Support Vectors, Duals, and the Kernel Trick

Machine Learning
Spring 2020
Support vector machines

- Training by maximizing margin
- The SVM objective
- Solving the SVM optimization problem
- Support vectors, duals and kernels
This lecture

1. Dual forms, and support vectors
2. Kernels & kernel trick
3. Properties of kernels
4. Nonlinear SVM
This lecture

1. Dual forms, and support vectors

2. Kernels & kernel trick

3. Properties of kernels

4. Nonlinear SVM
So far we have seen

- Support vector machines
- Hinge loss and optimizing the regularized loss

More broadly, different algorithms for learning linear classifiers
So far we have seen

- Support vector machines
- Hinge loss and optimizing the regularized loss

More broadly, different algorithms for learning linear classifiers

What about non-linear models?
One way to learn non-linear models

Explicitly introduce non-linearity into the feature space

If the true separator is quadratic
One way to learn non-linear models

Explicitly introduce non-linearity into the feature space

If the true separator is quadratic

Transform all input points as

$$\phi(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_2^2 \end{bmatrix}$$
One way to learn non-linear models

Explicitly introduce non-linearity into the feature space

If the true separator is quadratic

Transform all input points as

\[ \phi(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix} \]

Now, we can try to find a weight vector in this higher dimensional space

That is, predict using \( w^T \phi(x_1, x_2) + b \geq 0 \)
Primal and dual forms: constraint optimization

Primal: general constraint optimization problem

\[
\begin{align*}
\min_{\mathbf{x}} & \quad f(\mathbf{x}) \\
\text{s.t.} & \quad g_1(\mathbf{x}) \leq 0, \ldots, g_m(\mathbf{x}) \leq 0
\end{align*}
\]
Primal and dual forms: constraint optimization

Primal: general constraint optimization problem

\[
\begin{align*}
\min_{\mathbf{x}} & \quad f(\mathbf{x}) \\
\text{s.t.} & \quad g_1(\mathbf{x}) \leq 0, \ldots, g_m(\mathbf{x}) \leq 0
\end{align*}
\]

Lagrange form

\[
L(\mathbf{x}, \mathbf{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x})
\]
Primal and dual forms: constraint optimization

Primal: general constraint optimization problem

$$\min_x f(x)$$

s.t. $$g_1(x) \leq 0, \ldots, g_m(x) \leq 0$$

Lagrange form

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

Equivalent formulation, removing constraints on $$x$$

$$\min \max_x \min_{\lambda} L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

s.t. $$\lambda_1 \geq 0, \ldots, \lambda_m \geq 0$$
Primal and dual forms: constraint optimization

Primal: general constraint optimization problem

$$\min_{x} \ f(x)$$

s.t. \ $g_1(x) \leq 0, \ldots, g_m(x) \leq 0$

Lagrange form \ $L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$

Equivalent formulation, removing constraints on $x$

$$\min_{x} \ \max_{\lambda \geq 0} \ L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$
Primal and dual forms: constraint optimization

Primal: general constraint optimization problem

$$\min_x f(x)$$
$$\text{s.t. } g_1(x) \leq 0, \ldots, g_m(x) \leq 0$$

Why equivalent?

$$\max_{\lambda \geq 0} f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) = \begin{cases} f(x) & \text{if } x \text{ satisfy all the constraints} \\ \infty & \text{otherwise} \end{cases}$$
Primal and dual forms: constraint optimization

Primal: general constraint optimization problem

\[
\begin{align*}
\min_{\mathbf{x}} & \quad f(\mathbf{x}) \\
\text{s.t.} & \quad g_1(\mathbf{x}) \leq 0, \ldots, g_m(\mathbf{x}) \leq 0
\end{align*}
\]

Why equivalent?

\[
\begin{align*}
\max_{\lambda \geq 0} & \quad f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \text{ satisfy all the constraints} \\ \infty & \text{otherwise} \end{cases} \\
\lambda_1 = \ldots = \lambda_m = 0
\end{align*}
\]
Primal and dual forms: constraint optimization

**Primal:** general constraint optimization problem

\[
\max_{\lambda \geq 0} f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) = \begin{cases} f(x) & \text{if } x \text{ satisfy all the constraints} \\ \infty & \text{otherwise} \end{cases}
\]

Outer minimization:

Search \( x \) over constraint space

To minimize \( f(x) \)

\[
\min \max_{x, \lambda \geq 0} L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)
\]
Primal and dual forms: constraint optimization

Primal: general constraint optimization problem

\[
\begin{align*}
\min_{x} \max_{\lambda \geq 0} \quad & L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \\
\text{s.t.} \quad & g_1(x) \leq 0, \ldots, g_m(x) \leq 0
\end{align*}
\]

Totally equivalent!

It is very trial to incorporate equality constraints! How?
Primal and dual forms: constraint optimization

We focus on this primal form

$$\min_x \max_{\lambda \geq 0} \quad L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$
Primal and dual forms: constraint optimization

**Primal form**

\[
\min_x \max_{\lambda \geq 0} \quad L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)
\]

**Dual Form**

\[
\max_{\lambda \geq 0} \min_x \quad L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)
\]

Just switch \(\min\) and \(\max\)!
Primal and dual forms: constraint optimization

Generally, Dual \leq Primal (Why?)

\[
\max_{\lambda \geq 0} \min_x L(x, \lambda) \leq \min_x \max_{\lambda \geq 0} L(x, \lambda)
\]
Primal and dual forms: constraint optimization

Generally, Dual $\leq$ Primal \textbf{(Why?)}

$$\max_{\lambda \geq 0} \min_x L(x, \lambda) \leq \min_x \max_{\lambda \geq 0} L(x, \lambda)$$

