1^2 y_1^2 + x_2^2 + y_2^2 + 2x_1y_1x_2y_2 = (x_1y_1 + x_2y_2)^2$$

$$= (x \cdot y)^2$$

Kernelized Perceptron

- Recall perceptron update rule:

$$w^{t+1} \leftarrow w^t + y^{t+1}x^{t+1}\mathbb{I}(sign(x^{t+1} \cdot w^{t+1}) \neq y^{t+1})$$

  - Implies: $w^t = \sum_{i \in M^t} y^i x^i$ where M^t is mistake indices up to t

- Classification rule: $\hat{y} = sign(w^t \cdot x) = sign(\sum_{i \in M^t} y^i (x^i \cdot x))$

- With mapping ϕ :

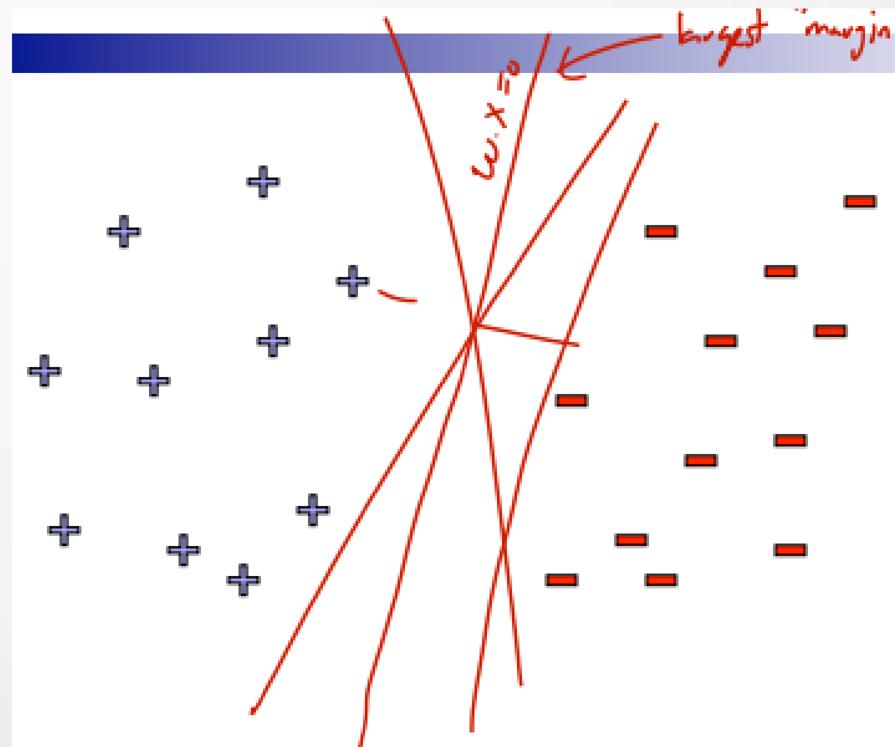
$$\hat{y} = sign(w^t \cdot \phi(x))$$
$$= sign(\sum_{i \in M^t} y^i (\phi(x^i) \cdot \phi(x)))$$

- If have kernel $k(u, v) = \phi(u) \cdot \phi(v)$:

$$\hat{y} = sign(w^t \cdot x) = sign(\sum_{i \in M^t} y^i k(x^i, x))$$

SVM Basics

- Linear classifier (without kernels)
- Find separating hyperplane by maximizing margin
- One of the most popular and robust classifiers



Setting Up SVM Optimization

- Weights w and margin γ

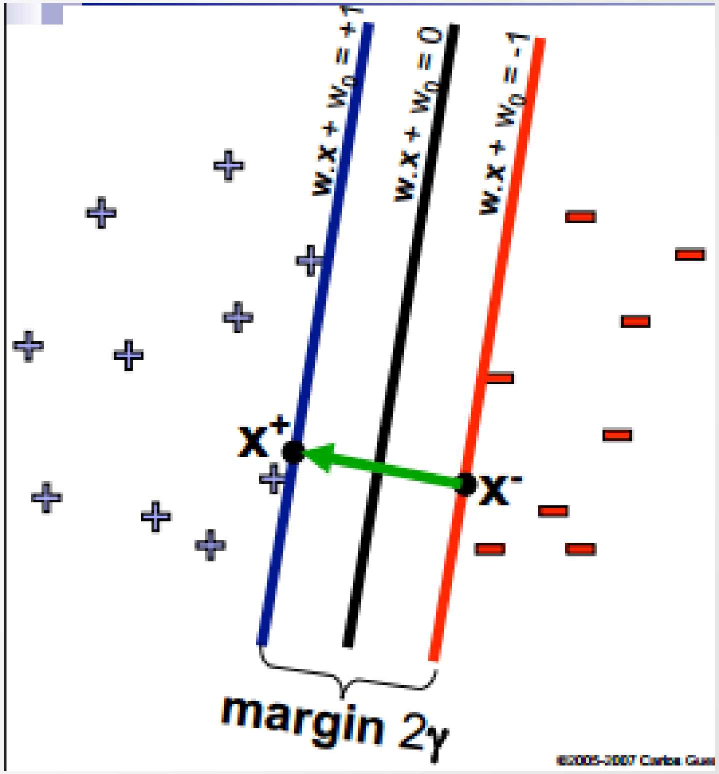
$$\max_{\gamma,\mathbf{w},w_0} \quad \gamma$$

$$y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0) \geq \gamma, \forall j \in \{1,\ldots,N\}$$

  - Optimization unbounded
- Use canonical hyperplanes to remedy
  - $\gamma = 1/||w||$
- If linearly separable data, can solve

$$\min_{\mathbf{w},w_0} \quad ||w||_2^2$$

$$y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0) \geq 1, \forall j \in \{1,\ldots,N\}$$



SVM Optimization

- If non-linearly separable data, could map to new space
  - But doesn't guarantee separability
- Therefore, remove separability constraints

  $y^j(w \cdot x^j + w_0) \geq 1$

  and instead penalize the violation in the objective

$$\min \|w\|_2^2 + C \sum_{j=1}^{N} (1 - y^j(w \cdot x^j + w_0))_+$$

  - Soft-margin SVM minimizes regularized hinge loss

SVM vs Perceptron

- SVM

$$\min \|w\|_2^2 + C\sum_{j=1}^{N}(1 - y^j(w \cdot x^j + w_0))_+$$

has almost same goal as L2-regularized perceptron

- Perceptron

$$\min \sum_{j=1}^{N}\left(-y^j(w \cdot x^j + w_0)\right)_+$$

Other SVM Comments

- C > 0 is “soft margin”
  - High C means we care more about getting a good separation
  - Low C means we care more about getting a large margin
- How to implement SVM?
  - Suboptimal method is SGD (see HW 3)
  - More advanced methods can be used to employ the kernel trick

Questions?