Chapter 3: Introduction to Classification

Reza Rezazadegan

Sharif University of Technology

November 8, 2022

Support Vector Machine (SVM)

- The idea behind SVM, for binary classification is to find a hyperplane that provides the largest margin between the two classes.

Support Vector Machine (SVM)

- The idea behind SVM, for binary classification is to find a hyperplane that provides the largest margin between the two classes.

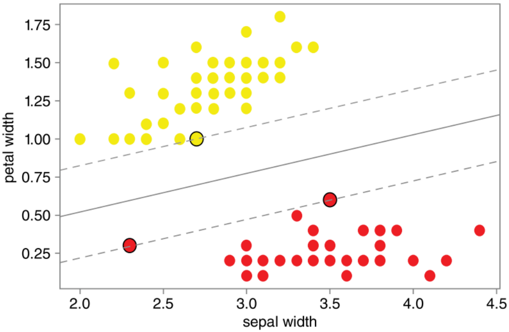


Figure: SVM for two classes in the Iris dataset.

Support Vector Machine (SVM)

- The idea behind SVM, for binary classification is to find a hyperplane that provides the largest margin between the two classes.

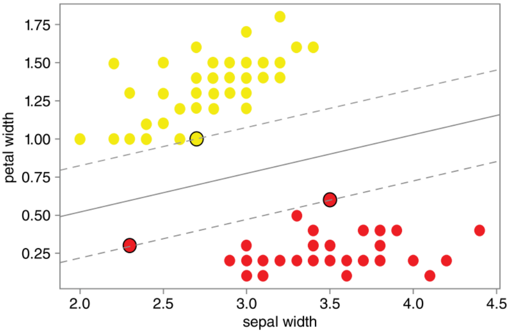


Figure: SVM for two classes in the Iris dataset.

- The points in either class that are closest to this hyperplane are called *support vectors*.

The math of SVM

- We represent the hyperplane we are looking for by

$$\langle \mathbf{x}, \mathbf{a} \rangle + a_0 = \mathbf{x}^t \mathbf{a} + a_0 = 0. \tag{1}$$

The math of SVM

- We represent the hyperplane we are looking for by

$$\langle \mathbf{x}, \mathbf{a} \rangle + a_0 = \mathbf{x}^t \mathbf{a} + a_0 = 0. \tag{1}$$

- The two classes are represented by $y = +1$ and $y = -1$.

The math of SVM

- We represent the hyperplane we are looking for by

$$\langle \mathbf{x}, \mathbf{a} \rangle + a_0 = \mathbf{x}^t \mathbf{a} + a_0 = 0. \quad (1)$$

- The two classes are represented by $y = +1$ and $y = -1$.
- We declare that an instance $\mathbf{x}$ belongs to the class -1 or 1 depending on whether it lies on the "positive" or "negative" side of this hyperplane;

The math of SVM

- We represent the hyperplane we are looking for by

$$\langle \mathbf{x}, \mathbf{a} \rangle + a_0 = \mathbf{x}^t \mathbf{a} + a_0 = 0. \tag{1}$$

- The two classes are represented by $y = +1$ and $y = -1$.
- We declare that an instance $\mathbf{x}$ belongs to the class -1 or 1 depending on whether it lies on the "positive" or "negative" side of this hyperplane; which is determined by the sign of $y = \langle \mathbf{x}, \mathbf{a} \rangle + a_0$.

The math of SVM

- We represent the hyperplane we are looking for by

$$\langle \mathbf{x}, \mathbf{a} \rangle + a_0 = \mathbf{x}^t \mathbf{a} + a_0 = 0. \tag{1}$$

- The two classes are represented by $y = +1$ and $y = -1$.
- We declare that an instance $\mathbf{x}$ belongs to the class -1 or 1 depending on whether it lies on the "positive" or "negative" side of this hyperplane; which is determined by the sign of $y = \langle \mathbf{x}, \mathbf{a} \rangle + a_0$.
- To have a margin between either class and the hyperplane, we require that for each datapoint $(\mathbf{x}_i, y_i)$,

$$y_i(\langle \mathbf{x}_i, \mathbf{a} \rangle + a_0) = |\langle \mathbf{x}_i, \mathbf{a} \rangle + a_0| \geq 1 \tag{2}$$

The math of SVM

- We represent the hyperplane we are looking for by

$$\langle \mathbf{x}, \mathbf{a} \rangle + a_0 = \mathbf{x}^t \mathbf{a} + a_0 = 0. \quad (1)$$

- The two classes are represented by $y = +1$ and $y = -1$.
- We declare that an instance $\mathbf{x}$ belongs to the class -1 or 1 depending on whether it lies on the "positive" or "negative" side of this hyperplane; which is determined by the sign of $y = \langle \mathbf{x}, \mathbf{a} \rangle + a_0$.
- To have a margin between either class and the hyperplane, we require that for each datapoint $(\mathbf{x}_i, y_i)$,

$$y_i(\langle \mathbf{x}_i, \mathbf{a} \rangle + a_0) = |\langle \mathbf{x}_i, \mathbf{a} \rangle + a_0| \geq 1 \quad (2)$$

  For support vectors, this inequality is an equality.
- The distance between a point $\mathbf{x}$ and this hyperplane is given by

The math of SVM

- We represent the hyperplane we are looking for by
$$\langle \mathbf{x}, \mathbf{a} \rangle + a_0 = \mathbf{x}^t \mathbf{a} + a_0 = 0. \quad (1)$$

- The two classes are represented by $y = +1$ and $y = -1$.
- We declare that an instance $\mathbf{x}$ belongs to the class -1 or 1 depending on whether it lies on the "positive" or "negative" side of this hyperplane; which is determined by the sign of $y = \langle \mathbf{x}, \mathbf{a} \rangle + a_0$.
- To have a margin between either class and the hyperplane, we require that for each datapoint $(\mathbf{x}_i, y_i)$,
$$y_i(\langle \mathbf{x}_i, \mathbf{a} \rangle + a_0) = |\langle \mathbf{x}_i, \mathbf{a} \rangle + a_0| \geq 1 \quad (2)$$
For support vectors, this inequality is an equality.
- The distance between a point $\mathbf{x}$ and this hyperplane is given by $\frac{|\langle \mathbf{x}, \mathbf{a} \rangle + a_0|}{||\mathbf{a}||}$.