Denote $L(x^*, \lambda^*) = \min_x \max_{\lambda \geq 0} L(x, \lambda)$
Primal and dual forms: constraint optimization

Generally, Dual $\leq$ Primal (Why?)

\[
\max_{\lambda \geq 0} \min_x L(x, \lambda) \leq \min_x \max_{\lambda \geq 0} L(x, \lambda)
\]

Denote $L(x^*, \lambda^*) = \min_x \max_{\lambda \geq 0} L(x, \lambda)$

\[
\forall \lambda \geq 0 \quad L(x^*, \lambda) \leq L(x^*, \lambda^*)
\]
Primal and dual forms: constraint optimization

Generally, Dual $\leq$ Primal (Why?)

$$
\max_{\lambda \geq 0} \min_x L(x, \lambda) \leq \min_x \max_{\lambda \geq 0} L(x, \lambda)
$$

Denote $L(x^*, \lambda^*) = \min_x \max_{\lambda \geq 0} L(x, \lambda)$

$$
\forall \lambda \geq 0 \quad L(x^*, \lambda) \leq L(x^*, \lambda^*)
$$

$$
\forall \lambda \geq 0 \quad \min_x L(x, \lambda) \leq L(x^*, \lambda)
$$
Primal and dual forms: constraint optimization

Generally, Dual $\leq$ Primal (Why?)

$$\max_{\lambda \geq 0} \min_x L(x, \lambda) \leq \min_x \max_{\lambda \geq 0} L(x, \lambda)$$

Denote $L(x^*, \lambda^*) = \min_x \max_{\lambda \geq 0} L(x, \lambda)$

$$\forall \lambda \geq 0 \quad L(x^*, \lambda) \leq L(x^*, \lambda^*)$$

$$\forall \lambda \geq 0 \quad \min_x L(x, \lambda) \leq L(x^*, \lambda) \leq L(x^*, \lambda^*)$$
Primal and dual forms: constraint optimization

Generally, Dual ≤ Primal (Why?)

\[
\max_{\lambda \geq 0} \min_x L(x, \lambda) \leq \min_x \max_{\lambda \geq 0} L(x, \lambda)
\]

Denote \[L(x^*, \lambda^*) = \min_x \max_{\lambda \geq 0} L(x, \lambda)\]

\[
\forall \lambda \geq 0 \quad L(x^*, \lambda) \leq L(x^*, \lambda^*)
\]

\[
\forall \lambda \geq 0 \quad \min_x L(x, \lambda) \leq L(x^*, \lambda) \leq L(x^*, \lambda^*)
\]
Primal and dual forms: constraint optimization

Generally, Dual ≤ Primal (Why?)

\[
\max_{\lambda \geq 0} \min_{x} L(x, \lambda) \leq \min_{x} \max_{\lambda \geq 0} L(x, \lambda)
\]

Denote \( L(x^*, \lambda^*) = \min_{x} \max_{\lambda \geq 0} L(x, \lambda) \)

\[
\forall \lambda \geq 0 \quad L(x^*, \lambda) \leq L(x^*, \lambda^*)
\]

\[
\forall \lambda \geq 0 \quad \min_{x} L(x, \lambda) \leq L(x^*, \lambda) \leq L(x^*, \lambda^*)
\]

\[
\max_{\lambda} \min_{x} L(x, \lambda) \leq L(x^*, \lambda^*)
\]
Primal and dual forms: constraint optimization

Generally, Dual ≤ Primal (Why?)

\[
\max_{\lambda \geq 0} \min_x L(x, \lambda) \leq \min_x \max_{\lambda \geq 0} L(x, \lambda)
\]

Fortunately, for SVM, Dual = Primal!
SVM: Primal and dual

The soft SVM objective

\[
\min_{\mathbf{w}, \{\xi_i\}} \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_i \xi_i \\
\text{s.t.} \forall i, \quad y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\
\xi_i \geq 0
\]
SVM: Primal and dual

Primal:
\[
\begin{align*}
\min_{\mathbf{w}, b, \{\xi_i\}} & \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_i \xi_i \\
\text{s.t.} & \quad \forall i, \quad y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0
\end{align*}
\]

\[
\begin{align*}
\min_{\mathbf{w}, b, \{\xi_i\}} \max_{\alpha_i \geq 0, \beta_i \geq 0} & \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i)
\end{align*}
\]
Let us look at the dual form of SVM

**Primal:**

\[
\min_{\mathbf{w}, b, \{\xi_i\}} \{\alpha_i \geq 0, \beta_i \geq 0\} \quad \max \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i)
\]

**Dual:**

\[
\max \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i)
\]

According to Slater’s condition for convex optimization, they have the same solution! Solving Dual = Solving Primal!
Let us look at the dual form of SVM

\[
\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} \min_{w, b, \{\xi_i\}} \frac{1}{2} w^T w + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i(w^T x_i + b) - 1 + \xi_i)
\]

Let us fix Lagrangian multipliers, solve the **inner** optimization

\[
\frac{\partial L}{\partial w} = 0, \quad \frac{\partial L}{\partial b} = 0, \quad \frac{\partial L}{\partial \xi_i} = 0
\]

Solve

What can you get?
Let us look at the dual form of SVM

\[
\begin{align*}
\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} \min_{w, b, \{\xi_i\}} & \quad \frac{1}{2} w^T w + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) \\
& \quad \text{Let us fix Lagrangian multipliers, solve the } \textbf{inner} \text{ optimization} \\
\frac{\partial L}{\partial w} = 0 & \implies w = \sum_i \alpha_i y_i x_i \\
\frac{\partial L}{\partial b} = 0 & \implies \sum_i \alpha_i y_i = 0 \\
\frac{\partial L}{\partial \xi_i} = 0 & \implies \alpha_i + \beta_i = C
\end{align*}
\]
Let us look at the dual form of SVM

\[
\begin{align*}
\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} \quad & \min \ w, b, \{\xi_i\} \\
& \frac{1}{2} w^T w + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i)
\end{align*}
\]

