EECS E6690: SL for Bio & Info
Lecture 8: Support Vector Machines, Optimization, and Gene Expression Classification

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: http://www.ee.columbia.edu/∼predrag

Final Project Outline

- Done in groups of 4 students - assemble the groups
- Deliverables: 15+ page **paper** & **presentation** with slides
- **Due:** during the finals week: Dec 16 - 23, very likely **Dec 17**.
  One slot for presentations on **Tue, Dec 14, 4:10-6:40pm**.
- **Data Repositories**: First, select a paper(s) from either:
  - UC Irvine Machine Learning Repository
    `https://archive.ics.uci.edu/ml/datasets.php`
  - GEO Data Repository `https://www.ncbi.nlm.nih.gov/geo/`, or Bioconductor Datasets
    `http://www.bioconductor.org/packages/release/data/experiment/`
- **Final Paper Outline:** 5 sections
  1. Introduction: e.g., describe the application area, problems considered, etc
  2. Data set(s) and paper(s): e.g., describe data in detail, what was done in the paper(s), common stat/machine learning tools, etc
  3. Reproduce the results from the paper(s)
  4. Try different techniques learned in class, or propose new ones
  5. Discussion and conclusion: e.g., compare different techniques, pros and cons, future work, etc

Bioconductor and Additional Datasets

- Bioconductor provides tools in R for the analysis genomic data: `https://https://www.bioconductor.org/`
- Installing Bioconductor: `https://https://www.bioconductor.org/install/`
  Run the following code:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.12")
```

- Then, install Bioconductor packages: `https://www.bioconductor.org/install/#install-bioconductor-packages`
- Datasets supported by Bioconductor: `http://www.bioconductor.org/packages/release/data/experiment/`

Last lecture: Bootstrap Methods

- Stochastic search
- Works for both: classifiers or regressions
- Bumping, Bagging, Random forests, Boosting
- Train a classifier or regression model $\hat{f}_0$ on $\boldsymbol{Z}$
- For $b = 1, \ldots, B$:
  1. Draw a bootstrap sample $\boldsymbol{Z}^{*b}$ of size n from training data
  2. Train a classifier or regression model $\hat{f}_b$ on $\boldsymbol{Z}^{*b}$
- Bumping: Select the best model, e.g.,

$$\hat{b} = \arg \min_{0 \leq b \leq B} \sum_{i=1}^{n} \left(y_i - \hat{f}_b(\boldsymbol{z}_i) \right)^2$$

- Bagging: average out - reduces variance
  For a "new" point $\boldsymbol{x}_0$, compute:

$$\hat{f}_{\mathsf{avg}}(\boldsymbol{x}_0) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(\boldsymbol{x}_0)$$

Last lecture: Random forests

Works for both: classifiers or regressions

- Improvement over bagged trees
- Idea: Decorrelate trees
  - Still learn a tree on each bootstrap set
  - To split a region, consider only a subset of predictors

- Input parameter: $m \leq p$, often $m \approx \sqrt{p}$
- For $b = 1, \ldots, B$
  - Draw a bootstrap sample $\boldsymbol{Z}^{*b}$ of size n from the training data
  - Train a tree classifier on $\boldsymbol{Z}^{*b}$, each split is computed as:
    - Randomly select m predictors, newly chosen for each b
    - Make the best split restricted to that subsets of predictors

Last lecture: Boosting for regression

Slow learning

1. Set $\hat{f}(\boldsymbol{x}_i) = 0$ and $r_i = y_i$ for all i in the training set
2. For $b = 1, \ldots, B$, repeat:
   2.1 Fit a tree $\hat{f}_b$ with d splits ($d + 1$ terminal nodes) to training data $(\boldsymbol{X}, \boldsymbol{r})$
   2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(\boldsymbol{x}) \leftarrow \hat{f}(\boldsymbol{x}) + \lambda \hat{f}^b(\boldsymbol{x})$$
   2.3 Update residuals:
   $$r_i \leftarrow r_i - \lambda \hat{f}_b(\boldsymbol{x}_i)$$
3. Output the boosted model:
$$\hat{f}(\boldsymbol{x}) = \sum_{b=1}^{B} \lambda \hat{f}_b(\boldsymbol{x})$$

- Notes:
  - λ is a small positive number (e.g., 0.01 or 0.001)
  - Often $d = 1$ works

Boosting for Classification: AdaBoost

- Due to Freund and Schapire (1997)
- Consider two classes: $Y \in \{-1, 1\}$
- For a classifier $G(x) \in \{-1, 1\}$, the training error is

$$\mathrm{e}_m = \frac{1}{n} \sum_{i=1}^{n} 1_{\{y_i \neq G(x_i)\}}$$

- Main idea: construct weak classifiers
  - Weak classifier: slightly better than random guessing (50% error)
  - Sequentially, construct weak classifiers, $G_m(x)$, on modified training data.