The math of SVM

- We represent the hyperplane we are looking for by
$$\langle \mathbf{x}, \mathbf{a} \rangle + a_0 = \mathbf{x}^t \mathbf{a} + a_0 = 0. \tag{1}$$
- The two classes are represented by $y = +1$ and $y = -1$.
- We declare that an instance $\mathbf{x}$ belongs to the class -1 or 1 depending on whether it lies on the "positive" or "negative" side of this hyperplane; which is determined by the sign of $y = \langle \mathbf{x}, \mathbf{a} \rangle + a_0$.
- To have a margin between either class and the hyperplane, we require that for each datapoint $(\mathbf{x}_i, y_i)$,
$$y_i(\langle \mathbf{x}_i, \mathbf{a} \rangle + a_0) = |\langle \mathbf{x}_i, \mathbf{a} \rangle + a_0| \geq 1 \tag{2}$$
For support vectors, this inequality is an equality.
- The distance between a point $\mathbf{x}$ and this hyperplane is given by $\frac{|\langle \mathbf{x}, \mathbf{a} \rangle + a_0|}{||\mathbf{a}||}$.
Therefore the minimum distance from the points in either class to the hyperplane equals $||\mathbf{a}||^{-1}$.

Solving SVM optimization problem

- We thus want to maximize $||\mathbf{a}||^{-1}$ (or equivalently minimize $||\mathbf{a}||^2$) subject to the constraints of (2).

Solving SVM optimization problem

- We thus want to maximize $||\mathbf{a}||^{-1}$ (or equivalently minimize $||\mathbf{a}||^2$) subject to the constraints of (2).
- These constraints determine a subset of $\mathbb{R}^d$ which is bounded by the n hyperplanes given in (2).

Solving SVM optimization problem

- We thus want to maximize $||\mathbf{a}||^{-1}$ (or equivalently minimize $||\mathbf{a}||^2$) subject to the constraints of (2).
- These constraints determine a subset of $\mathbb{R}^d$ which is bounded by the n hyperplanes given in (2).
- To solve this problem, remember the **method of Lagrange multipliers**: the extreme points of $F(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$ occur at points where $\nabla F = \lambda \nabla g$ for some constant λ.

Solving SVM optimization problem

- We thus want to maximize $||\mathbf{a}||^{-1}$ (or equivalently minimize $||\mathbf{a}||^2$) subject to the constraints of (2).
- These constraints determine a subset of $\mathbb{R}^d$ which is bounded by the n hyperplanes given in (2).
- To solve this problem, remember the **method of Lagrange multipliers**: the extreme points of $F(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$ occur at points where $\nabla F = \lambda \nabla g$ for some constant λ.
- Equivalently we can consider $L(\mathbf{x}, \lambda) = f(x) - \lambda g(\mathbf{x})$ and find the extremal points of L.

Solving SVM optimization problem

- We thus want to maximize $||\mathbf{a}||^{-1}$ (or equivalently minimize $||\mathbf{a}||^2$) subject to the constraints of (2).
- These constraints determine a subset of $\mathbb{R}^d$ which is bounded by the n hyperplanes given in (2).
- To solve this problem, remember the **method of Lagrange multipliers**: the extreme points of $F(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$ occur at points where $\nabla F = \lambda \nabla g$ for some constant λ.
- Equivalently we can consider $L(\mathbf{x}, \lambda) = f(x) - \lambda g(\mathbf{x})$ and find the extremal points of L. If we have multiple constraints $\mathcal{C} = \{\mathbf{x} | g_1(\mathbf{x}) = g_2(\mathbf{x}) = \cdots = g_k(\mathbf{x}) = 0\}$ then at the extremal points, ∇f is perpendicular to $\mathcal{C}$ i.e. ∇F lies in the span of $\{\nabla g_i(\mathbf{x})\}$.

Solving SVM optimization problem

- We thus want to maximize $||\mathbf{a}||^{-1}$ (or equivalently minimize $||\mathbf{a}||^2$) subject to the constraints of (2).
- These constraints determine a subset of $\mathbb{R}^d$ which is bounded by the n hyperplanes given in (2).
- To solve this problem, remember the **method of Lagrange multipliers**: the extreme points of $F(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$ occur at points where $\nabla F = \lambda \nabla g$ for some constant λ.
- Equivalently we can consider $L(\mathbf{x}, \lambda) = f(x) - \lambda g(\mathbf{x})$ and find the extremal points of L. If we have multiple constraints $\mathcal{C} = \{\mathbf{x} | g_1(\mathbf{x}) = g_2(\mathbf{x}) = \cdots = g_k(\mathbf{x}) = 0\}$ then at the extremal points, ∇f is perpendicular to $\mathcal{C}$ i.e. ∇F lies in the span of $\{\nabla g_i(\mathbf{x})\}$.
- This is equivalent to $\nabla(\mathbf{x}) = \sum_i \lambda_i \nabla g_i(\mathbf{x})$.

