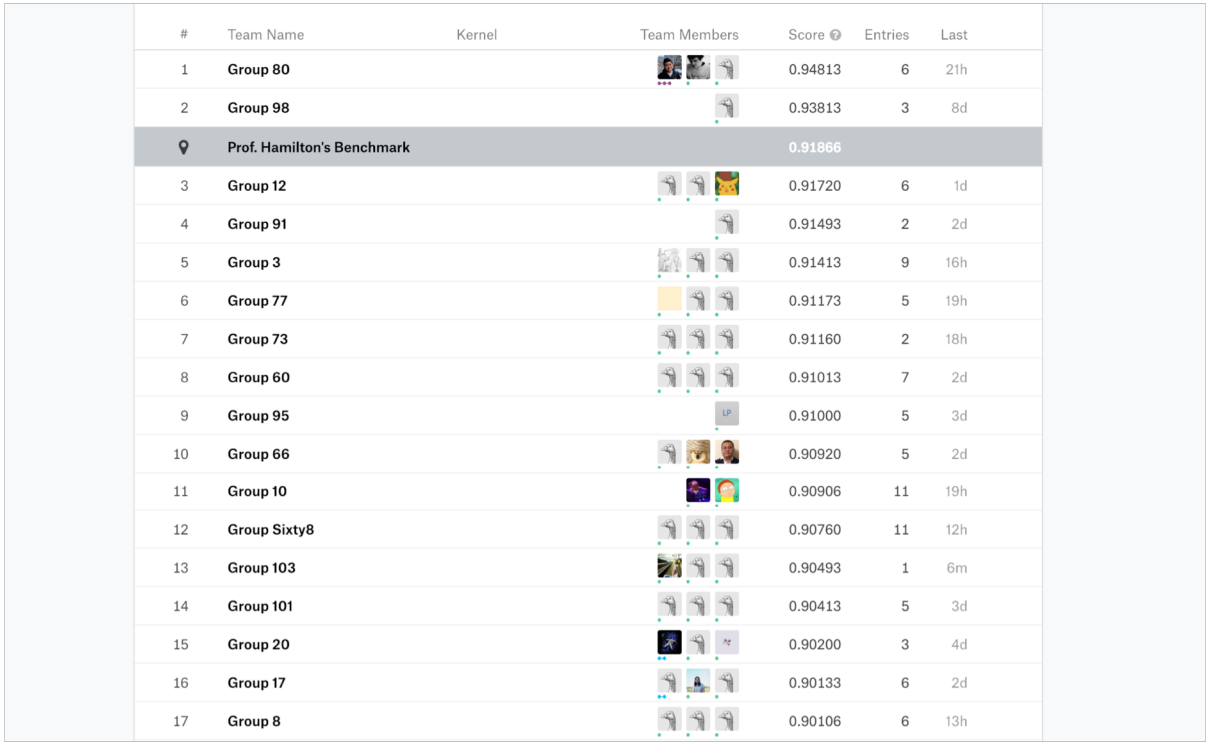
COMP 551 - Applied Machine Learning Lecture 11 — SVMs continued

William L. Hamilton

(with slides and content from Joelle Pineau)

MiniProject 2



#	Team Name	Kernel	Team Members	Score	Entries	Last
1	Group 80			0.94813	6	21h
2	Group 98			0.93813	3	8d
	Prof. Hamilton's Benchmark			0.91866		
3	Group 12			0.91720	6	1d
4	Group 91			0.91493	2	2d
5	Group 3			0.91413	9	16h
6	Group 77			0.91173	5	19h
7	Group 73			0.91160	2	18h
8	Group 60			0.91013	7	2d
9	Group 95			0.91000	5	3d
10	Group 66			0.90920	5	2d
11	Group 10			0.90906	11	19h
12	Group Sixty8			0.90760	11	12h
13	Group 103			0.90493	1	6m
14	Group 101			0.90413	5	3d
15	Group 20			0.90200	3	4d
16	Group 17			0.90133	6	2d
17	Group 8			0.90106	6	13h

How to write a good project report?

- Read some good ML papers ☺
  - ACL and EMNLP publish short papers (4-5 pages).
  - Workshops are also good and have short papers (4-5 pages)
- Som examples:
  - http://aclweb.org/anthology/D18-1546
  - http://petar-v.com/dgi_nips18_camera.pdf
  - http://emnlp2014.org/papers/pdf/EMNLP2014024.pdf
  - https://arxiv.org/pdf/1811.05868.pdf
  - http://www.aclweb.org/anthology/D13-1148
  - https://arxiv.org/pdf/1704.04008.pdf

SVMs (Review)

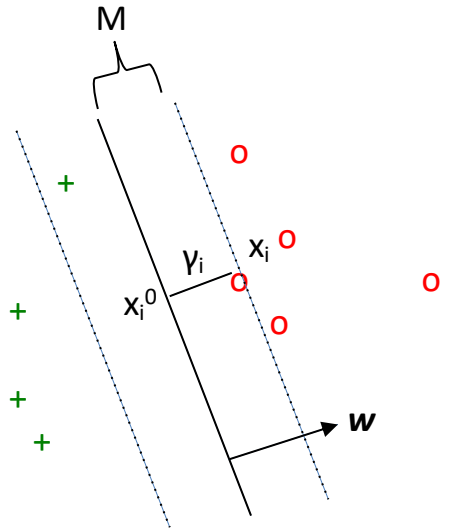
- First suggestion:

  Maximize M

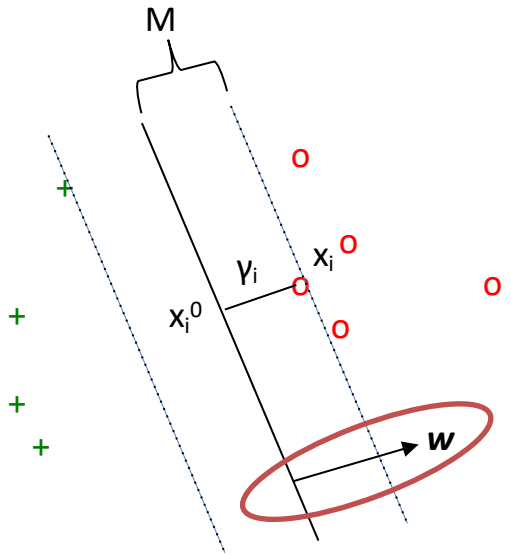
  with respect to $\mathbf{w}$

  subject to $y_i \mathbf{w}^T \mathbf{x}_i / ||\mathbf{w}|| \geq \mathrm{M}, \forall i$

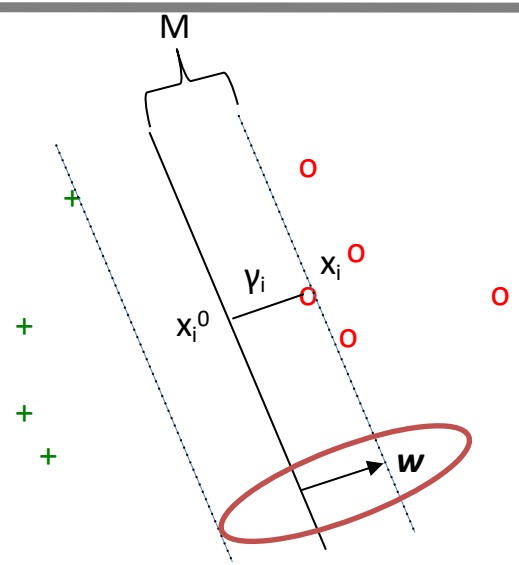
- This is not very convenient for optimization:
  - $\mathbf{w}$ appears nonlinearly in the constraints.
  - Problem is underconstrained. If $(\mathbf{w}, \mathrm{M})$ is optimal, so is $(\beta\mathbf{w}, \mathrm{M})$, for any $\beta > 0$.
  - Add a constraint: $||\mathbf{w}|| = 1/\mathrm{M}$
    - Intuition: Changing $||\mathbf{w}||$ doesn't change the decision hyperplane, so without loss of generality we say that $y_i \mathbf{w}^T \mathbf{x}_i = 1$ for points that are right on the margin.



SVMs (Review)



$\mathbf{w}$ specifies the orientation of the decision boundary, but $||\mathbf{w}||$ is unconstrained and unrelated to M .

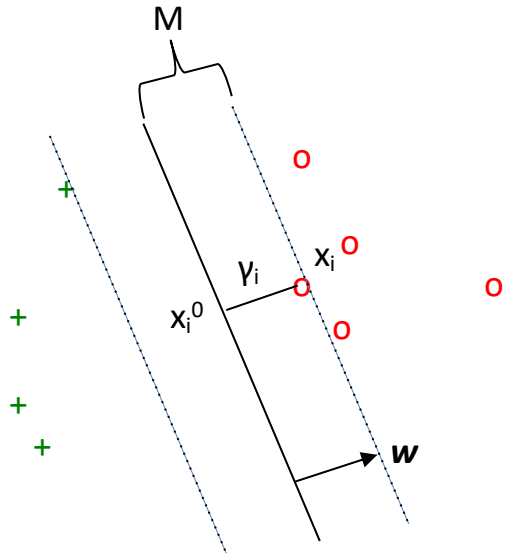


$||\mathbf{w}||=1/\mathrm{M}$ so $\mathbf{w}$ specifies both the orientation of the decision boundary and the size of the margin.

SVMs (review)

- Revised formulation: Min $\frac{1}{2}\,||\mathbf{w}||^2$
  w.r.t. $\mathbf{w}$
  s.t. $y_i \mathbf{w}^T \mathbf{x}_i \geq 1, \forall i$

- Note: we set $||\mathbf{w}|| = 1/M$, so minimizing $||\mathbf{w}||^2$ is equivalent to maximizing M^2, which is equivalent to maximizing M.



Lagrangian optimization

- Consider $P(\mathbf{w}) = \max_{\boldsymbol{\alpha}:\alpha i \geq 0} L(\mathbf{w},\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}:\alpha i \geq 0} f(\mathbf{w}) + \sum_{i=1:k} \alpha_i g_i(\mathbf{w})$
  - (P stands for "primal")
- Recall: $L(\mathbf{w}, \boldsymbol{\alpha}) =$ Observe that the following is true:

$$P(\mathbf{w}) = \{ \quad f(\mathbf{w}), \quad \text{if all constraints are satisfied,}$$
$$+\infty, \quad \text{otherwise } \}$$

- Hence, instead of computing $\min_{\mathbf{w}} f(\mathbf{w})$ subject to the original constraints, we can compute:

$$p^* = \min_{\mathbf{w}} P(\mathbf{w}) = \min_{\mathbf{w}} \max_{\boldsymbol{\alpha}:\alpha i \geq 0} L(\mathbf{w},\boldsymbol{\alpha})$$ *Primal*

- Alternately, invert max and min to get:

$$d^* = \max_{\boldsymbol{\alpha}:\alpha i \geq 0} \min_{\mathbf{w}} L(\mathbf{w},\boldsymbol{\alpha})$$ *Dual*

SVMs (review)