Let us fix Lagrangian multipliers, solve the *inner* optimization

\[
\frac{\partial L}{\partial w} = 0 \implies w = \sum_i \alpha_i y_i x_i
\]

\[
\frac{\partial L}{\partial b} = 0 \implies \sum_i \alpha_i y_i = 0 \quad \text{Substitute them into } L
\]

\[
\frac{\partial L}{\partial \xi_i} = 0 \implies \alpha_i + \beta_i = C
\]
Let us look at the dual form of SVM

\[
\begin{aligned}
\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} \min w, b, \{\xi_i\} & \quad \frac{1}{2} w^T w + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) \\
\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} & \quad -\frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j + \sum \alpha_i \\
\text{s.t.} & \quad \sum \alpha_i y_i = 0,
\forall i, \alpha_i + \beta_i = C
\end{aligned}
\]
Let us look at the dual form of SVM

\[
\begin{align*}
\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} & \quad \min_{w, b, \xi_i} \quad \frac{1}{2} w^T w + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) \\
\end{align*}
\]

\[
\begin{align*}
\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} & \quad -\frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j + \sum_i \alpha_i \\
\text{s.t.} & \quad \sum_i \alpha_i y_i = 0, \\
& \quad \forall i, \alpha_i + \beta_i = C
\end{align*}
\]

Note we can remove $\beta_i$ by constraining that $\alpha_i \leq C$
Let us look at the dual form of SVM

Finally, we can solve the dual form by

\[
\min_{\{0 \leq \alpha_i \leq C\}, \sum \alpha_i y_i = 0} \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_i \alpha_i
\]

Quadratic convex optimization problem!
Let us look at the dual form of SVM

Finally, we can solve the dual form by

\[
\begin{align*}
\min_{\{0 \leq \alpha_i \leq C\}, \sum_i \alpha_i y_i = 0} & \quad \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_i \alpha_i \\
\end{align*}
\]

Quadratic convex optimization problem!

Why would like to solve the dual form?

As we will see, it will enable kernel trick and nonlinear classification in \textit{infinite} dimensional space!

It will also save the model storage through support vectors!
Let us look at the dual form of SVM

Let us first solve the dual

\[
\min_{\{0 \leq \alpha_i \leq C\}, \sum_i \alpha_i y_i = 0} \quad \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_i \alpha_i
\]

After we obtain the optimal Lagrangian multipliers \( \alpha^*_1, \ldots, \alpha^*_N \)

How to obtain optimal \( w, b \)?
Let us look at the dual form of SVM

Let us first solve the dual

\[
\min_{\{0 \leq \alpha_i \leq C\}, \sum_i \alpha_i y_i = 0} \quad \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_i \alpha_i
\]

After we obtain the optimal Lagrangian multipliers \( \alpha_1^*, \ldots, \alpha_N^* \)

\[
\frac{\partial L}{\partial w} = 0 \implies w = \sum_i \alpha_i y_i x_i
\]

\[
w^* = \sum_i \alpha_i^* y_i x_i
\]
Let us look at the dual form of SVM

Let us first solve the dual

$$\min_{\{0 \leq \alpha_i \leq C\}, \sum_i \alpha_i y_i = 0} \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_i \alpha_i$$

After we obtain the optimal Lagrangian multipliers $\alpha_1^*, \ldots, \alpha_N^*$

$$\frac{\partial L}{\partial w} = 0 \implies w = \sum_i \alpha_i y_i x_i$$

$$w^* = \sum_i \alpha_i^* y_i x_i$$

How to get $b$?
**The Karush-Kuhn-Tucker (KKT) conditions**

**KKT (inequality only)**
The KKT conditions for the convex program

\[
\begin{align*}
\text{minimize } & f_0(x) \quad \text{subject to } f_1(x) \leq 0 \\
& f_2(x) \leq 0 \\
& \vdots \\
& f_M(x) \leq 0
\end{align*}
\]

in \( x \in \mathbb{R}^N \) and \( \lambda \in \mathbb{R}^M \) are

\[
\begin{align*}
f_m(x) & \leq 0, \quad m = 1, \ldots, M, \quad (K1) \\
\lambda & \geq 0, \quad (K2) \\
\lambda_m f_m(x) & = 0, \quad m = 1, \ldots, M, \quad (K3) \\
\nabla f_0(x) + \sum_{m=1}^{M} \lambda_m \nabla f_m(x) & = 0, \quad (K4)
\end{align*}
\]
The Karush-Kuhn-Tucker (KKT) conditions

**KKT (inequality only)**
The KKT conditions for the convex program

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \quad \text{subject to} \quad f_1(x) \leq 0, \\
& \quad f_2(x) \leq 0, \\
& \quad \vdots \\
& \quad f_M(x) \leq 0
\end{align*}
\]

in \( x \in \mathbb{R}^N \) and \( \lambda \in \mathbb{R}^M \) are

\[
\begin{align*}
f_m(x) & \leq 0, \quad m = 1, \ldots, M, \quad (K1) \\
\lambda & > 0, \quad (K2) \\
\lambda_m f_m(x) & = 0, \quad m = 1, \ldots, M, \quad (K3) \\
\nabla f_0(x) + \sum_{m=1}^M \lambda_m \nabla f_m(x) & = 0, \quad (K4)
\end{align*}
\]
The Karush-Kuhn-Tucker (KKT) conditions

Sufficient condition

If the KKT conditions hold for $\mathbf{x}^*$ and some $\lambda^* \in \mathbb{R}^M$, then $\mathbf{x}^*$ is a solution to the program (1).