Solving SVM optimization problem

- We thus want to maximize $||\mathbf{a}||^{-1}$ (or equivalently minimize $||\mathbf{a}||^2$) subject to the constraints of (2).
- These constraints determine a subset of $\mathbb{R}^d$ which is bounded by the n hyperplanes given in (2).
- To solve this problem, remember the **method of Lagrange multipliers**: the extreme points of $F(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$ occur at points where $\nabla F = \lambda \nabla g$ for some constant λ.
- Equivalently we can consider $L(\mathbf{x}, \lambda) = f(x) - \lambda g(\mathbf{x})$ and find the extremal points of L. If we have multiple constraints $\mathcal{C} = \{\mathbf{x} | g_1(\mathbf{x}) = g_2(\mathbf{x}) = \cdots = g_k(\mathbf{x}) = 0\}$ then at the extremal points, ∇f is perpendicular to $\mathcal{C}$ i.e. ∇F lies in the span of $\{\nabla g_i(\mathbf{x})\}$.
- This is equivalent to $\nabla(\mathbf{x}) = \sum_i \lambda_i \nabla g_i(\mathbf{x})$.
- Equivalently consider $L(\mathbf{x}, \Lambda) = f(\mathbf{x}) - \sum_i \lambda_i g_i(\mathbf{x})$ and set $\nabla L = 0$.

Solving SVM optimization problem cont.

- However for SVM we have inequality constrains of the form $g_i(\mathbf{a}) \leq 0$.

Solving SVM optimization problem cont.

- However for SVM we have inequality constrains of the form $g_i(\mathbf{a}) \leq 0$.
- We introduce *slack variables* s_i and set $g_i(\mathbf{a}) + s_i^2 = 0$. This turns the inequality constraints to equality constraints.

Solving SVM optimization problem cont.

- However for SVM we have inequality constrains of the form $g_i(\mathbf{a}) \leq 0$.
- We introduce *slack variables* s_i and set $g_i(\mathbf{a}) + s_i^2 = 0$. This turns the inequality constraints to equality constraints. We thus consider the Lagrangian

$$L(\mathbf{a}, \Lambda) = \frac{1}{2}||\mathbf{a}||^2 + \sum_{i=1}^{n} \lambda_i (g_i(\mathbf{a}) + s_i^2) \tag{3}$$

  where $g_i(\mathbf{a}) = 1 - y_i(\mathbf{a}^t \mathbf{x}_i + a_0)$.
- The λ_i are non-negative and thus the summands on RHS are positive only if $\mathbf{x}_i$ is inside the margin i.e. $g_i(\mathbf{a}) < 0$.

Solving SVM optimization problem cont.

- However for SVM we have inequality constrains of the form $g_i(\mathbf{a}) \leq 0$.
- We introduce *slack variables* s_i and set $g_i(\mathbf{a}) + s_i^2 = 0$. This turns the inequality constraints to equality constraints. We thus consider the Lagrangian

$$L(\mathbf{a}, \Lambda) = \frac{1}{2}||\mathbf{a}||^2 + \sum_{i=1}^{n} \lambda_i (g_i(\mathbf{a}) + s_i^2) \tag{3}$$

  where $g_i(\mathbf{a}) = 1 - y_i(\mathbf{a}^t \mathbf{x}_i + a_0)$.
- The λ_i are non-negative and thus the summands on RHS are positive only if $\mathbf{x}_i$ is inside the margin i.e. $g_i(\mathbf{a}) < 0$.
- At the minimum $(\mathbf{a}_0, \Lambda_0)$ we have $\nabla L(\mathbf{a}_0, \Lambda_0) = 0$ and $g_i(\mathbf{a}_0) \leq 0$ for each i.

Solving SVM optimization problem cont.

- However for SVM we have inequality constrains of the form $g_i(\mathbf{a}) \leq 0$.
- We introduce *slack variables* s_i and set $g_i(\mathbf{a}) + s_i^2 = 0$. This turns the inequality constraints to equality constraints. We thus consider the Lagrangian

$$L(\mathbf{a}, \Lambda) = \frac{1}{2}||\mathbf{a}||^2 + \sum_{i=1}^{n} \lambda_i (g_i(\mathbf{a}) + s_i^2) \tag{3}$$

  where $g_i(\mathbf{a}) = 1 - y_i(\mathbf{a}^t \mathbf{x}_i + a_0)$.
- The λ_i are non-negative and thus the summands on RHS are positive only if $\mathbf{x}_i$ is inside the margin i.e. $g_i(\mathbf{a}) < 0$.
- At the minimum $(\mathbf{a}_0, \Lambda_0)$ we have $\nabla L(\mathbf{a}_0, \Lambda_0) = 0$ and $g_i(\mathbf{a}_0) \leq 0$ for each i. However we also have more important conditions known as *complementary slackness*:

$$\lambda_i g_i(\mathbf{a}_0) = \lambda_i [1 - y_i(\mathbf{a}_0^t \mathbf{x}_i + a_0)] = 0 \tag{4}$$

  for every i.

Solving SVM optimization problem cont.

- However for SVM we have inequality constrains of the form $g_i(\mathbf{a}) \leq 0$.
- We introduce *slack variables* s_i and set $g_i(\mathbf{a}) + s_i^2 = 0$. This turns the inequality constraints to equality constraints. We thus consider the Lagrangian

$$L(\mathbf{a}, \Lambda) = \frac{1}{2}||\mathbf{a}||^2 + \sum_{i=1}^{n} \lambda_i (g_i(\mathbf{a}) + s_i^2) \tag{3}$$

  where $g_i(\mathbf{a}) = 1 - y_i(\mathbf{a}^t \mathbf{x}_i + a_0)$.
- The λ_i are non-negative and thus the summands on RHS are positive only if $\mathbf{x}_i$ is inside the margin i.e. $g_i(\mathbf{a}) < 0$.
- At the minimum $(\mathbf{a}_0, \Lambda_0)$ we have $\nabla L(\mathbf{a}_0, \Lambda_0) = 0$ and $g_i(\mathbf{a}_0) \leq 0$ for each i. However we also have more important conditions known as *complementary slackness*:

$$\lambda_i g_i(\mathbf{a}_0) = \lambda_i [1 - y_i(\mathbf{a}_0^t \mathbf{x}_i + a_0)] = 0 \tag{4}$$

  for every i.