- Final classifier, combination of weak classifiers through a weighted majority vote

$$G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

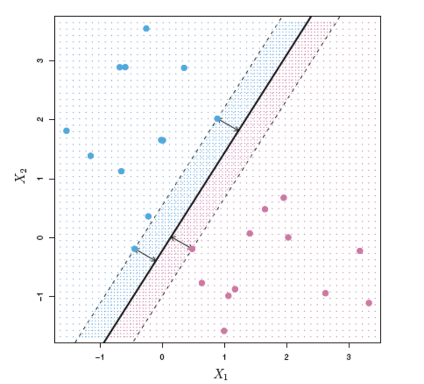
Boosting for Classification: AdaBoost

1. Set $w_i = 1/n, i = 1, 2, \ldots, n$, where n is the number of training points.
2. For $m = 1, \ldots, M$, repeat:
   (a) Fit a (weak) classifier $G_m(x)$ to the training data using wights w_i.
   (b) Compute the weighted error
   $$\mathsf{e}_m = \frac{\sum_{i=1}^n w_i 1_{\{y_i \neq G_m(x_i)\}}}{\sum_{i=1}^n w_i}$$
   (c) Compute
   $$\alpha_m = \log((1 - \mathsf{e}_m)/\mathsf{e}_m).$$
   (d) Update
   $$w_i \leftarrow w_i \exp(\alpha_m 1_{\{y_i \neq G_m(x_i)\}}), \quad , i = 1, 2, \ldots, n.$$
3. Final classifier
$$G(x) = \mathsf{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

Last lecture: The Maximal Margin Classifier

Optimal hyperplane

- *Margin*: Distance from an observation to the hyperplane
- *Maximal margin hyperplane*: One whose smallest margin is maximal
- *Support vectors*: points that support the maximal margin hyperplane



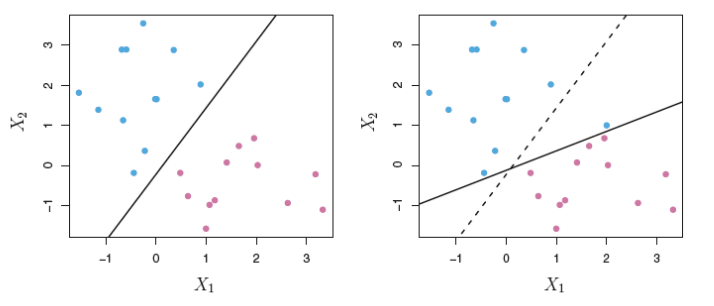
Problems

Non-separable case

- There is no hyperplane that separates the two classes

Highly sensitive to support vectors

- The hyperplane moves if one moves or introduces new support vector points



Need a "softer" separator

Last lecture: Support Vector Classifier

- Greater robustness to individual observations
- More general: Works for most points

The hyperplane is the solution to

$$\max_{\beta_j, \epsilon_j M} M \tag{1}$$

$$\text{subject to} \sum_{1}^{p} \beta_j^2 = 1 \tag{2}$$

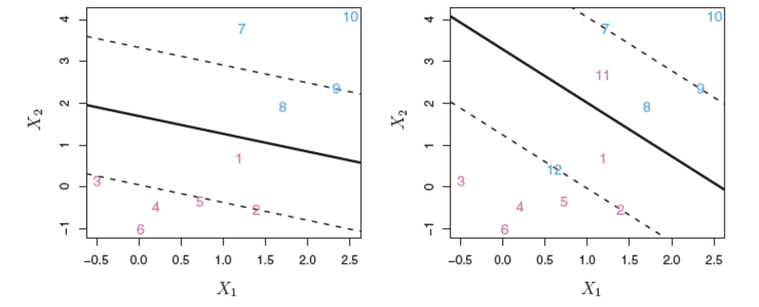
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \qquad \forall i \tag{3}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \tag{4}$$

Notes:

- ϵ_i - *slack variables*
  - $\epsilon_i = 0$ - i-th observation on the correct side of the margin
  - $0 < \epsilon_i < 1$ - i-th observation on the wrong side of the margin
  - $\epsilon_i > 1$ - i-th observation on the wrong side of the hyperplane
- C - budget for slackness

Example



- Left:
  - Purple observations: 3,4,5, and 6 = correct side of the margin; 2 is on the margin, and 1 is on the wrong side of the margin.