Necessary condition

Suppose $\mathbf{x}^*$ is a solution to a convex program with affine inequality constraints:

$$\min_{\mathbf{x} \in \mathbb{R}^N} f_0(\mathbf{x}) \quad \text{subject to} \quad A\mathbf{x} \leq \mathbf{b}.$$ 

Then there exists a $\mathbf{\lambda}^*$ such that $\mathbf{x}^*$, $\mathbf{\lambda}^*$ obey the KKT conditions.
The Karush-Kuhn-Tucker (KKT) conditions

Sufficient condition

If the KKT conditions hold for \( \mathbf{x}^* \) and some \( \lambda^* \in \mathbb{R}^M \), then \( \mathbf{x}^* \) is a solution to the program (1).

Necessary condition

Suppose \( \mathbf{x}^* \) is a solution to a convex program with affine inequality constraints:

\[
\min_{\mathbf{x} \in \mathbb{R}^N} f_0(\mathbf{x}) \quad \text{subject to} \quad A\mathbf{x} \leq \mathbf{b}.
\]

Then there exists a \( \lambda^* \) such that \( \mathbf{x}^*, \lambda^* \) obey the KKT conditions.

We can use KKT condition to analyze our solution! Both sufficient and necessary! Why?

Georgia Tech ECE 8823a Notes by J. Romberg. Last updated 9:52, February 22, 2017
SVM: Primal and dual

Primal:

\[
\min_{w, b, \{\xi_i\}} \frac{1}{2} w^\top w + C \sum_i \xi_i \\
\text{s.t. } \forall i, \quad y_i (w^\top x_i + b) \geq 1 - \xi_i \\
\quad \xi_i \geq 0
\]

\[
\min_{w, b, \{\xi_i\}} \max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} \frac{1}{2} w^\top w + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (w^\top x_i + b) - 1 + \xi_i)
\]
Optimal solution: KKT condition

\[
\min_{w, b, \{\xi_i\}} \max_{\alpha_i \geq 0, \beta_i \geq 0} \quad \frac{1}{2} w^\top w + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (w^\top x_i + b) - 1 + \xi_i)
\]

Complementary Slackness

\[
\forall i, \beta_i^* \xi_i^* = 0
\]

\[
\forall i, \alpha_i^* (y_i (w^*^\top x_i + b^*) - 1 + \xi_i^*) = 0
\]
Let us look at the dual form of SVM

\[
\begin{align*}
\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} \quad & \min_{w, b, \{\xi_i\}} \quad \frac{1}{2} w^\top w + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (w^\top x_i + b) - 1 + \xi_i) \\
L & = 0 \implies w = \sum_i \alpha_i y_i x_i \\
L & = 0 \implies \sum_i \alpha_i y_i = 0 \\
L & = 0 \implies \alpha_i + \beta_i = C
\end{align*}
\]

Let us fix Lagrangian multipliers, solve the \textit{inner} optimization
Optimal solution: KKT condition

\[
\min_{\mathbf{w}, b, \{\xi_i\}} \max_{\alpha_i \geq 0, \beta_i \geq 0} \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_i \xi_i - \sum_i \beta_i \xi_i - \sum_i \alpha_i (y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i)
\]

**Complementary Slackness**

\[
\forall i, \beta_i^* \xi_i^* = 0
\]

\[
\forall i, \alpha_i^* (y_i (\mathbf{w}^* \top \mathbf{x}_i + b^*) - 1 + \xi_i^*) = 0
\]

When solving the dual, we have

\[
\forall i, \alpha_i^* + \beta_i^* = C
\]
Optimal solution: KKT condition

\[ \forall i, \beta^*_i \xi^*_i = 0 \quad \forall i, \alpha^*_i (y_i (w^* \mathbf{x}_i + b^*) - 1 + \xi^*_i) = 0 \]

\[ \forall i, \alpha^*_i + \beta^*_i = C \]

Let us find some \( j \), \( 0 < \alpha^*_j < C \), what can we get?
Optimal solution: KKT condition

\[ \forall i, \beta_i^* \xi_i^* = 0 \quad \forall i, \alpha_i^* (y_i (w^* \mathbf{x}_i + b^*) - 1 + \xi_i^*) = 0 \]

\[ \forall i, \alpha_i^* + \beta_i^* = C \]

Let us find some \( j \), \( 0 < \alpha_j^* < C \), what can we get?

\[ \beta_j^* > 0 \implies \xi_j^* = 0 \]

\[ \alpha_j^* > 0 \implies y_j (w^* \mathbf{x}_j + b^*) - 1 + \xi_j^* = 0 \]

\[ \implies y_j (w^* \mathbf{x}_j + b^*) - 1 = 0 \]

\[ \implies w^* \mathbf{x}_j + b^* = y_j \]

\[ \implies b^* = y_j - w^* \mathbf{x}_j \]
Optimal solution: KKT condition

\[ \forall i, \beta_i^* \xi_i^* = 0 \quad \forall i, \alpha_i^*(y_i(w^*\mathbf{x}_i + b^*) - 1 + \xi_i^*) = 0 \]

\[ \forall i, \alpha_i^* + \beta_i^* = C \]

Let us find some \( j \), \( 0 < \alpha_j^* < C \), what can we get?