- We wanted to solve:

$$\begin{aligned} &\text{Min} && \tfrac{1}{2}\,||\mathbf{w}||^2 \\ &\text{w.r.t.} && \mathbf{w} \\ &\text{s.t.} && y_i \mathbf{w}^T \mathbf{x}_i \geq 1 \end{aligned}$$

- The Lagrangian:

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \tfrac{1}{2}\,||\mathbf{w}||^2 + \sum_i \alpha_i \,(1 - y_i\,(\mathbf{w}^T \mathbf{x}_i)\,)$$

$$\max_{\boldsymbol{\alpha}:\alpha i \geq 0} L(\mathbf{w},\boldsymbol{\alpha}) \;=\; \{\; \tfrac{1}{2}\,||\mathbf{w}||^2 \text{ if all constraints are satisfied, } +\infty \text{ otherwise } \}$$

- The primal problem is: $\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}:\alpha i \geq 0} L(\mathbf{w},\boldsymbol{\alpha})$
- The dual problem is: $\max_{\boldsymbol{\alpha}:\alpha i \geq 0} \min_{w} L(\mathbf{w},\boldsymbol{\alpha})$

SVMs (review)

- Consider both solutions:

$$p^* = \min_{\mathbf{w}} \max_{\alpha:\alpha i \geq 0} L(\mathbf{w},\alpha) \qquad \textit{Primal}$$
$$d^* = \max_{\alpha:\alpha i \geq 0} \min_{\mathbf{w}} L(\mathbf{w},\alpha) \qquad \textit{Dual}$$

- If f and g_i are convex and the g_i can all be satisfied simultaneously for some $\mathbf{w}$, then we have equality: $d^* = p^* = L(\mathbf{w}^*, \boldsymbol{\alpha}^*)$.
  - $\mathbf{w}^*$ is the optimal weight vector (= primal solution)
  - $\boldsymbol{\alpha}^*$ is the optimal set of support vectors (= dual solution)
  - For SVMs, we have a quadratic objective and linear constraints so both f and g_i are convex.
  - For linearly separable data, all g_i can be satisfied simultaneously.
  - Note: $\mathbf{w}^*, \boldsymbol{\alpha}^*$ solve the primal and dual if and only if they satisfy the Karush-Kunh-Tucker conditions **(see Section 7.1 of the Bishop book).**

SVMs (review)

- Taking derivatives of $\mathbf{L(w, \alpha)}$ wrt $\mathbf{w}$, setting to 0, and solving for $\mathbf{w}$:

$$\mathbf{L(w, \alpha)} = \tfrac{1}{2} ||\mathbf{w}||^2 + \sum_i \alpha_i \left(1 - y_i \left(\mathbf{w}^T \mathbf{x}_i\right)\right)$$

$$\delta \mathbf{L} / \delta \mathbf{w} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$

$$\mathbf{w^*} = \sum_i \alpha_i y_i \mathbf{x}_i$$

- The optimal solution $\mathbf{w^*}$ is a linear combination of the $\mathbf{x}_i$.
- Plugging this back into $\mathbf{L}$ we get the dual: $\max_\alpha \sum_i \alpha_i - \tfrac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x})$ with constraints $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$. (Quadratic programming problem).

SVMs (review)

- Suppose we find the optimal α 's (e.g. using a QP package.)
- Constraint i is active when $\alpha_i > 0$. This corresponds for the points for which $(1\text{-}y_i \mathbf{w}^T \mathbf{x}_i)=0$.
- These are the points lying on the edge of the margin. We call them support vectors. They define the decision boundary.

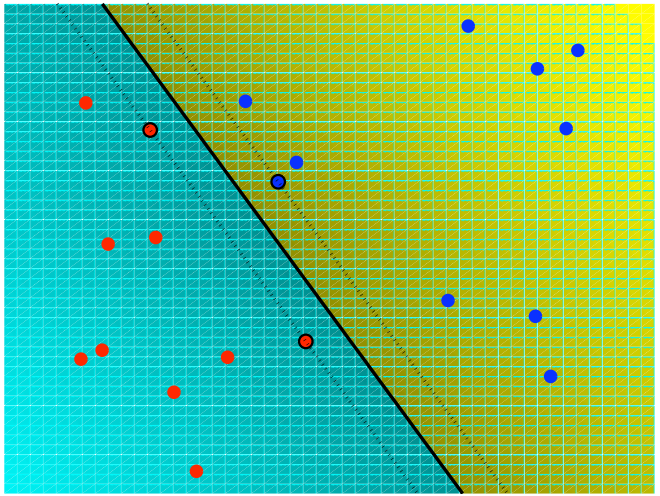
- The output of the classifier for query point $\mathbf{x}$ is computed as:

$$h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\sum_{i=1:n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}))$$

  - I.e., it is determined by computing the dot product of the query point with the support vectors.

SVMs (review)

- See pages 325-343 in the Bishop textbook for a detailed review of these derivations.
- You should know this material, but you are not responsible for anything in the Appendix.



Support vectors are in bold

Non-linearly separable data

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

- Option 1: Relax the constraints and allow some points to be misclassified by the margin.

- Option 2: Allow a nonlinear decision boundary in the input space by finding a linear decision boundary in an expanded space (*similar to adding polynomial terms in linear regression*.)
  - Here x_i is replaced by $\phi(x_i)$, where ϕ is called a feature mapping.