  - Blue observations: 7 and 10 = correct side of the margin; 9 is on the margin, and 8 is on the wrong side of the margin.
- Right: Same as left panel with two additional points, 11 and 12. 11 and 12 = wrong side of both the hyperplane and the margin.
- Robustness: the hyperplane on the right did not move much (!)

General Support Vector Machines

Nonlinear decision boundary - example

- p features: $X_1, X_2, \ldots, X_p$
- expand bases - $2p$ features: $X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$

Fit the hyperplane to the expanded bases

$$\max_{\beta_j, \epsilon_j M} M \tag{5}$$

$$\text{subject to} \sum_{1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1 \tag{6}$$

$$y_i(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2) \geq M(1 - \epsilon_i), \qquad \forall i \tag{7}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \tag{8}$$

- Nonlinear surface since these are quadratic expressions
  Quadratic can be replaced by polynomial of any degree
- The SVM explores further this idea using kernels

Understanding geometry: Distance from a hyperplane

Consider a hyperplane in p-dimensions

$$\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = 0$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ and $\boldsymbol{x} = (x_1, \ldots, x_p)$

- **Claim:** $\boldsymbol{\beta}$ is a perpendicular vector to this hyperplane

  Proof: Let $\boldsymbol{x}$ and $\boldsymbol{x}'$ be two points on this hyperplane. Then

$$\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle - (\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}' \rangle) = \langle \boldsymbol{\beta}, (\boldsymbol{x}' - \boldsymbol{x}) \rangle = 0$$

- Perpendicular unit vector

$$\frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}$$

  where $\|\boldsymbol{\beta}\| \equiv \|\boldsymbol{\beta}\|_2$ is the usual euclidian norm.

- Signed distance to the hyperplane: Let $\boldsymbol{x}_0$ be a point outside the hyperplane and $\boldsymbol{x}$ inside (note $\langle \boldsymbol{\beta}, \boldsymbol{x} \rangle = -\beta_0$)

$$\frac{\langle \boldsymbol{\beta}, (\boldsymbol{x}_0 - \boldsymbol{x}) \rangle}{\|\boldsymbol{\beta}\|} = \frac{\langle \boldsymbol{\beta}, \boldsymbol{x}_0 \rangle + \beta_0}{\|\boldsymbol{\beta}\|}$$

SVC Geometry

Using the preceding distance formula, we get

$$\max_{\beta_j, \epsilon_j M} M$$

$$\text{subject to} \quad y_i \frac{\langle \boldsymbol{\beta}, \boldsymbol{x} \rangle + \beta_0}{\|\boldsymbol{\beta}\|} \geq M(1 - \epsilon_i), \qquad \forall i$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$$

Recall that in the last lecture we set $\|\boldsymbol{\beta}\|^2 = \sum_1^p \beta_j^2 = 1$, in which case $\boldsymbol{\beta}$ is the unit normal vector.