\[ \beta_j^* > 0 \implies \xi_j^* = 0 \]

\[ \alpha_j^* > 0 \implies y_j(w^*\mathbf{x}_j + b^*) - 1 + \xi_j^* = 0 \]

\[ \implies y_j(w^*\mathbf{x}_j + b^*) - 1 = 0 \]

\[ \implies w^*\mathbf{x}_j + b^* = y_j \]

\[ \implies b^* = y_j - w^*\mathbf{x}_j \]

For robustness, we often average the results over all such \( j \)
Optimal solution: KKT condition

\[ \forall i, \beta_i^* \xi_i^* = 0 \quad \forall i, \alpha_i^* \left( y_i (w^*^\top x_i + b^*) - 1 + \xi_i^* \right) = 0 \]

\[ \forall i, \alpha_i^* + \beta_i^* = C \]

What if some \( \alpha_j^* = 0 \)?

\[ w^* = \sum_i \alpha_i^* y_i x_i \]

\( (x_j, y_j) \) will not affect the weight vector!
Optimal solution: KKT condition

\[ \forall i, \beta_i^* \xi_i^* = 0 \quad \forall i, \alpha_i^* (y_i (w^* \mathbf{x}_i + b^*) - 1 + \xi_i^*) = 0 \]

\[ \forall i, \alpha_i^* + \beta_i^* = C \]

What if some \( \alpha_j^* = 0 \)?

\[ w^* = \sum_i \alpha_i^* y_i x_i \]

(\( \mathbf{x}_j, y_j \)) will not affect the weight vector!

The weight vector is only determined by samples with \textbf{nonzero} \( \alpha_j^* \)!
Optimal solution: KKT condition

\[ \forall i, \beta_i^* \xi_i^* = 0 \quad \forall i, \alpha_i^* \left( y_i (\mathbf{w}^* \mathbf{x}_i + b^*) - 1 + \xi_i^* \right) = 0 \]

\[ \forall i, \alpha_i^* + \beta_i^* = C \]

What if some \( \alpha_j^* = 0 \)?

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

\((x_j, y_j)\) will not affect the weight vector!

The weight vector is only determined by samples with \textbf{nonzero} \( \alpha_j^* \)!

We call these subset of samples as **Support Vectors**!
Where are the support vectors?

\[ S = \{ x_j \mid \alpha_j^* > 0 \} \quad \Rightarrow \quad w^* = \sum_{j \in S} \alpha_j^* y_j x_j \]

For support vector \( x_j \), according to Complementary Slackness

\[ \alpha_j^* (y_j (w^* \top x_j + b^*) - 1 + \xi_j^*) = 0 \]

We must have

\[ y_j (w^* \top x_j + b^*) = 1 - \xi_j^* \leq 1 \]
Where are the support vectors?

\[ S = \{ x_j | \alpha_j^* > 0 \} \]

\[ w^* = \sum_{j \in S} \alpha_j^* y_j x_j \]

For support vector \( x_j \), according to Complementary Slackness

\[ \alpha_j^* (y_j (w^*^\top x_j + b^*) - 1 + \xi_j^*) = 0 \]

We must have

\[ y_j (w^*^\top x_j + b^*) = 1 - \xi_j^* \leq 1 \]

Support vectors must be on the margin/inside the margin!
Where are the support vectors?

\[
S = \{ \mathbf{x}_j | \alpha_j^* > 0 \} \quad \rightarrow \quad \mathbf{w}^* = \sum_{j \in S} \alpha_j^* y_j \mathbf{x}_j
\]

For support vector \( \mathbf{x}_j \), according to Complementary Slackness

\[
\alpha_j^* (y_j (\mathbf{w}^* \mathbf{x}_j + b^*) - 1 + \xi_j^*) = 0
\]

We must have

\[
y_j (\mathbf{w}^* \mathbf{x}_j + b^*) = 1 - \xi_j^* \leq 1
\]

Support vectors must be on the margin/inside the margin!

Question: what if a support vector stays inside the margin?
Support Vectors

\[ w^* = \sum_{i} \alpha_i^* y_i x_i \]

The solution tends to be sparse

Most \( \alpha_i^* = 0 \)

No need to store those points to Computer weights/make predictions

Non-support vectors \( \alpha_i^* = 0 \)

Support vectors \( \alpha_i^* > 0 \)
This lecture

- Dual forms, and support vectors

2. Kernels & kernel trick

3. Properties of kernels

4. Nonlinear SVM
Predicting with SVM dual solution

• Prediction = \( sgn(\mathbf{w}^* \mathbf{x} + b^*) \), and \( \mathbf{w}^* = \sum_i \alpha_i^* y_i x_i \)
Predicting with SVM dual solution

- Prediction $= sgn(\mathbf{w}^* \mathbf{x} + b^*)$, and $\mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$

- That is, we just showed that $\mathbf{w}^* \mathbf{x} = \sum \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}$
Predicting with SVM dual solution

• Prediction = \( sgn(\mathbf{w}^* \mathbf{x} + b^*) \), and \( \mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x}_i \)

• That is, we just showed that \( \mathbf{w}^* \mathbf{x} = \sum \alpha_i^* y_i \mathbf{x}_i \mathbf{x} \)

  – We only need to compute dot products between training examples (that are support vectors) and the new example \( \mathbf{x} \)
Predicting with SVM dual solution

• Prediction = \( \text{sgn}(\mathbf{w}^* \mathbf{x} + b^*) \), and \( \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i \)

• That is, we just showed that \( \mathbf{w}^* \mathbf{x} = \sum \alpha_i y_i \mathbf{x}_i \mathbf{x} \)