Soften the primal objective

- We wanted to solve:

$$\min_{\mathbf{w}} \quad \tfrac{1}{2}\,||\mathbf{w}||^2$$
$$\text{s.t.} \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

- This can be re-written:

$$\min_{\mathbf{w}} \quad \sum_i L_{0\text{-}\infty}\,(w^T x_i, y_i) \;+\; \tfrac{1}{2}\,||\mathbf{w}||^2$$
$$\text{s.t.} \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

where $\sum_i L_{0\text{-}\infty}\,(w^T x_i, y_i) = (\infty$ for a misclassification, 0 correct classification)

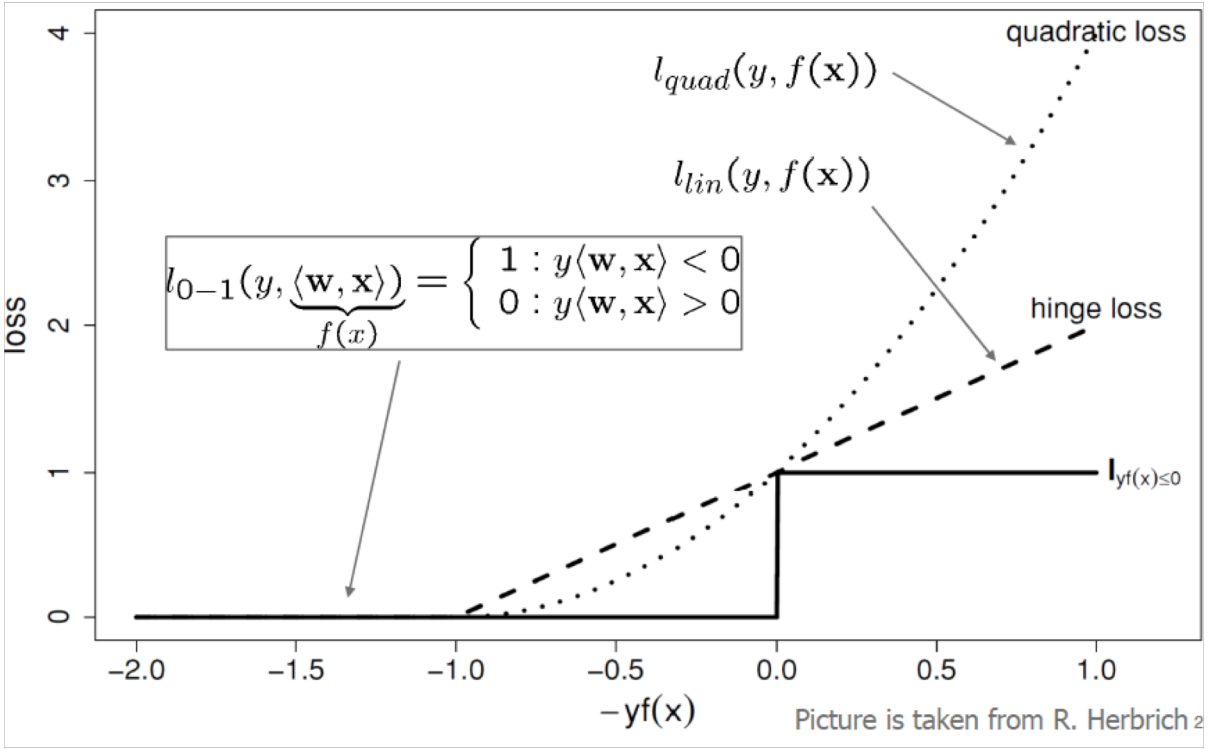
- Soften misclassification cost:

$$\min_{\mathbf{w}} \quad \sum_i L_{0\text{-}1}\,(w^T x_i, y_i) + \tfrac{1}{2}\,||\mathbf{w}||^2$$
$$\text{s.t.} \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

where $\sum_i L_{0\text{-}1}\,(w^T x_i, y_i) = (1$ for a misclassification, 0 correct classification)

- But this is a **non-convex** objective!

Approximation of the $L_{0\text{-}1}$ function

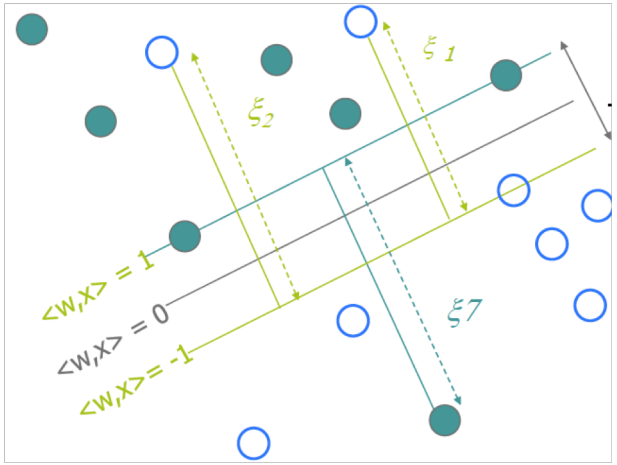


SVM with hinge loss

- Hinge loss: $L_{hin}\,(w^T x_i, y_i) = \max\{1\text{-}y_i \mathbf{w}^T \mathbf{x}_i\,, 0\}$

- Soften misclassification cost: $\min_{\mathbf{w}} \quad C \sum_i L_{hin}\,(w^T x_i, y_i) + \frac{1}{2}\,||\mathbf{w}||^2$
  where C controls trade-off between slack penalty and margin.