- We can set $M = 1/\|\boldsymbol{\beta}\|$ in the preceding optimization

  Reason: If $(\beta_0, \boldsymbol{\beta})$ satisfies the preceding equations, then any scaled version of it satisfies, and in particular, we can set $M = 1/\|\boldsymbol{\beta}\|$, instead of $\|\boldsymbol{\beta}\| = 1$.

SVC: Convex Optimization

Next, optimizing $\max(1/\|\boldsymbol{\beta}\|)$ is equivalent to $\min(\|\boldsymbol{\beta}\|^2/2)$

$$\min_{\beta_j,\epsilon_j} \frac{\|\boldsymbol{\beta}\|^2}{2}$$

$$\text{subject to} \quad y_i(\langle \boldsymbol{\beta}, \boldsymbol{x} \rangle + \beta_0) \geq (1 - \epsilon_i), \qquad \forall i$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$$

Constrained Optimization: Crash Course

Primal problem

$$\begin{aligned} \min_{\boldsymbol{x} \in R^n} & f(\boldsymbol{x}) \\ \text{subject to} \quad & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{9}$$

Lagrangian

- Incorporate the constraints into one equation

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \mu_i h_i(\boldsymbol{x}) \tag{10}$$

Lagrange multiplier vectors or *dual vectors*

$$\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m), \qquad \boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$$

Constrained Optimization

Lagrange dual function

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) := \inf_{\boldsymbol{x} \in R^n} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{11}$$

$$= \inf_{\boldsymbol{x} \in R^n} \left(f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \mu_i h_i(\boldsymbol{x}) \right)$$

- Let $f^* =$ optimal value of the primal problem
- **Lower bound**: for any $\boldsymbol{\lambda} \geq 0$

$$f^* \geq g(\boldsymbol{\lambda}, \boldsymbol{\mu})$$

since, for any feasible point $\tilde{x}$ ($\tilde{x}$ satisfies the primal problem)

$$\sum_{i=1}^{m} \lambda_i f_i(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{p} \mu_i h_i(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{m} \lambda_i f_i(\tilde{\boldsymbol{x}}) \leq 0$$

since all $\lambda_i \geq 0$.

Constrained Optimization

Lagrange dual problem

$$\max g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \quad (12)$$
$$\text{subject to} \quad \boldsymbol{\lambda} \geq 0$$

- Let $g^* = g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be the optimal value of the dual problem
  $\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ are the optimal Lagrange multipliers
- Clearly
  $$f^* \geq g^*$$
- When are primal and dual problems equal, i.e., $f^* = g^*$?
- **Duality gap**
  $$f(\boldsymbol{x}) - g(\boldsymbol{\lambda}, \boldsymbol{\mu})$$
- Can be used for computation as a **stopping criterion**.

Constrained Optimization

Complementary slackness

- If primal and dual problem values are equal, $f^* = g^*$, and attained at values $f^* = f(\boldsymbol{x}^*)$ and $g^* = g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, then

$$\lambda_i^* f_i(\boldsymbol{x}^*) = 0, \qquad i = 1, \ldots, m$$

- **Proof**

$$\begin{aligned} f^* = f(\boldsymbol{x}^*) = g^* = g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) & \\ &= \inf_{\boldsymbol{x} \in R^n} \left(f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \mu_i h_i^*(\boldsymbol{x}) \right) \\ &\leq f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{x}^*) + \sum_{i=1}^{p} \mu_i h_i^*(\boldsymbol{x}^*) \\ &\leq f(\boldsymbol{x}^*) \end{aligned}$$

since $\lambda_i^* \geq 0, f_i(\boldsymbol{x}^*) \leq 0$ and $h_i^*(\boldsymbol{x}^*) = 0$.
Hence, $\sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{x}) = 0$, and therefore $\lambda_i^* f_i(\boldsymbol{x}^*) = 0$.

Karush-Kuhn-Tucker (KKT) Conditions

Necessary conditions for nonconvex problems

- Let $\boldsymbol{x}^*$ and $\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ be any primal and dual points with zero duality gap.
