Instructor: Joelle Pineau (jpineau@cs.mcgill.ca)

TAs: Pierre-Luc Bacon (pbacon@cs.mcgill.ca)
    Ryan Lowe (ryan.lowe@mail.mcgill.ca)

Class web page: www.cs.mcgill.ca/~jpineau/comp598

Non-linearly separable data

- A linear boundary might be too simple to capture the data.

- One option is to relax the constraints and allow some points to be misclassified by the margin.

- Another way to get a nonlinear decision boundary in the input space is to find a linear decision boundary in an expanded space (e.g. recall adding polynomial terms in linear regression.)
  - Thus $x_i$ is replaced by $\phi(x_i)$, where $\phi$ is called a feature mapping.
Soften the primal objective

- We wanted to solve:
  \[
  \min_w \frac{1}{2} ||w||^2 \\
  \text{s.t. } yw^T x_i \geq 1
  \]

- This can be re-written:
  \[
  \min_w \sum_i L_{\infty}(w^T x_i, y_i) + \frac{1}{2} ||w||^2 \\
  \text{s.t. } yw^T x_i \geq 1
  \]
  where \(\sum_i L_{\infty}(w^T x_i, y_i) = (\infty \text{ for a misclassification, } 0 \text{ correct classification})\)

- Soften misclassification cost:
  \[
  \min_w \sum_i L_{0-1}(w^T x_i, y_i) + \frac{1}{2} ||w||^2 \\
  \text{s.t. } yw^T x_i \geq 1
  \]
  where \(\sum_i L_{0-1}(w^T x_i, y_i) = (1 \text{ for a misclassification, } 0 \text{ correct classification})\)

- But this is a non-convex objective!

Approximation of the \(L_{0-1}\) function

\[L_{0-1}(y, f(x)) = \begin{cases} 
  1 & y(f(x)) < 0 \\
  0 & y(f(x)) \geq 0 
\end{cases}\]
SVM with hinge loss

- Hinge loss: \( L_{\text{hin}}(w^T x_i, y) = \max \{1 - y w^T x_i, 0\} \)

- Softhen misclassification cost: 
  \[
  \min_w C \sum_i L_{\text{hin}}(w^T x_i, y) + \frac{1}{2} ||w||^2
  \]
  where \( C \) controls trade-off between slack penalty and margin.

- The hinge loss upper-bounds the 0-1 loss.
  \[
  \xi_i \geq 1 - y w^T x_i \geq L_{0-1}(w^T x_i, y)
  \]

Primal Soft SVM problem

- Define slack variables \( \xi_i = L_{\text{hin}}(w^T x_i, y) = \max \{1 - y w^T x_i, 0\} \)

- Solve: 
  \[
  \hat{w}_{\text{soft}} = \arg\min_w C \sum_{i=1}^n \xi_i + \frac{1}{2} ||w||^2 
  \]
  Add Lagrange mult: 
  s. t. 
  \[
  y_i w^T x_i \geq 1 - \xi_i, \quad i = 1, ..., n 
  \]
  \[
  \xi_i \geq 0, \quad i = 1, ..., n
  \]
  where \( w \in \mathbb{R}^m, \xi \in \mathbb{R}^n \)

- Introduce Lagrange multipliers: 
  \[
  \alpha = (\alpha_1, \alpha_2, ..., \alpha_n)^T, \ 0 \leq \alpha_i
  \]
  \[
  \beta = (\beta_1, \beta_2, ..., \alpha_n)^T, \ 0 \leq \beta_i
  \]
Soft SVM problem: Adding Lagrange multipliers

- **Primal** objective: 
  \[(w, \xi, \alpha, \beta) = \arg \min_{w, \xi, \alpha, \beta} L(w, \xi, \alpha, \beta)\]
  where 
  \[L(w, \xi, \alpha, \beta) = \frac{1}{2} ||w||^2 + C \sum_{i=1:n} \xi_i - \sum_{i=1:n} \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_{i=1:n} \beta_i \xi_i\]

- **Dual** (invert min and max): 
  \[(w, \xi, \alpha, \beta) = \arg \max_{\alpha, \beta} \min_{w, \xi} L(w, \xi, \alpha, \beta)\]

- Solve:
  \[
  \frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i = 0 \implies w^* = \sum_i \alpha_i y_i x_i
  \]
  \[
  \frac{\partial L}{\partial \xi} = C1_n - \alpha - \beta = 0 \implies \beta = C1_n - \alpha
  \]
  Lagrange multipliers are positive, so we have: 
  \[0 \leq \beta_i, \ 0 \leq \alpha_i \leq C\]

- Plug into dual: 
  \[\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)\]
  with constraints \[0 \leq \alpha_i \leq C\] and \[\sum \alpha_i y_i = 0\].