- The hinge loss upper-bounds the 0-1 loss.
  $\xi_i \geq 1\text{-}\, y_i \mathbf{w}^T \mathbf{x}_i \geq L_{0\text{-}1}\,(w^T x_i, y_i)$



Primal Soft SVM problem

- Define slack variables $\xi_i = L_{hin}(w^T x_i, y_i) = \max\{1 - y_i \mathbf{w}^T \mathbf{x}_i, 0\}$

- Solve: $\hat{\mathbf{w}}_{soft} = \text{argmin}_{\mathbf{w},\xi}\ C \sum_{i:1:n} \xi_i + \frac{1}{2}||\mathbf{w}||^2$

$$\text{s. t.} \qquad y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i, \qquad i = 1, \ldots, n$$

$$\xi_i \geq 0, \qquad i = 1, \ldots, n$$

$$\text{where} \qquad w \in R^m, \xi \in R^n$$

Primal Soft SVM problem

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

- Solve: $\hat{w}_{soft} = \text{argmin}_{w,\xi}\ C \sum_{i:1:n} \xi_i + \frac{1}{2}||w||^2$ Add Lagrange mult:

  s. t. $y_i w^T x_i \geq 1 - \xi_i$, $i = 1, ..., n$ <= Call this α_i

  $\xi_i \geq 0$, $i = 1, ..., n$ <= Call this β_i

  where $w \in R^m, \xi \in R^n$

- 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: Adding Lagrange multipliers

- Primal objective: $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\min_{\mathbf{w},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

  where $L(w, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i \mathbf{w}^T \mathbf{x}_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i$

Soft SVM: Adding Lagrange multipliers

- Primal objective: $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\min_{\mathbf{w},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

  where $L(w, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i \mathbf{w}^T \mathbf{x}_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i$

- Dual (invert min and max): $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \min_{\mathbf{w},\boldsymbol{\xi}} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

Soft SVM: Adding Lagrange multipliers

- Primal objective: $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\min_{w,\xi} \max_{\alpha,\beta} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

  where $L(w, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i \mathbf{w}^T \mathbf{x}_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i$

- Dual (invert min and max): $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\max_{\alpha,\beta} \min_{w,\xi} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$
- Solve: $\delta L / \delta \mathbf{w} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$
- $\delta L / \delta \boldsymbol{\xi} = C 1_n - \boldsymbol{\alpha} - \boldsymbol{\beta} = 0 \Rightarrow \boldsymbol{\beta} = C 1_n - \boldsymbol{\alpha}$
- Lagrange multipliers are positive, so we have: $0 \leq \beta_i$, $0 \leq \alpha_i \leq C$

Soft SVM: Adding Lagrange multipliers

- Primal objective: $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\min_{\mathbf{w},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

  where $L(w, \xi, \alpha, \beta) = ½||w||^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i \mathbf{w}^T \mathbf{x}_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i$

- Dual (invert min and max): $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \min_{\mathbf{w},\boldsymbol{\xi}} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$
- Solve: $\delta L/\delta \mathbf{w} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$
- $\delta L/\delta \boldsymbol{\xi} = C1_n - \boldsymbol{\alpha} - \boldsymbol{\beta} = 0 \Rightarrow \boldsymbol{\beta} = C1_n - \boldsymbol{\alpha}$
- Lagrange multipliers are positive, so we have: $0 \leq \beta_i$, $0 \leq \alpha_i \leq C$

- Plug into dual : $\max_\alpha \sum_i \alpha_i - ½ \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x})$

  with constraints $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0$.
- This is a quadratic programming problem (similar to Hard SVM).

Soft SVM: Adding Lagrange multipliers

- Primal objective: $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\min_{\mathbf{w},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

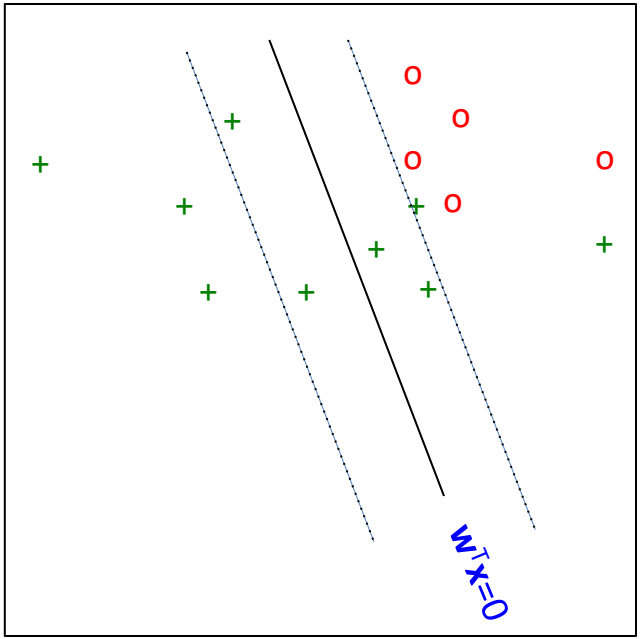
  where $L(w, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i \mathbf{w}^T \mathbf{x}_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i$

- Dual (invert min and max): $(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \min_{\mathbf{w},\boldsymbol{\xi}} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

See Section 7.1.1 in the Bishop book for detailed derivations.