Solving SVM optimization problem cont.

- Complementary slackness follows from the equivalence of primal and dual problems for convex functions.

Solving SVM optimization problem cont.

- Complementary slackness follows from the equivalence of primal and dual problems for convex functions.
- Intuitively, at a point $\mathbf{a}_0$ if for some k, $g_k(\mathbf{a}_0) < 0$ and $\lambda_k \neq 0$ then consider $\mathbf{a}_1$ such that $g_k(\mathbf{a}_1) > g_k(\mathbf{a}_0)$ and thus we can find a point $\mathbf{a}_t$ near $\mathbf{a}_1$ such that $L(\mathbf{a}_t, \Lambda) > L(\mathbf{a}_0, \Lambda)$.

Solving SVM optimization problem cont.

- Complementary slackness follows from the equivalence of primal and dual problems for convex functions.
- Intuitively, at a point $\mathbf{a}_0$ if for some k, $g_k(\mathbf{a}_0) < 0$ and $\lambda_k \neq 0$ then consider $\mathbf{a}_1$ such that $g_k(\mathbf{a}_1) > g_k(\mathbf{a}_0)$ and thus we can find a point $\mathbf{a}_t$ near $\mathbf{a}_1$ such that $L(\mathbf{a}_t, \Lambda) > L(\mathbf{a}_0, \Lambda)$. This implies that $\mathbf{a}_0$ is not an extremal point of L.

Solving SVM optimization problem cont.

- Complementary slackness follows from the equivalence of primal and dual problems for convex functions.
- Intuitively, at a point $\mathbf{a}_0$ if for some k, $g_k(\mathbf{a}_0) < 0$ and $\lambda_k \neq 0$ then consider $\mathbf{a}_1$ such that $g_k(\mathbf{a}_1) > g_k(\mathbf{a}_0)$ and thus we can find a point $\mathbf{a}_t$ near $\mathbf{a}_1$ such that $L(\mathbf{a}_t, \Lambda) > L(\mathbf{a}_0, \Lambda)$. This implies that $\mathbf{a}_0$ is not an extremal point of L.
- It follows that if $\lambda_i \neq 0$ then $y_i(\mathbf{a}_0^t \mathbf{x}_i - a_0) = 1$ and thus $s_i = 0$.

Solving SVM optimization problem cont.

- Complementary slackness follows from the equivalence of primal and dual problems for convex functions.
- Intuitively, at a point $\mathbf{a}_0$ if for some k, $g_k(\mathbf{a}_0) < 0$ and $\lambda_k \neq 0$ then consider $\mathbf{a}_1$ such that $g_k(\mathbf{a}_1) > g_k(\mathbf{a}_0)$ and thus we can find a point $\mathbf{a}_t$ near $\mathbf{a}_1$ such that $L(\mathbf{a}_t, \Lambda) > L(\mathbf{a}_0, \Lambda)$. This implies that $\mathbf{a}_0$ is not an extremal point of L.
- It follows that if $\lambda_i \neq 0$ then $y_i(\mathbf{a}_0^t \mathbf{x}_i - a_0) = 1$ and thus $s_i = 0$.
- Therefore, if we add or remove points to our dataset, the solution to SVM is not changed, as long as these new points are farther from the hyperplane than the support vectors and "on the correct side".

Soft Margin SVM

- What we described so far is called *hard-margin SVM*.

Soft Margin SVM

- What we described so far is called *hard-margin SVM*.
- In practice, the two classes may not be linearly separable.

Soft Margin SVM

- What we described so far is called *hard-margin SVM*.
- In practice, the two classes may not be linearly separable. Or they may be separable but have outliers.

Soft Margin SVM

- What we described so far is called *hard-margin SVM*.
- In practice, the two classes may not be linearly separable. Or they may be separable but have outliers.

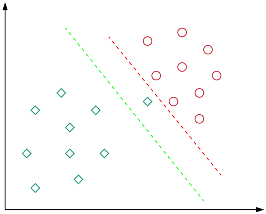


Figure: Hard margin SVM line is plotted in red. Credit: Rishabh Misra

Soft Margin SVM

- What we described so far is called *hard-margin SVM*.
- In practice, the two classes may not be linearly separable. Or they may be separable but have outliers.

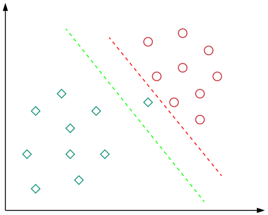


Figure: Hard margin SVM line is plotted in red. Credit: Rishabh Misra

- In **soft-margin SVM**, we remove the *strict* constrains of (2) and instead add a penalty term for each violation of the margin.

Soft Margin SVM

- What we described so far is called *hard-margin SVM*.
- In practice, the two classes may not be linearly separable. Or they may be separable but have outliers.

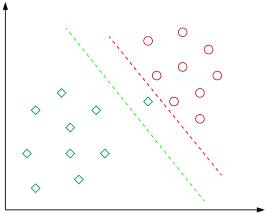


Figure: Hard margin SVM line is plotted in red. Credit: Rishabh Misra

- In **soft-margin SVM**, we remove the *strict* constrains of (2) and instead add a penalty term for each violation of the margin.
- The penalty is 0 if $y_i(\mathbf{a}^t\mathbf{x}_i + a_0) \geq 1$ and equals $1 - y_i(\mathbf{a}^t\mathbf{x}_i + a_0)$ otherwise.

Soft Margin SVM

- What we described so far is called *hard-margin SVM*.
- In practice, the two classes may not be linearly separable. Or they may be separable but have outliers.