- Then, the following KKT conditions hold

$$
\begin{aligned}
f_i(\boldsymbol{x}^*) &\leq 0, && \text{(primal feasibility)}\\
h_i(\boldsymbol{x}^*) &= 0, && \text{(primal feasibility)}\\
\lambda_i^* &\geq 0, && \text{(dual feasibility)}\\
\lambda_i^* f_i(\boldsymbol{x}^*) &= 0, && \text{(complem. slackness)}\\
\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(\boldsymbol{x}^*) + \sum_{i=1}^{p} \mu_i^* \nabla h_i(\boldsymbol{x}^*) &= 0, && \text{(stationarity)}
\end{aligned}
$$

Recall $\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \cdots, \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right)$ is the gradient

- 4th condition = complementary slackness $\Rightarrow$
  $\lambda_i = 0, f_i(\boldsymbol{x}^*) < 0$: $\boldsymbol{x}^*$ is **inside**, or
  $\lambda_i > 0, f_i(\boldsymbol{x}^*) = 0$: $\boldsymbol{x}^*$ on the **boundary**

Karush-Kuhn-Tucker (KKT) Conditions

Sufficient conditions for convex problems

- Assume f, f_i **are convex, and** h_i **are affine (linear)**
- Let $\tilde{\boldsymbol{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$ be any points that satisfy KKT conditions
- Then, $\tilde{\boldsymbol{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$ are
  primal and dual optimal with zero duality gap
- **Proof:**

$$\begin{aligned} g(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}) &= L(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}) \\ &= f(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{m} \tilde{\lambda}_i f_i(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{p} \tilde{\mu}_i h_i(\tilde{\boldsymbol{x}}) \\ &= f(\tilde{\boldsymbol{x}}) \end{aligned}$$

  We used in the second equality that $L(\boldsymbol{x}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$ is convex in $\boldsymbol{x}$ since $\tilde{\lambda}_i \geq 0$ and h_i are affine, implying, by the last KKT condition, that $\tilde{\boldsymbol{x}}$ minimizes $L(\boldsymbol{x}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$.
  In the last line, we used $h_i(\boldsymbol{x}^*) = 0$ and $\lambda_i^* f_i(\boldsymbol{x}^*) = 0$ (complementary slackness)

SVC Continued: Dual Problem

Incorporate the SVC constraints into the Lagrange function

$$L(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{\|\boldsymbol{\beta}\|^2}{2} + c\sum_{i=1}^{n} \epsilon_i - \sum_{i=1}^{n} \alpha_i [y_i(\langle \boldsymbol{\beta}, \boldsymbol{x}_i \rangle + \beta_0) - (1 - \epsilon_i)] - \sum_{i=1}^{n} \mu_i \epsilon_i$$

Next, we define the **dual function**

$$g(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \inf_{\boldsymbol{\beta}, \beta_0, \epsilon_i} L(\boldsymbol{\beta}, \beta_0, c, \boldsymbol{\alpha}, \boldsymbol{\mu})$$

Then, since $L(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}, \boldsymbol{\mu})$ is quadratic in β_i, we can explicitly compute its derivatives with respect to β_i and ϵ_i

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \qquad (\partial/\partial \boldsymbol{\beta})$$

$$0 = \sum_{i=1}^{n} \alpha_i y_i \qquad (\partial/\partial \beta_0)$$

$$\alpha_i = c - \mu_i \qquad (\partial/\partial \epsilon_i)$$

Plugging the preceding derivatives into L, yields an explicit formula for g, after some algebra.

SVC: Dual Problem

Now, dual problem is to maximize $g(\boldsymbol{\alpha}, \boldsymbol{\mu})$, i.e.,

$$\max_{\alpha_i} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \right)$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq c, \sum \alpha_i y_i = 0$$

KKT conditions to the rescue

$$\begin{aligned} \alpha_i [y_i(\langle \boldsymbol{\beta}, \boldsymbol{x}_i \rangle + \beta_0) - (1 - \epsilon_i)] &= 0 \qquad \textbf{slackness} \\ \mu_i \epsilon_i &= 0 \\ y_i(\langle \boldsymbol{\beta}, \boldsymbol{x}_i \rangle + \beta_0) - (1 - \epsilon_i) &\geq 0 \end{aligned}$$

plus the preceding derivative equations.