Soft SVM solution

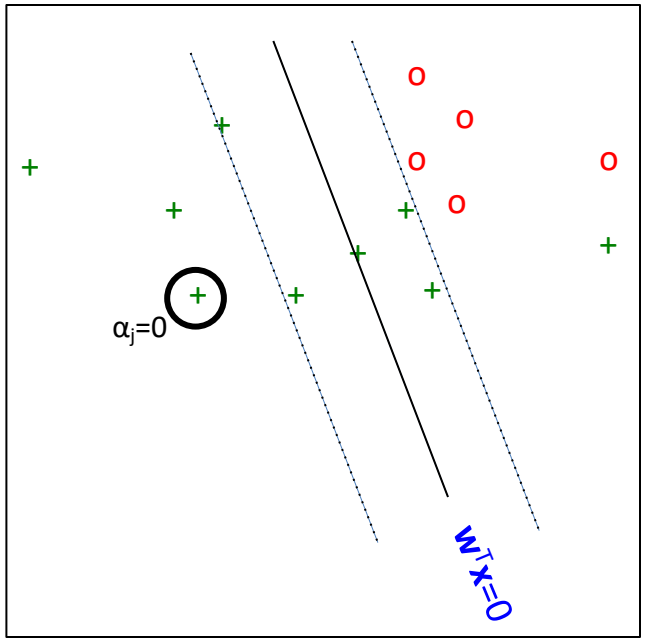
- Soft-SVM has one more constraint $0 \leq a_i \leq C$ (vs $0 \leq a_i$ in Hard SVM).
- When $C \to \infty$, then Soft-SVM=>Hard-SVM.



Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM=>Hard-SVM.

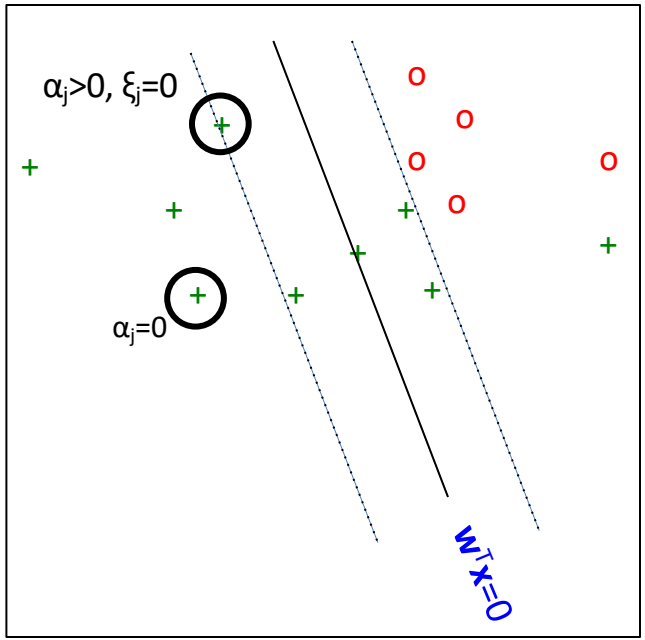
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM=>Hard-SVM.

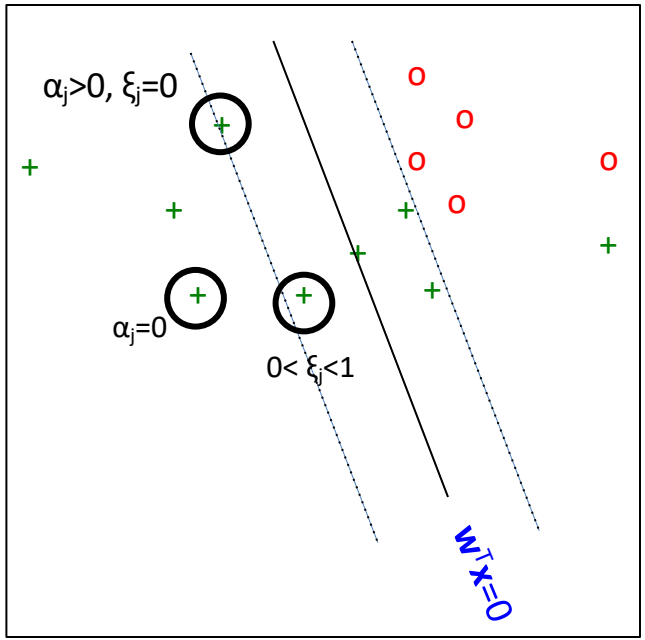
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM=>Hard-SVM.

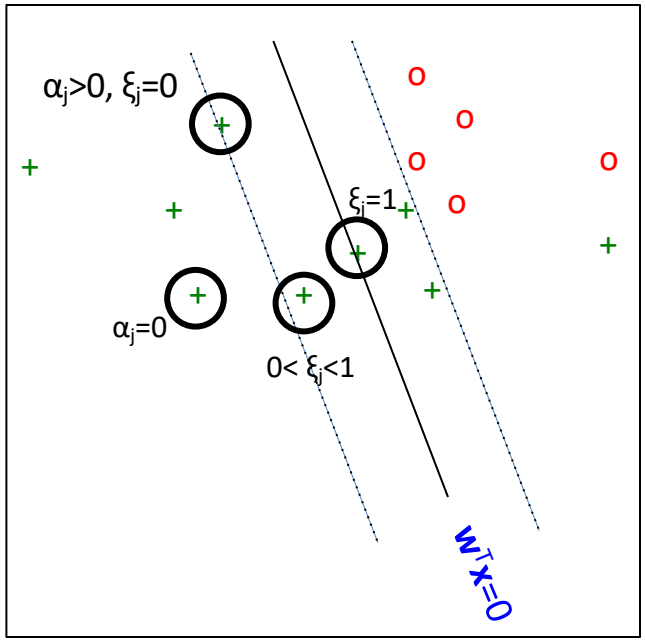
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