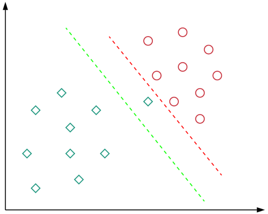


Figure: Hard margin SVM line is plotted in red. Credit: Rishabh Misra

- In **soft-margin SVM**, we remove the *strict* constrains of (2) and instead add a penalty term for each violation of the margin.
- The penalty is 0 if $y_i(\mathbf{a}^t\mathbf{x}_i + a_0) \geq 1$ and equals $1 - y_i(\mathbf{a}^t\mathbf{x}_i + a_0)$ otherwise. In short, the penalty is given by $\max(0, 1 - y_i(\mathbf{a}^t\mathbf{x}_i - a_0))$.

Soft Margin SVM cont.

- The soft-margin loss function is given by

$$||\mathbf{a}||^2 + C \cdot \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{a}^t \mathbf{x}_i + a_0)) \quad (5)$$

Soft Margin SVM cont.

- The soft-margin loss function is given by

$$||\mathbf{a}||^2 + C \cdot \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{a}^t \mathbf{x}_i + a_0)) \quad (5)$$

- We can divide this loss function by C, which shows that soft-margin SVM is actually a regularized form of SVM.

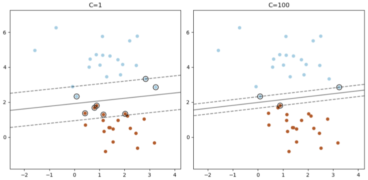


Figure: Hard-margin and soft-margin SVM

Soft Margin SVM cont.

- The soft-margin loss function is given by

$$||\mathbf{a}||^2 + C \cdot \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{a}^t \mathbf{x}_i + a_0)) \quad (5)$$

- We can divide this loss function by C, which shows that soft-margin SVM is actually a regularized form of SVM.

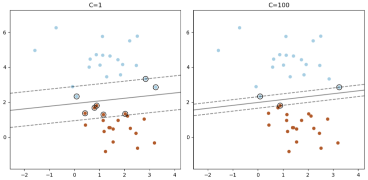


Figure: Hard-margin and soft-margin SVM

- In Python, linear SVM is given by the class `sklearn.svm.LinearSVC`

Gradient Descent for SVM

- The loss function for soft-margin SVM is convex.

Gradient Descent for SVM

- The loss function for soft-margin SVM is convex.
- It is differentiable except on the boundary of the margin.

Gradient Descent for SVM

- The loss function for soft-margin SVM is convex.
- It is differentiable except on the boundary of the margin.
- One can use its subgradient at each point to be able to use the method of Gradient Descent.

Gradient Descent for SVM

- The loss function for soft-margin SVM is convex.
- It is differentiable except on the boundary of the margin.
- One can use its subgradient at each point to be able to use the method of Gradient Descent.
- Exercise: find the subgradient vector for the loss function (8).

Gradient Descent for SVM

- The loss function for soft-margin SVM is convex.
- It is differentiable except on the boundary of the margin.
- One can use its subgradient at each point to be able to use the method of Gradient Descent.
- Exercise: find the subgradient vector for the loss function (8).
- This method is available as `sklearn.linear_model.SGDClassifier(loss='hinge')`

Gradient Descent for SVM

- The loss function for soft-margin SVM is convex.
- It is differentiable except on the boundary of the margin.
- One can use its subgradient at each point to be able to use the method of Gradient Descent.
- Exercise: find the subgradient vector for the loss function (8).
- This method is available as `sklearn.linear_model.SGDClassifier(loss='hinge')`
- One advantage of this method is the ability to use SGD which scales well with the size of the training dataset.

SVM for non-linearly separable classes

- Often the two classes cannot be separated by a hyperplane.

SVM for non-linearly separable classes

- Often the two classes cannot be separated by a hyperplane.
- In such cases we may hope that by adding more features, the two classes will become linearly separable.

SVM for non-linearly separable classes

- Often the two classes cannot be separated by a hyperplane.
- In such cases we may hope that by adding more features, the two classes will become linearly separable.
- This is akin to applying a map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ which maps our data to a higher-dimensional space.

SVM for non-linearly separable classes

- Often the two classes cannot be separated by a hyperplane.
- In such cases we may hope that by adding more features, the two classes will become linearly separable.
- This is akin to applying a map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ which maps our data to a higher-dimensional space.
- This can be done e.g. by adding nonlinear combinations of features, as new features.

SVM for non-linearly separable classes

- Often the two classes cannot be separated by a hyperplane.
- In such cases we may hope that by adding more features, the two classes will become linearly separable.
- This is akin to applying a map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ which maps our data to a higher-dimensional space.
- This can be done e.g. by adding nonlinear combinations of features, as new features. For example if our features are x, y, we can consider x^2, xy, y^2 as new features, which maps our data from $\mathbb{R}^2$ to $\mathbb{R}^3$.

SVM for non-linearly separable classes

- Often the two classes cannot be separated by a hyperplane.
- In such cases we may hope that by adding more features, the two classes will become linearly separable.
- This is akin to applying a map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ which maps our data to a higher-dimensional space.
- This can be done e.g. by adding nonlinear combinations of features, as new features. For example if our features are x, y, we can consider x^2, xy, y^2 as new features, which maps our data from $\mathbb{R}^2$ to $\mathbb{R}^3$.
- The SVM hyperplane in $\mathbb{R}^D$ has the form:

$$\langle \Phi(\mathbf{a}), \Phi(\mathbf{x}) \rangle + a_0 = 0 \tag{6}$$

SVM for non-linearly separable classes

- Often the two classes cannot be separated by a hyperplane.
- In such cases we may hope that by adding more features, the two classes will become linearly separable.
- This is akin to applying a map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ which maps our data to a higher-dimensional space.
- This can be done e.g. by adding nonlinear combinations of features, as new features. For example if our features are x, y, we can consider x^2, xy, y^2 as new features, which maps our data from $\mathbb{R}^2$ to $\mathbb{R}^3$.
- The SVM hyperplane in $\mathbb{R}^D$ has the form:

$$\langle \Phi(\mathbf{a}), \Phi(\mathbf{x}) \rangle + a_0 = 0 \tag{6}$$

- It is a linear hyperplane in $\mathbb{R}^D$ but is a nonlinear *hypersurface* in $\mathbb{R}^d$.