From **slackness, $\alpha_i > 0$ only for support vectors**, when

$$y_i(\langle \boldsymbol{\beta}, \boldsymbol{x}_i \rangle + \beta_0) = (1 - \epsilon_i)$$

SVC Simplification

From the derivative condition (2 slides ago)

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \\ &= \sum_{i \in \mathcal{S}} \alpha_i y_i \boldsymbol{x}_i\end{aligned}$$

And, thus, the classification hyperplane is easy to compute

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle$$

Support Vector Machines: Hilbert spaces

Hilbert spaces: Generalized linear spaces

- Let $\phi(\boldsymbol{x}_i)$ be a transformation of feature variables such that
- **Kernel** - $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive definite
- Generalized inner product:

$$\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Hence, with arbitrary nonlinear transformation, Kernel, we can repeat the preceding optimization, and derive an SVC.

The resulting classifier is known as **SVM**, with the decision function taking the following nonlinear form in general

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

Support Vector Machines: Kernels

Kernels:

- *Linear*

$$K(x_i, x_j) = \langle x_j, x_i \rangle$$

- *Polynomial* - for positive integer d

$$K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^d$$

- *Radial* - for $\gamma > 0$ - (or Gaussian $\gamma = 1/(2\sigma^2)$)

$$K(x_i, x_j) = \exp\left(-\gamma \sum_{k=1}^{p} (x_{ik} - x_{jk})^2\right)$$

Deriving The Quadratic Kernel

Kernels:

- Consider data with two predictors: $x_i = (x_{i1}, x_{i2})$
- *Quadratic Kernel* $(d = 2)$: $K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^2$
- We can obtain the preceding kernel by considering feature map

$$\phi(x_i) = (1, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2)$$

- Then, the inner product

$$\begin{aligned} \langle \phi(x_i), \phi(x_j) \rangle &= 1 + 2x_{i1}x_{j1} + 2x_{i1}x_{j1} + x_{i1}^2 x_{j1}^2 \\ &\quad + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2 x_{j2}^2 \\ &= 1 + 2\left\langle x_i, x_j \right\rangle + \left\langle x_i, x_j \right\rangle^2 \\ &= (1 + \langle x_j, x_i \rangle)^2 = K(x_i, x_j) \end{aligned}$$

Finding Feature Maps Is Hard

In general, finding feature maps for a corresponding kernel is had.
Example: Radial Kernel, $x \in \mathbb{R}$

- Consider

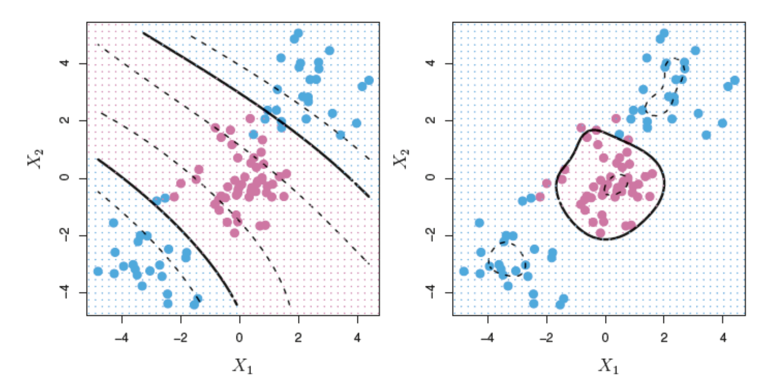
$$\phi(x) = (1, \frac{xe^{-x^2/2}}{\sqrt{1!}}, \frac{x^2e^{-x^2/2}}{\sqrt{2!}}, \cdots, \frac{x^ne^{-x^2/2}}{\sqrt{n!}}, \cdots)$$

- Then

$$\begin{aligned}\langle \phi(x_i), \phi(x_j) \rangle &= \sum_{n=0}^{\infty} \frac{x_i^n e^{-x_i^2/2}}{\sqrt{n!}} \frac{x_j^n e^{-x_j^2/2}}{\sqrt{n!}} \\ &= e^{-x_i^2/2 - x_j^2/2} \sum_{n=0}^{\infty} \frac{x_i^n x_j^n}{n!} = e^{-\|x_i - x_j\|/2} = K(x_i, x_j)\end{aligned}$$

Fortunately, we can use Kernels without knowing the feature maps.