- This is a quadratic programming problem (similar to Hard SVM).

---

Soft SVM solution

- Soft-SVM has one more constraint \[0 \leq \alpha_i \leq C\] (vs \[0 \leq \alpha_i\] in Hard SVM).
- When \[C=\infty\], then Soft-SVM=>Hard-SVM.
- Points on the margin have \[\alpha_i > 0\] and \[\xi_i=0\].
- Points away from margin have \[\alpha_i = 0\].
- Points on the decision line have \[\xi_i = 1\].
- Misclassified points have \[\xi_i > 1\].

- To predict on test data: 
  \[
  h_w(x) = \text{sign}(\sum_{i=1:n} \alpha_i y_i (x_i \cdot x))
  \]

- Only need to store the support vectors (i.e. points on the margin) to predict.
Multiple classes

- One-vs-All: Learn K separate binary classifiers.
  - Can lead to inconsistent results.
  - Training sets are imbalanced, e.g. assuming n examples per class,
    each binary classifier is trained with positive class having 1*n of the
data, and negative class having (K-1)*n of the data.

- Multi-class SVM: Define the margin to be the gap between the
  correct class and the nearest other class.

SVMs for regression

- Minimize a regularized error function:
  \[
  \hat{w} = \arg\min_w C \sum_{i=1:n} (y_i^*w^T x_i)^2 + \frac{1}{2}\|w\|^2
  \]

- Typically, relax to \(\varepsilon\)-sensitive error on the linear target to ensure sparse
  solution (i.e. few support vectors):
  \[
  \hat{w} = \arg\min_w C \sum_{i=1:n} E_{\varepsilon}(y_i^*w^T x_i)^2 + \frac{1}{2}\|w\|^2
  \]
  where
  \[
  E_{\varepsilon} = \begin{cases} 
  0 & \text{if } (y_i^*w^T x_i) < \varepsilon, \\
  (y_i^*w^T x_i) - \varepsilon & \text{otherwise}
  \end{cases}
  \]

- Introduce slack variables to optimize a
  “tube” around the regression function.
Non-linearly separable data: Another trick

- A linear boundary might be too simple to capture the data.

- One option is to relax the constraints and allow some points to be misclassified by the margin (Soft SVM).

- Another way to get a nonlinear decision boundary in the input space is to find a linear decision boundary in an expanded space (e.g. recall adding polynomial terms in linear regression.)
  - Thus $x_i$ is replaced by $\phi(x_i)$, where $\phi$ is called a feature mapping.

Margin optimization in feature space

- Replacing $x_i$ by $\phi(x_i)$, the optimization problem for $w$ becomes:

  - **Primal form**: $\text{Min} \quad \frac{1}{2} ||w||^2$
    
    w.r.t. $w$
    
    s.t. $y_i w^T \phi(x_i) \geq 1$

  - **Dual form**: $\text{Max} \quad \sum_i a_i - \frac{1}{2} \sum_{ij} y_i y_j a_i a_j (\phi(x_i) \cdot \phi(x_j))$
    
    w.r.t. $a_i$
    
    s.t. $a_i \geq 0$
    
    $\sum_i a_i y_i = 0$
Feature space solution

- The optimal weights, in the expended feature space, are
  \[ w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \]
- Classification of an input \( x \) is given by:
  \[ h_w(x) = \text{sign}( \sum_{i=1}^{n} \alpha_i y_i (\phi(x_i) \cdot \phi(x))) \]
- Note that to solve the SVM optimization problem in dual form and to make a prediction, we only ever need to compute dot-products of feature vectors.