  – We only need to compute dot products between training examples (that are support vectors) and the new example \( \mathbf{x} \)
  – This is true even if we map examples to a high dimensional space

\[
\mathbf{w}^* \mathbf{x} = \sum \alpha_i y_i \mathbf{x}_i \mathbf{x} = \sum \alpha_i y_i \mathbf{x}_i \mathbf{x} \]

\[
\mathbf{w}^* \mathbf{x} = \sum \alpha_i y_i \mathbf{x}_i \mathbf{x} = \sum \alpha_i y_i \phi(\mathbf{x}_i) \mathbf{x} \]

\[
\mathbf{w}^* \mathbf{x} = \sum \alpha_i y_i \phi(\mathbf{x}_i) \mathbf{x} = \sum \alpha_i y_i \phi(\mathbf{x}_i) \mathbf{x} \]

\[
\mathbf{w}^* \mathbf{x} = \sum \alpha_i y_i \phi(\mathbf{x}_i) \mathbf{x} = \sum \alpha_i y_i \phi(\mathbf{x}_i) \mathbf{x} \]

\[
\mathbf{w}^* \mathbf{x} = \sum \alpha_i y_i \phi(\mathbf{x}_i) \mathbf{x} = \sum \alpha_i y_i \phi(\mathbf{x}_i) \mathbf{x} \]
Predicting with SVM dual solution

- Prediction =
- That is, we just showed that
- This is true even if we map examples to a high dimensional space
- That is we only need to compute dot products between training examples and the new example

One way to learn non-linear models

Explicitly introduce non-linearity into the feature space

If the true separator is quadratic

Transform all input points as

\[
\phi(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}
\]

Now, we can try to find a weight vector in this higher dimensional space

That is, predict using \( \mathbf{w}^T \phi(x_1, x_2) + b \geq 0 \)
Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

\[ K(x, z) = \phi(x)^T \phi(z) \]
Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

\[ K(x, z) = \phi(x)^T \phi(z) \]

So prediction with this *high dimensional lifting map* is

\[ \text{sgn}(w^T \phi(x) + b) = \text{sgn}(\sum_i \alpha_i y_i K(x_i, x) + b) \]

because \( w^T \phi(x) = \sum \alpha_i y_i \phi(x_i)^T \phi(x) \)
Kernel based methods

\[ K(x, z) = \phi(x)^T \phi(z) \]

Predict using

\[ \text{sgn}(w^T \phi(x) + b) = \text{sgn} \left( \sum_{i} \alpha_i y_i K(x_i, x) + b \right) \]

What does this new formulation give us?

If we have to compute \( \phi \) every time anyway, we gain nothing
Kernel based methods

\[ K(x, z) = \phi(x)^T \phi(z) \]

Predict using

\[ \text{sgn}(w^T \phi(x) + b) = \text{sgn}\left( \sum_i \alpha_i y_i K(x_i, x) + b \right) \]

*What does this new formulation give us?*

If we have to compute \( \phi \) every time anyway, we gain nothing

*If we can compute the value of \( K \) without explicitly writing the blown up representation, then we will have a computational advantage*
Example: Polynomial Kernel

• Given two examples $\mathbf{x}$ and $\mathbf{z}$ we want to map them to a high dimensional space [for example, quadratic]

$$
\phi(x_1, x_2, \cdots, x_n) = [1, x_1, x_2, \cdots, x_n, x_1^2, x_2^2, \cdots x_n^2, x_1x_2, \cdots, x_{n-1}x_n]^T
$$
Example: Polynomial Kernel

• Given two examples \( x \) and \( z \) we want to map them to a high dimensional space [for example, quadratic]

\[
\phi(x_1, x_2, \cdots, x_n) = [1, x_1, x_2, \cdots, x_n, x_1^2, x_2^2, \cdots x_n^2, x_1x_2, \cdots, x_{n-1}x_n]^T
\]

All degree zero terms
Example: Polynomial Kernel

- Given two examples $\mathbf{x}$ and $\mathbf{z}$ we want to map them to a high dimensional space [for example, quadratic]

$$
\phi(x_1, x_2, \cdots, x_n) = \begin{bmatrix} 1, x_1, x_2, \cdots, x_n, x_1^2, x_2^2, \cdots x_n^2, x_1x_2, \cdots, x_{n-1}x_n \end{bmatrix}^T
$$

All degree zero terms

All degree one terms
Example: Polynomial Kernel

- Given two examples $x$ and $z$ we want to map them to a high dimensional space [for example, quadratic]

$$
\phi(x_1, x_2, \ldots, x_n) = [1, x_1, x_2, \ldots, x_n, x_1^2, x_2^2, \ldots, x_n^2, x_1x_2, \ldots, x_{n-1}x_n]^T
$$

- All degree zero terms
- All degree one terms
- All degree two terms
Example: Polynomial Kernel

- Given two examples $\mathbf{x}$ and $\mathbf{z}$ we want to map them to a high dimensional space [for example, quadratic]

$$\phi(x_1, x_2, \cdots, x_n) = [1, x_1, x_2, \cdots, x_n, x_1^2, x_2^2, \cdots, x_n^2, x_1x_2, \cdots, x_{n-1}x_n]^T$$

  All degree zero terms  |  All degree one terms  |  All degree two terms

and compute the dot product $A = \phi(\mathbf{x})^T \phi(\mathbf{z})$ [takes time]
Example: Polynomial Kernel

• Given two examples $x$ and $z$ we want to map them to a high dimensional space [for example, quadratic]

$$
\phi(x_1, x_2, \cdots, x_n) = [1, x_1, x_2, \cdots, x_n, x_1^2, x_2^2, \cdots x_n^2, x_1x_2, \cdots, x_{n-1}x_n]^T
$$

and compute the dot product $A = \phi(x)^T \phi(z)$ [takes time]
Example: Polynomial Kernel

