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

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

- 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
  \[
  \ell(w, x, y) = \begin{cases} 
  0 & \text{if } yw \cdot x \geq 0 \\
  -yw \cdot x & \text{if } yw \cdot x < 0
  \end{cases}
  \]
Hinge Loss

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

- 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

\[ \phi : (x_1, x_2) \rightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2) \]

\[ \left( \frac{x_1}{a} \right)^2 + \left( \frac{x_2}{b} \right)^2 = 1 \quad \rightarrow \quad \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1 \]
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. \[ \phi((x_1, x_2)) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \]
  \[ \phi(x) \cdot \phi(y) = x_1^2y_1^2 + x_2^2 + y_2^2 + 2x_1y_1x_2y_2 = (x_1y_1 + x_2y_2)^2 = (x \cdot y)^2 \]
Kernelized Perceptron

- Recall perceptron update rule:
  \[ w^{t+1} \leftarrow w^t + y^{t+1} x^{t+1} \mathbb{I}(\text{sign}(x^{t+1} \cdot w^{t+1}) \neq y^{t+1}) \]
  \[ w^t = \sum_{i \in M^t} y^i x^i \text{ where } M^t \text{ is mistake indices up to } t \]

- Classification rule:
  \[ \hat{y} = \text{sign}(w^t \cdot x) = \text{sign}\left( \sum_{i \in M^t} y^i (x^i \cdot x) \right) \]

- With mapping \( \phi \):
  \[ \hat{y} = \text{sign}(w^t \cdot \phi(x)) = \text{sign}\left( \sum_{i \in M^t} y^i (\phi(x^i) \cdot \phi(x)) \right) \]

- If have kernel \( k(u, v) = \phi(u) \cdot \phi(v) \):
  \[ \hat{y} = \text{sign}(w^t \cdot x) = \text{sign}\left( \sum_{i \in M^t} y^i k(x^i, x) \right) \]
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 $\mathbf{w}$ and margin $\gamma$
  
  \[
  \max_{\gamma, \mathbf{w}, w_0} \gamma \\
  y^j (\mathbf{w} \cdot \mathbf{x}^j + w_0) \geq \gamma, \forall j \in \{1, \ldots, N\} 
  \]

  - Optimization unbounded

- Use canonical hyperplanes to remedy
  
  - $\gamma = 1/\|\mathbf{w}\|$

- If linearly separable data, can solve
  
  \[
  \min_{\mathbf{w}, w_0} \|\mathbf{w}\|_2^2 \\
  y^j (\mathbf{w} \cdot \mathbf{x}^j + w_0) \geq 1, \forall j \in \{1, \ldots, N\} 
  \]
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
  \[
  \min \sum_{j=1}^{N} (-y^j (w \cdot x^j + w_0))_+
  \]
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?