Kernel functions

- Whenever a learning algorithm (such as SVMs) can be written in terms of dot-products, it can be generalized to kernels.
- A kernel is any function \( K: \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R} \), which corresponds to a dot product for some feature mapping \( \phi \):
  \[ K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2) \text{ for some } \phi \]
- Conversely, by choosing feature mapping \( \phi \), we implicitly choose a kernel function.
- Recall that \( \phi(x_1) \cdot \phi(x_2) = \cos \angle(x_1, x_2) \), where \( \angle \) denotes the angle between the vectors, so a kernel function can be thought of as a notion of similarity.
Example: Quadratic kernel

- Let $K(x, z) = (x \cdot z)^2$.
- Is this a kernel?
  \[
  K(x, z) = (\sum_{i=1}^{m} x_i z_i) (\sum_{j=1}^{m} x_j z_j)
  = \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j
  = \sum_{i,j \in \{1..m\}} (x_i x_j) (z_i z_j)
  \]
- Hence it is a kernel, with feature mapping:
  \[
  \phi(x) = \langle x_1^2, x_1 x_2, \ldots, x_1 x_m, x_2^2, \ldots, x_m^2 \rangle
  \]
  Feature vector includes all squares of elements and all cross terms.
- Note that computing $\phi$ takes $O(m^2)$ but computing $K$ only takes $O(m)$.

Polynomial kernels

- More generally, $K(x, z) = (x \cdot z)^d$ is a kernel, for any positive integer $d$:
  \[
  K(x, z) = (\sum_{i=1}^{m} x_i z_i)^d
  \]
- If we expanded the sum above in the obvious way, we’d get $n^d$ terms (i.e. feature expansion).
- Terms are monomials (products of $x_i$) with total power equal to $d$.
- If we use the primal form of the SVM, each term gets a weight.
- **Curse of dimensionality**: it is very expensive both to optimize and to predict with an SVM in primal form.
- However, evaluating the dot-produce of any two feature vectors can be done using $K$ in $O(m)$.  

The “kernel trick”

• If we work with the dual, we do not have to ever compute the feature mapping \( \phi \). We just compute the similarity kernel \( K \).

• We can solve the dual for the \( \alpha_i \):

\[
\begin{align*}
\text{Max} & \quad \sum_{i=1:n} \alpha_i - \frac{1}{2} \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\
\text{w.r.t.} & \quad \alpha_i \\
\text{s.t.} & \quad \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1:n} \alpha_i y_i = 0
\end{align*}
\]

• The class of a new input \( x \) is computed as:

\[
h_w(x) = \text{sign} \left( \sum_{i=1:n} \alpha_i y_i K(x_i, x) \right)
\]

where \( x_i \) are the support vectors (defining the margin).

• Remember, \( K(\cdot, \cdot) \) can be evaluated in \( O(m) \) time = big savings!

Some other kernel functions

• \( K(x, z) = (1 + x \cdot z)^d \) - feature expansion has all monomial terms of total power.

• Radial basis / Gaussian kernel: \( K(x, z) = \exp \left( -||x-z||^2 / 2\sigma^2 \right) \)
  – This kernel has an infinite-dimensional feature expansion, but dot-products can still be computed in \( O(m) \) (where \( m=\#\text{features} \))

• Sigmoidal kernel: \( K(x, z) = \tanh(c_1 x \cdot z + c_2) \)
Example: Gaussian kernel

Note the non-linear decision boundary

Kernels beyond SVMs

- A lot of current research has to do with defining new kernel functions, suitable to particular tasks / kinds of inputs.

- Many kernels are available:
  - Information diffusion kernels (Lafferty and Lebanon, 2002)
  - Diffusion kernels on graphs (Kondor and Jebara, 2003)
  - String kernels for text classification (Lodhi et al, 2002)
  - String kernels for protein classification (Leslie et al, 2002)
  ... and others!
Example: String kernels

- Very important for DNA matching, text classification, …
- Example in DNA matching, we use a sliding window of length \( k \) over the two strings that we want to compare.
- The window is of a given size, and inside we can do various things:
  - Count exact matches.
  - Weigh mismatches based on how bad they are.
  - Count certain markers, e.g. AGT.
- The kernel is the sum of these similarities over the two sequences.

Kernelizing other ML algorithms

- Many other machine learning algorithms have a “dual formulation”, in which dot-products of features can be replaced by kernels.

- Examples:
  - Perceptron
  - Logistic regression
  - Linear regression
What you should know

From last class:
- The perceptron algorithm.
- The margin definition for linear SVMs.
- The use of Lagrange multipliers to transform optimization problems.

From today:
- The primal and dual optimization problems for SVMs.
- Feature space version of SVMs.
- The kernel trick and examples of common kernels.