We'll talk about this after the midterm.

Example



- Left: An SVM with a polynomial kernel of degree 3
- Right: An SVM with a radial kernel
- Either kernel is capable of capturing the decision boundary

SVM with More than Two Classes

One-Versus-One classification

- For $K > 2$ classes, consider $\binom{K}{2}$ pairs
- For example, we might compare the kth class, coded as $+1$, to the k'th class, coded as -1
- We tally the number of times that the test observation is assigned to each of the K classes
- The final classification is performed by assigning the test observation to the class to which it was most frequently assigned in these $\binom{K}{2}$ pairwise classifications

SVM with More than Two Classes

One-Versus-All classification

- We fit K SVMs, each time comparing one of the K classes to the remaining $K-1$ classes.
- Let $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the kth class (coded as $+1$) to the others (coded as -1).
- Let $\boldsymbol{x}_0$ denote a test observation
- We assign $\boldsymbol{x}^*$ to the class with maximum

$$\beta_{0k} + \beta_{1k}x_1^* + \cdots + \beta_{pk}x_p^*$$

Khan - Gene Expression Data

- Gene expression measurements for four cancer types of small round blue cell tumors.
- For each tissue sample, 2308 gene expression measurements are available.

```
> library(ISLR)
> names(Khan)
[1]   "xtrain"   "xtest"   "ytrain"   "ytest"
> dim(Khan$xtrain)
[1]   63  2308
> dim(Khan$xtest)
[1]   20  2308
> length(Khan$ytrain)
[1]  63
> length(Khan$ytest)
[1]  20
```

Khan - Gene Expression Data

- The training and test sets consist of 63 and 20 observations respectively
- Belong to 4 cancer types

```
> table(Khan$ytrain)
 1  2  3  4
 8 23 12 20
> table(Khan$ytest)
1 2 3 4
3 6 6 5
```

Khan - SVM Classification

Perfect fit with linear kernels (!) - no training errors.
Is this a surprise?

```
> dat=data.frame(x=Khan$xtrain, y=as.factor(Khan$ytrain))
> out=svm(y~., data=dat, kernel="linear",cost=10)
> summary(out)
Call:
svm(formula = y ~ ., data = dat, kernel = "linear",
    cost = 10)
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.000433
Number of Support Vectors:  58
 ( 20 20 11 7 )
Number of Classes:  4
Levels:
 1 2 3 4
> table(out$fitted, dat$y)

     1  2  3  4
  1  8  0  0  0
  2  0 23  0  0
  3  0  0 12  0
  4  0  0  0 20
```

Khan - SVM Test Performance

Two testing errors

```
> dat.te=data.frame(x=Khan$xtest, y=as.factor(Khan$ytest))
> pred.te=predict(out, newdata=dat.te)
> table(pred.te, dat.te$y)

pred.te  1  2  3  4
      1  3  0  0  0
      2  0  6  2  0
      3  0  0  4  0
      4  0  0  0  5
```

Reading on Support Vector Machines

ISL: Chapter 9. In particular, read Section 9.6 on experiments in R, including: e1071 library, "Khan" gene expression data (and ROC curves)

ESL: Chapter 12

Paper: Vladimir N. Vapnik (1999), "An Overview of Statistical Learning"

Homework: Start working on the final project.

Optional reading: Optimization - book

Convex Optimization, Stephen Boyd and Lieven Vandenberghe, Cambridge University Press, 2004
Free download:
`http://stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`

Today's presentation can be found in Chapter 5.