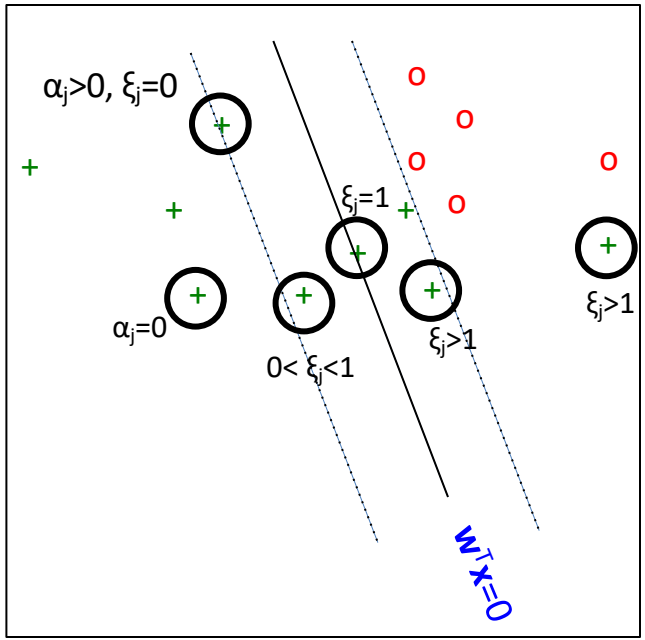
- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C=>\infty$, then Soft-SVM=>Hard-SVM.

- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i=0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM=>Hard-SVM.



- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.

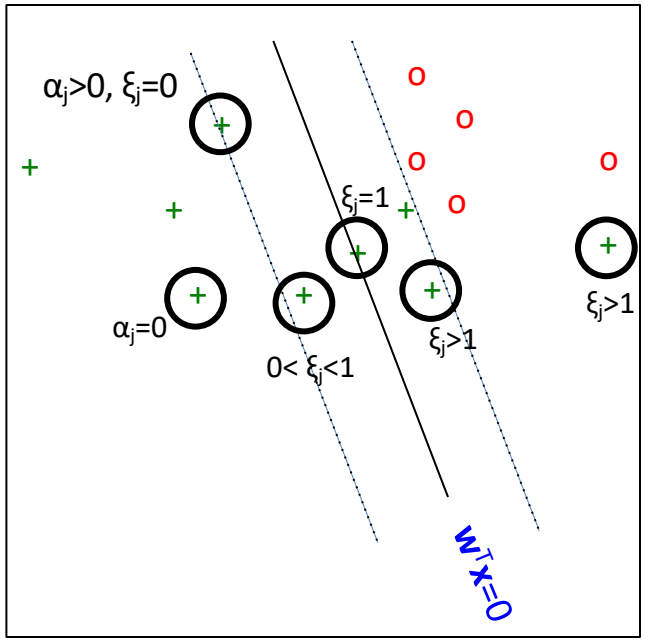
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM=>Hard-SVM.

- To predict on test data:

$$h_{\mathbf{w}}(\mathbf{x}) = \text{sign}\left(\sum_{i=1:n} \alpha_i y_i \, (\mathbf{x}_i \cdot \mathbf{x}) \right)$$

- 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.

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.

Non-linearly separable data

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

- Option 1: Relax the constraints and allow some points to be misclassified by the margin.

- Option 2: Allow a nonlinear decision boundary in the input space by finding a linear decision boundary in an expanded space (*similar to adding polynomial terms in linear regression*.)
  - Here x_i is replaced by $\phi(x_i)$, where ϕ is called a feature mapping.

Margin optimization in feature space

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

  - Primal form: $\text{Min} \quad \frac{1}{2} ||\mathbf{w}||^2$

    w.r.t. $\mathbf{w}$

    s.t. $y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1$

Margin optimization in feature space

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

  - Primal form: $\text{Min} \quad \frac{1}{2}\,||\mathbf{w}||^2$

    w.r.t. $\mathbf{w}$

    s.t. $y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1$

  - Dual form: $\text{Max} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}))$

    w.r.t. α_i

    s.t. $\alpha_i \geq 0$

    $\sum_i \alpha_i y_i = 0$

Feature space solution

- The optimal weights, in the expended feature space, are

$$\mathbf{w} = \sum_{i=1:n} \alpha_i y_i \phi(\mathbf{x}_i)$$

- Classification of an input $\mathbf{x}$ is given by:

$$h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\sum_{i=1:n} \alpha_i y_i (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})))$$

Feature space solution

- The optimal weights, in the expended feature space, are

$$\mathbf{w} = \sum_{i=1:n} \alpha_i y_i \phi(\mathbf{x}_i)$$

- Classification of an input $\mathbf{x}$ is given by:

$$h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\sum_{i=1:n} \alpha_i y_i (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})))$$

- 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 $K: R^m \times R^m \rightarrow R$, which corresponds to a dot product for some feature mapping ϕ:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) \text{ for some } \phi$$

- Conversely, by choosing feature mapping ϕ, we implicitly choose a kernel function.
- Recall that $\phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) = \cos\angle(\mathbf{x}_1, \mathbf{x}_2)$, 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 $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$.
- Is this a kernel?

$$
\begin{aligned}
K(\mathbf{x}, \mathbf{z}) &= \left(\sum_{i=1:m} x_i z_i \right) \left(\sum_{j=1:m} x_j z_j \right) \\
&= \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j \\
&= \sum_{i,j \in \{1..m\}} \left(x_i x_j \right) \left(z_i z_j \right)
\end{aligned}
$$

Example: Quadratic kernel

- Let $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$.
- Is this a kernel?

$$K(\mathbf{x}, \mathbf{z}) = \left(\sum_{i=1:m} x_i z_i\right)\left(\sum_{j=1:m} x_j z_j\right)$$
$$= \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j$$
$$= \sum_{i,j \in \{1..m\}} (x_i x_j)(z_i z_j)$$

- We see it is a kernel, with feature mapping:

$$\phi(\mathbf{x}) = < x_1^2, x_1 x_2, \ldots, x_1 x_m, x_2 x_1, x_2^2, \ldots, x_m^2 >$$

Feature vector includes all squares of elements and all cross terms.