- Given two examples $x$ and $z$ we want to map them to a high dimensional space [for example, quadratic]

$$\phi(x_1, x_2, \cdots, x_n) = [1, x_1, x_2, \cdots, x_n, x_1^2, x_2^2, \cdots x_n^2, x_1x_2, \cdots, x_{n-1}x_n]^T$$

and compute the dot product $A = \phi(x)^T \phi(z)$ [takes time]

- Instead, in the original space, compute
Example: Polynomial Kernel

• Given two examples \( x \) and \( z \) we want to map them to a high dimensional space [for example, quadratic]

\[
\phi(x_1, x_2, \cdots, x_n) = [1, x_1, x_2, \cdots, x_n, x_1^2, x_2^2, \cdots x_n^2, x_1 x_2, \cdots, x_{n-1} x_n]^T
\]

and compute the dot product \( A = \phi(x)^T \phi(z) \) [takes time ]

• Instead, in the original space, compute a simple function

\[
B = K(x, z) = (1 + x^T z)^2
\]
Example: Polynomial Kernel

- Given two examples $x$ and $z$ we want to map them to a high dimensional space [for example, quadratic]

\[ \phi(x_1, x_2, \cdots, x_n) = [1, x_1, x_2, \cdots, x_n, x_1^2, x_2^2, \cdots, x_n^2, x_1 x_2, \cdots, x_{n-1} x_n]^T \]

and compute the dot product $A = \phi(x)^T \phi(z)$ [takes time]

- Instead, in the original space, compute a simple function

\[ B = K(x, z) = (1 + x^T z)^2 \]

Claim: $A = B$ (Coefficients do not really matter)
Example: Two dimensions, quadratic kernel

\[
A = \phi(x)^T \phi(z) \quad \quad B = K(x, z) = (1 + x^T z)^2
\]

\[
\phi(x_1, x_2) = \begin{bmatrix}
1 \\
x_1 \\
x_2 \\
x_1^2 \\
x_2^2 \\
x_1 x_2
\end{bmatrix}
\]
The Kernel Trick

Suppose we wish to compute $K(x, z) = \phi(x)^T \phi(z)$

Here $\phi$ maps $x$ and $z$ to a high dimensional space

**The Kernel Trick:** Save time/space by computing the value of $K(x, z)$ by performing operations in the original space (without a feature transformation!)
Computing dot products efficiently

**Kernel Trick:** You want to work with degree 2 polynomial features, \( \phi(x) \). Then, your dot product will be operate using vectors in a space of dimensionality \( 1 + n + \frac{n(n+1)}{2} \).

The kernel trick allows you to save time/space and compute dot products in an \( n \) dimensional space.  

*(Not just for degree 2 polynomials)*
This lecture

✓ Dual forms, and support vectors
✓ Kernels and kernel trick

3. Properties of kernels

4. Nonlinear SVM
Which functions are kernels?

**Kernel Trick:** You want to work with degree 2 polynomial features, $\phi(x)$. Then, your dot product will be operate using vectors in a space of dimensionality $1 + n + n(n+1)/2$.

The kernel trick allows you to save time/space and compute dot products in an $n$ dimensional space.  

*Not just for degree 2 polynomials*

- Can we use any function $K(.,.)$?
Which functions are kernels?

**Kernel Trick:** You want to work with degree 2 polynomial features, $\phi(x)$. Then, your dot product will be operate using vectors in a space of dimensionality $1 + n + \frac{n(n+1)}{2}$.

The kernel trick allows you to save time/space and compute dot products in an $n$ dimensional space. *(Not just for degree 2 polynomials)*

• **Can we use any function $K(.,.)$?**
  - No! A function $K(x,z)$ is a valid kernel if it corresponds to an inner product in some (perhaps infinite dimensional) feature space.
Which functions are kernels?

**Kernel Trick**: You want to work with degree 2 polynomial features, \( \phi(x) \). Then, your dot product will be operate using vectors in a space of dimensionality \( 1 + n + \frac{n(n+1)}{2} \).

The kernel trick allows you to save time/space and compute dot products in an \( n \) dimensional space.  

(Not just for degree 2 polynomials)

- **Can we use any function \( K(\cdot,\cdot) \)?**
  - **No!** A function \( K(x,z) \) is a valid kernel if it corresponds to an inner product in some (perhaps infinite dimensional) feature space.

- **General condition**: construct the Gram matrix \( \{K(x_i ,z_j)\} \); check that it’s positive semi definite
Reminder: Positive semi-definite matrices

A symmetric matrix $M$ is positive semi-definite if it is

- For any vector non-zero $z$, we have $z^T M z \geq 0$

(A useful property characterizing many interesting mathematical objects)
The Kernel Matrix

- The **Gram matrix** of a set of $n$ vectors $S = \{x_1...x_n\}$ is the $n \times n$ matrix $G$ with $G_{ij} = x_i^T x_j$
  - the Gram matrix of $\{\phi(x_1), ..., \phi(x_n)\}$
  - (size depends on the # of examples, not dimensionality)
  - Gram matrix is positive semidefinite
The Kernel Matrix

• The **Gram matrix** of a set of $n$ vectors $S = \{x_1,...,x_n\}$ is the $n \times n$ matrix $G$ with $G_{ij} = x_i^T x_j$
  
  – the Gram matrix of $\{\phi(x_1), ..., \phi(x_n)\}$
  
  – (size depends on the # of examples, not dimensionality)
  
  – Gram matrix is positive semidefinite

• **Showing that a function $K$ is a valid kernel**
  
  – **Direct approach:** If you have the $\phi(x_i)$
  
  – **Indirect:** Write down the Kernel matrix $K_{ij} = k(x_i, x_j)$ and show that it is a legitimate kernel, without an explicit construction of $\phi(x_i)$
Mercer’s condition

Let $K(\mathbf{x}, \mathbf{z})$ be a function that maps two $n$ dimensional vectors to a real number.