Kernel functions in nonlinear SVM

- To simplify the computations, we find a **kernel function** $K(\mathbf{x}, \mathbf{z})$ such that

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle = K(\mathbf{x}, \mathbf{z}) \tag{7}$$

Kernel functions in nonlinear SVM

- To simplify the computations, we find a **kernel function** $K(\mathbf{x}, \mathbf{z})$ such that

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle = K(\mathbf{x}, \mathbf{z}) \tag{7}$$

- For example if $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ and $\Phi(x_1, x_2, \ldots, x_d) = (x_i x_j)_{i,j}$ consist of all the quadratic terms in the x_i. Then we have $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z})) \rangle$.

Kernel functions in nonlinear SVM

- To simplify the computations, we find a **kernel function** $K(\mathbf{x}, \mathbf{z})$ such that

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle = K(\mathbf{x}, \mathbf{z}) \tag{7}$$

- For example if $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ and $\Phi(x_1, x_2, \ldots, x_d) = (x_i x_j)_{i,j}$ consist of all the quadratic terms in the x_i. Then we have $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z})) \rangle$.
- Another example of a kernel is the Radial Basis Function (RBF) kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-||\mathbf{x} - \mathbf{z}||^2). \tag{8}$$

Kernel functions in nonlinear SVM

- To simplify the computations, we find a **kernel function** $K(\mathbf{x}, \mathbf{z})$ such that

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle = K(\mathbf{x}, \mathbf{z}) \tag{7}$$

- For example if $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ and $\Phi(x_1, x_2, \dots, x_d) = (x_i x_j)_{i,j}$ consist of all the quadratic terms in the x_i. Then we have $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z})) \rangle$.
- Another example of a kernel is the Radial Basis Function (RBF) kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-||\mathbf{x} - \mathbf{z}||^2). \tag{8}$$

- To recap: we are looking for the widest-margin SVM hyperplane P in $\mathbb{R}^D$ for the transformed data $\{(\Phi(\mathbf{x}_i), y_i)\}$.

Kernel functions in nonlinear SVM

- To simplify the computations, we find a **kernel function** $K(\mathbf{x}, \mathbf{z})$ such that

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle = K(\mathbf{x}, \mathbf{z}) \tag{7}$$

- For example if $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ and $\Phi(x_1, x_2, \ldots, x_d) = (x_i x_j)_{i,j}$ consist of all the quadratic terms in the x_i. Then we have $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z})) \rangle$.
- Another example of a kernel is the Radial Basis Function (RBF) kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-||\mathbf{x} - \mathbf{z}||^2). \tag{8}$$

- To recap: we are looking for the widest-margin SVM hyperplane P in $\mathbb{R}^D$ for the transformed data $\{(\Phi(\mathbf{x}_i), y_i)\}$. This is equivalent to finding a *hypersurface* S in $\mathbb{R}^d$ of the form $K(\mathbf{a}, \mathbf{x}) + a_0 = 0$.

Kernel functions in nonlinear SVM

- To simplify the computations, we find a **kernel function** $K(\mathbf{x}, \mathbf{z})$ such that

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle = K(\mathbf{x}, \mathbf{z}) \tag{7}$$

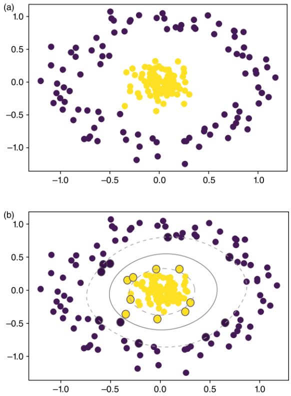
- For example if $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ and $\Phi(x_1, x_2, \dots, x_d) = (x_i x_j)_{i,j}$ consist of all the quadratic terms in the x_i. Then we have $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z})) \rangle$.
- Another example of a kernel is the Radial Basis Function (RBF) kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-||\mathbf{x} - \mathbf{z}||^2). \tag{8}$$

- To recap: we are looking for the widest-margin SVM hyperplane P in $\mathbb{R}^D$ for the transformed data $\{(\Phi(\mathbf{x}_i), y_i)\}$. This is equivalent to finding a *hypersurface* S in $\mathbb{R}^d$ of the form $K(\mathbf{a}, \mathbf{x}) + a_0 = 0$.
- Of course we have $\Phi(S) = P$.

Kernel functions in nonlinear SVM cont.

- An example of using the SVM with RBF kernel.



Pros and Cons of SVM

In general SVM optimization problem is solved using Sequential Minimal Optimization method which reduces the problem into a sequence of optimization problems involving only two Lagrange multipliers at each step.
See: John C. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, 1998.
Pros of SVM:

- SVM provides high classification accuracy.

Cons of SVM:

- It's not probabilistic.

Pros and Cons of SVM

In general SVM optimization problem is solved using Sequential Minimal Optimization method which reduces the problem into a sequence of optimization problems involving only two Lagrange multipliers at each step.
See: John C. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, 1998.
Pros of SVM:

- SVM provides high classification accuracy.

Cons of SVM:

- It's not probabilistic.
- It's sensitive to noise in the data.

Pros and Cons of SVM

In general SVM optimization problem is solved using Sequential Minimal Optimization method which reduces the problem into a sequence of optimization problems involving only two Lagrange multipliers at each step. See: John C. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, 1998.

Pros of SVM:

- SVM provides high classification accuracy.

Cons of SVM:

- It's not probabilistic.
- It's sensitive to noise in the data.
- Does not scale well with the size of the dataset.