Important: Computing ϕ takes $O(m^2)$ but computing K only takes $O(m)$.

Polynomial kernels

- More generally, $\mathbf{K(x, z) = (x \cdot z)^d}$ is a kernel, for any positive integer d:

$$\mathbf{K(x, z)} = \left(\sum_{i=1:m} x_i z_i \right)^d$$

- If we expanded the sum above in the naïve way, we get $\mathbf{O(m^d)}$ terms.
  - See Multinomial Theorem: https://en.wikipedia.org/wiki/Multinomial_theorem

Polynomial kernels

- More generally, $\mathrm{K}(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^{\mathrm{d}}$ is a kernel, for any positive integer d:

$$\mathrm{K}(\mathbf{x}, \mathbf{z}) = \left(\sum_{\mathrm{i}=1:\mathrm{m}} \mathrm{x}_{\mathrm{i}} \, \mathrm{z}_{\mathrm{i}} \right)^{\mathrm{d}}$$

- If we expanded the sum above in the naïve way, we get $\mathrm{O}(\mathrm{m}^{\mathrm{d}})$ terms.
  - See Multinomial Theorem: https://en.wikipedia.org/wiki/Multinomial_theorem

- Terms are monomials (products of x_{i} and/or z_{i}) with total power equal to d.
- If we use the primal form of the SVM, each term gets a weight.

Polynomial kernels

- More generally, $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^d$ is a kernel, for any positive integer d:

$$K(\mathbf{x}, \mathbf{z}) = \left(\sum_{i=1:m} x_i z_i \right)^d$$

- If we expanded the sum above in the naïve way, we get $O(m^d)$ terms.
  - See Multinomial Theorem: https://en.wikipedia.org/wiki/Multinomial_theorem

- Terms are monomials (products of x_i and/or z_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 K can be done in $O(m)$.

The "kernel trick"

- If we work with the dual, we do not have to ever compute the feature mapping ϕ. We just compute the similarity kernel K.

The "kernel trick"

- If we work with the dual, we do not have to ever compute the feature mapping ϕ. We just compute the similarity kernel K.
- We can solve the dual for the α_i :

$$\text{Max} \quad \sum_{i=1:n} \alpha_i - \tfrac{1}{2} \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{w.r.t.} \quad \alpha_i$$

$$\text{s.t.} \quad \alpha_i \geq 0 \ \textit{and} \ \sum_{i:1..n} \alpha_i y_i = 0$$

The "kernel trick"

- If we work with the dual, we do not have to ever compute the feature mapping ϕ. We just compute the similarity kernel K.
- We can solve the dual for the α_i :

$$\text{Max} \quad \sum_{i=1:n} \alpha_i - \tfrac{1}{2} \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{w.r.t.} \quad \alpha_i$$

$$\text{s.t.} \quad \alpha_i \geq 0 \ \textit{and} \ \sum_{i:1..n} \alpha_i y_i = 0$$

- The class of a new input x is computed as:

$$h_{\mathbf{w}}(\mathbf{x}) = \text{sign}\left(\sum_{i=1:n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

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

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

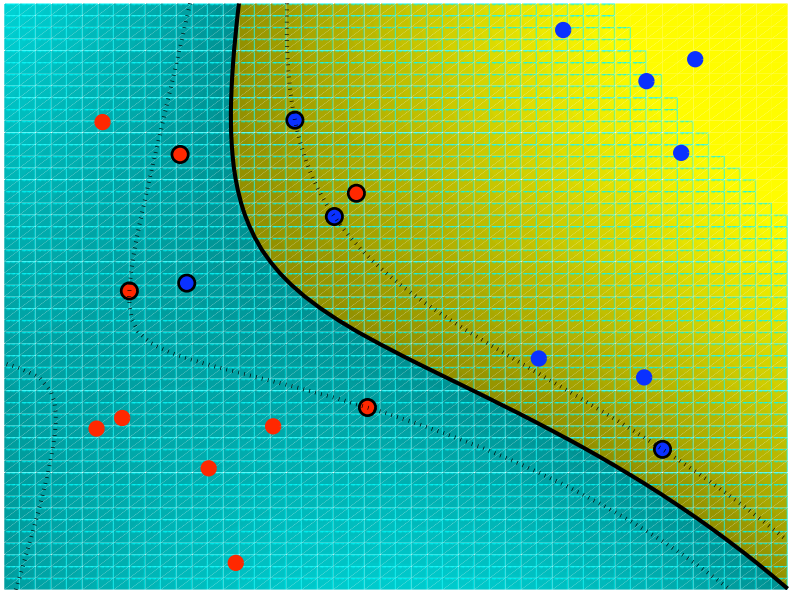
Some other kernel functions

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

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

- Sigmoidal kernel: $K(\mathbf{x}, \mathbf{z}) = \tanh(c_1\mathbf{x}\cdot\mathbf{z} + c_2)$

Example: Gaussian Kernel



Note the non-linear decision boundary

Kernels beyond SVMs

- A lot of research related to defining kernel functions suitable to particular tasks, or types of inputs (e.g. words, graphs, images).

- 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, …
- Often use a sliding window of length *k* over the two strings that we want to compare.
- Within the fixed-size window we can do many 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 and from today:

- Perceptron algorithm.
- Margin definition for linear SVMs.
- Use of Lagrange multipliers to transform optimization problems.
- Primal and dual optimization problems for SVMs.
- Feature space version of SVMs.
- The kernel trick and examples of common kernels.