$K$ is a valid kernel if for every finite set $\{x_1, x_2, \cdots \}$, for any choice of real valued $c_1, c_2, \cdots$, we have

$$\sum_i \sum_j c_i c_j K(x_i, x_j) \geq 0$$
Polynomial kernels

- Linear kernel: \( k(x, z) = x^T z \)

- Polynomial kernel of degree \( d \): \( k(x, z) = (x^T z)^d \)
  - only \( d \)th-order interactions

- Polynomial kernel up to degree \( d \): \( k(x, z) = (x^T z + c)^d \) (\( c > 0 \))
  - all interactions of order \( d \) or lower
Gaussian Kernel
(or the radial basis function kernel)

\[ K_{rbf}(x, z) = \exp \left( -\frac{||x - z||^2}{c} \right) \]

- \( ||x - z||^2 \): squared Euclidean distance between \( x \) and \( z \)
- \( c = \sigma^2 \): a free parameter
- very small \( c \): \( K \approx \) identity matrix (every item is different)
- very large \( c \): \( K \approx \) unit matrix (all items are the same)

- \( k(x, z) \approx 1 \) when \( x, z \) close
- \( k(x, z) \approx 0 \) when \( x, z \) dissimilar
Gaussian Kernel
(or the radial basis function kernel)

\[ K_{rbf}(x, z) = \exp \left( -\frac{||x - z||^2}{c} \right) \]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td></td>
</tr>
<tr>
<td>( c = \sigma^2 )</td>
<td>a free parameter</td>
</tr>
<tr>
<td>( K \approx \text{identity matrix} )</td>
<td>very small ( c )</td>
</tr>
<tr>
<td>( K \approx \text{unit matrix} )</td>
<td>very large ( c )</td>
</tr>
</tbody>
</table>

- \( k(x, z) \approx 1 \) when \( x, z \) close
- \( k(x, z) \approx 0 \) when \( x, z \) dissimilar

Exercises:
1. Prove that this is a kernel.
2. What is the “blown up” feature space for this kernel?
Constructing New Kernels

You can construct new kernels $k'(x, x')$ from existing ones:

- Multiplying $k(x, x')$ by a positive constant $c$
  
  $ck(x, x')$

- Multiplying $k(x, x')$ by a function $f$ applied to $x$ and $x'$
  
  $f(x)k(x, x')f(x')$

- Applying a polynomial (with non-negative coefficients) to $k(x, x')$
  
  $P( k(x, x') )$ with $P(z) = \sum_i a_i z^i$ and $a_i \geq 0$

- Exponentiating $k(x, x')$
  
  $\exp(k(x, x'))$
Constructing New Kernels (2)

• You can construct $k'(\mathbf{x}, \mathbf{x'})$ from $k_1(\mathbf{x}, \mathbf{x'})$, $k_2(\mathbf{x}, \mathbf{x'})$ by:
  
  – Adding $k_1(\mathbf{x}, \mathbf{x'})$ and $k_2(\mathbf{x}, \mathbf{x'})$:
    \[ k_1(\mathbf{x}, \mathbf{x'}) + k_2(\mathbf{x}, \mathbf{x'}) \]

  – Multiplying $k_1(\mathbf{x}, \mathbf{x'})$ and $k_2(\mathbf{x}, \mathbf{x'})$:
    \[ k_1(\mathbf{x}, \mathbf{x'})k_2(\mathbf{x}, \mathbf{x'}) \]
Constructing New Kernels (2)

• You can construct \( k'(x, x') \) from \( k_1(x, x'), k_2(x, x') \) by:
  
  – Adding \( k_1(x, x') \) and \( k_2(x, x') \):
    \[
    k_1(x, x') + k_2(x, x')
    \]

  – Multiplying \( k_1(x, x') \) and \( k_2(x, x') \):
    \[
    k_1(x, x')k_2(x, x')
    \]

• Also:
  
  – If \( \phi(x) \in \mathbb{R}^m \) and \( k_m(z, z') \) a valid kernel in \( \mathbb{R}^m \),
    \( k(x, x') = k_m(\phi(x), \phi(x')) \) is also a valid kernel

  – If \( A \) is a symmetric positive semi-definite matrix,
    \( k(x, x') = xAx' \) is also a valid kernel
This lecture

✓ Support vectors

✓ Kernels & kernel trick

✓ Properties of kernels

4. Nonlinear SVM
How to implement nonlinear SVM with kernels?

- Learning: plug the kernel in the dual form

\[
\min_{\{0 \leq \alpha_i \leq C\}, \sum_i \alpha_i y_i = 0} \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_i \alpha_i
\]

Still a quadratic convex optimization problem!
How to implement nonlinear SVM with kernels?

• Prediction

$$\text{sgn}(\mathbf{w}^\top \phi(\mathbf{x}) + b) = \text{sgn}(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b)$$

Support vectors now are on/within nonlinear “margin” (in the original space)!
Nonlinear SVM example: Gaussian kernel

Level sets, i.e. $w \cdot x = r$ for some $r$

Support vectors

From David Sontag
Summary

• Dual form of SVM leads to the concept of support vectors

• To make the final prediction, we are computing dot products

• The kernel trick is a computational trick to compute dot products in higher dimensional spaces

• This is applicable not just to SVMs. The same idea can be extended to Perceptron too: the Kernel Perceptron