SVM is

Pros and Cons of SVM

In general SVM optimization problem is solved using Sequential Minimal Optimization method which reduces the problem into a sequence of optimization problems involving only two Lagrange multipliers at each step. See: John C. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, 1998.

Pros of SVM:

- SVM provides high classification accuracy.

Cons of SVM:

- It's not probabilistic.
- It's sensitive to noise in the data.
- Does not scale well with the size of the dataset.

SVM is an instance-based method.

Support Vector Regression (SVR)

- In SVR, for a given dataset $\{(\mathbf{x}_i, y_i)\}$ we want to find a linear function f such that most of the data lies with an ϵ-margin of f.

Support Vector Regression (SVR)

- In SVR, for a given dataset $\{(\mathbf{x}_i, y_i)\}$ we want to find a linear function f such that most of the data lies with an ϵ-margin of f.
- This means that if $|f(\mathbf{x}_i) - y_i| < \epsilon$, there is no penalty, and otherwise the penalty is $|f(\mathbf{x}_i) - y_i| - \epsilon$.

Support Vector Regression (SVR)

- In SVR, for a given dataset $\{(\mathbf{x}_i, y_i)\}$ we want to find a linear function f such that most of the data lies with an ϵ-margin of f.
- This means that if $|f(\mathbf{x}_i) - y_i| < \epsilon$, there is no penalty, and otherwise the penalty is $|f(\mathbf{x}_i) - y_i| - \epsilon$. Here ϵ is a hyperparameter.

Support Vector Regression (SVR)

- In SVR, for a given dataset $\{(\mathbf{x}_i, y_i)\}$ we want to find a linear function f such that most of the data lies with an ϵ-margin of f.
- This means that if $|f(\mathbf{x}_i) - y_i| < \epsilon$, there is no penalty, and otherwise the penalty is $|f(\mathbf{x}_i) - y_i| - \epsilon$. Here ϵ is a hyperparameter.
- In SVR we want to find a linear $f(\mathbf{x})$ that minimizes the sum of the penalties.

Support Vector Regression (SVR)

- In SVR, for a given dataset $\{(\mathbf{x}_i, y_i)\}$ we want to find a linear function f such that most of the data lies with an ϵ-margin of f.
- This means that if $|f(\mathbf{x}_i) - y_i| < \epsilon$, there is no penalty, and otherwise the penalty is $|f(\mathbf{x}_i) - y_i| - \epsilon$. Here ϵ is a hyperparameter.
- In SVR we want to find a linear $f(\mathbf{x})$ that minimizes the sum of the penalties.
- In SVR the goal is to reduce the error to a certain range.

Support Vector Regression (SVR)

- In SVR, for a given dataset $\{(\mathbf{x}_i, y_i)\}$ we want to find a linear function f such that most of the data lies with an ϵ-margin of f.
- This means that if $|f(\mathbf{x}_i) - y_i| < \epsilon$, there is no penalty, and otherwise the penalty is $|f(\mathbf{x}_i) - y_i| - \epsilon$. Here ϵ is a hyperparameter.
- In SVR we want to find a linear $f(\mathbf{x})$ that minimizes the sum of the penalties.
- In SVR the goal is to reduce the error to a certain range. For example predicting house prices with at most $5000 error.

Support Vector Regression (SVR)

- In SVR, for a given dataset $\{(\mathbf{x}_i, y_i)\}$ we want to find a linear function f such that most of the data lies with an ϵ-margin of f.
- This means that if $|f(\mathbf{x}_i) - y_i| < \epsilon$, there is no penalty, and otherwise the penalty is $|f(\mathbf{x}_i) - y_i| - \epsilon$. Here ϵ is a hyperparameter.
- In SVR we want to find a linear $f(\mathbf{x})$ that minimizes the sum of the penalties.
- In SVR the goal is to reduce the error to a certain range. For example predicting house prices with at most \$5000 error.
- In hard-margin SVR we want to minimize $||\mathbf{a}||^2$ subject to $|\mathbf{a}^t\mathbf{x}_i + a_0 - y_i| < \epsilon$.

Support Vector Regression (SVR)

- In SVR, for a given dataset $\{(\mathbf{x}_i, y_i)\}$ we want to find a linear function f such that most of the data lies with an ϵ-margin of f.
- This means that if $|f(\mathbf{x}_i) - y_i| < \epsilon$, there is no penalty, and otherwise the penalty is $|f(\mathbf{x}_i) - y_i| - \epsilon$. Here ϵ is a hyperparameter.
- In SVR we want to find a linear $f(\mathbf{x})$ that minimizes the sum of the penalties.
- In SVR the goal is to reduce the error to a certain range. For example predicting house prices with at most \$5000 error.
- In hard-margin SVR we want to minimize $||\mathbf{a}||^2$ subject to $|\mathbf{a}^t\mathbf{x}_i + a_0 - y_i| < \epsilon$.
- In the soft-margin SVR we want to minimize $\frac{1}{2}||\mathbf{a}||^2 + K\sum_i \max(|\mathbf{a}^t\mathbf{x}_i + a_0 - y_i| - \epsilon, 0)$

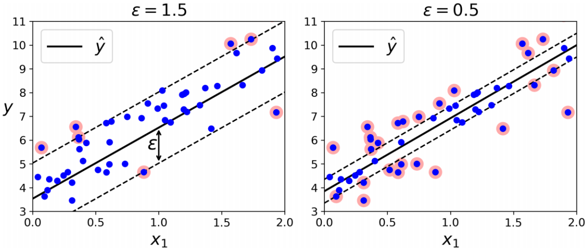


Figure: SVR for two different values of ϵ. Credit: A. Geron

- SVR can be used for nonlinear regression by replacing $\mathbf{a}^t\mathbf{x}$ with a kernel $K(\mathbf{a}, \mathbf{x})$!

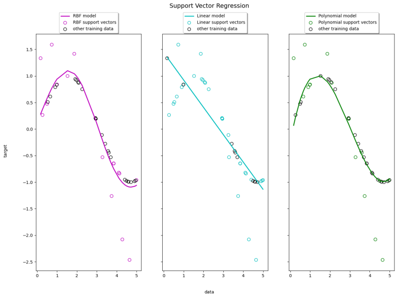


Figure: Kernel SVR for sinusoidal data (with added noise).

- Giving no penalty for when error is less than ϵ prevents unnecessary fluctuations of the predicted f.
- In Python (soft-margin) SVR is provided by the class `from sklearn.svm.SVR`

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods.

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods. Assume we have classes $\{1, 2, \ldots, m\}$

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods. Assume we have classes $\{1, 2, \ldots, m\}$
- **One-vs-All or One-vs-Rest:** for each class k we take a binary classifier that tries to separate class k from the union of the classes $i \neq k$.

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods. Assume we have classes $\{1, 2, \ldots, m\}$
- **One-vs-All or One-vs-Rest:** for each class k we take a binary classifier that tries to separate class k from the union of the classes $i \neq k$. This way we get k different classifiers.

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods. Assume we have classes $\{1, 2, \ldots, m\}$
- **One-vs-All or One-vs-Rest:** for each class k we take a binary classifier that tries to separate class k from the union of the classes $i \neq k$. This way we get k different classifiers.
- In this method, to a new instance, is assigned the class which has the highest classification score or probability.

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods. Assume we have classes $\{1, 2, \ldots, m\}$
- **One-vs-All or One-vs-Rest:** for each class k we take a binary classifier that tries to separate class k from the union of the classes $i \neq k$. This way we get k different classifiers.
- In this method, to a new instance, is assigned the class which has the highest classification score or probability. (In SVM, the confidence score for an instance is proportional to the signed distance of that sample to the hyperplane.

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods. Assume we have classes $\{1, 2, \ldots, m\}$
- **One-vs-All or One-vs-Rest:** for each class k we take a binary classifier that tries to separate class k from the union of the classes $i \neq k$. This way we get k different classifiers.
- In this method, to a new instance, is assigned the class which has the highest classification score or probability. (In SVM, the confidence score for an instance is proportional to the signed distance of that sample to the hyperplane.
- **One-vs-One:** for each pair of classes k, l we forget about the rest of classes and take the classifer that separates class k from class l.

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods. Assume we have classes $\{1, 2, \ldots, m\}$
- **One-vs-All or One-vs-Rest:** for each class k we take a binary classifier that tries to separate class k from the union of the classes $i \neq k$. This way we get k different classifiers.
- In this method, to a new instance, is assigned the class which has the highest classification score or probability. (In SVM, the confidence score for an instance is proportional to the signed distance of that sample to the hyperplane.
- **One-vs-One:** for each pair of classes k, l we forget about the rest of classes and take the classifer that separates class k from class l. We produce $m(m-1)/2$ classifiers.

Adapting a binary classifier for multi-class classification

- Any binary classifier, such as SVM, can be adapted for multi-class classification using one of the following methods. Assume we have classes $\{1, 2, \ldots, m\}$
- **One-vs-All or One-vs-Rest:** for each class k we take a binary classifier that tries to separate class k from the union of the classes $i \neq k$. This way we get k different classifiers.
- In this method, to a new instance, is assigned the class which has the highest classification score or probability. (In SVM, the confidence score for an instance is proportional to the signed distance of that sample to the hyperplane.
- **One-vs-One:** for each pair of classes k, l we forget about the rest of classes and take the classifer that separates class k from class l. We produce $m(m-1)/2$ classifiers.
- In this method the class that wins against the highest number of other classes, is assigned.

Adapting a binary classifier for multi-class classification, cont.

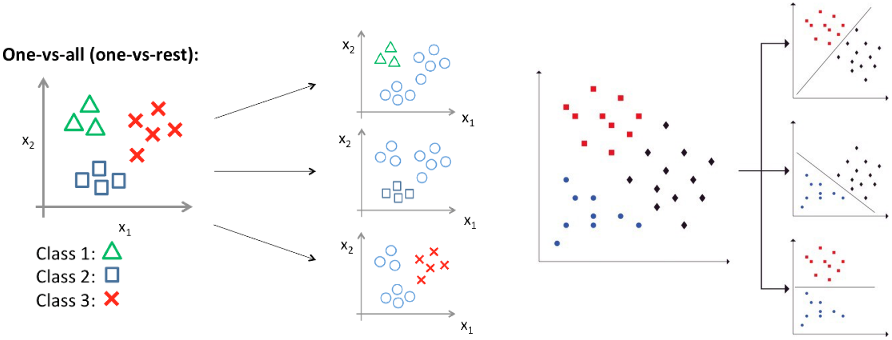


Figure: One-vs-All (left) and One-vs-One (right). Credit: Jatin Nanda, Zhongliang Zhang, *et al.*

Adapting a binary classifier for multi-class classification, cont.

- Scikit-Learn uses One-vs-All for most classification problems except for SVM

Adapting a binary classifier for multi-class classification, cont.

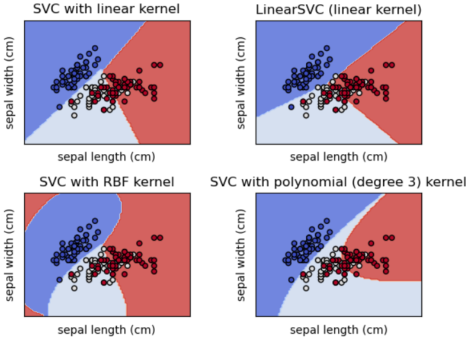
- Scikit-Learn uses One-vs-All for most classification problems except for SVM because the latter does not scale well with the size of the dataset.

Adapting a binary classifier for multi-class classification, cont.

- Scikit-Learn uses One-vs-All for most classification problems except for SVM because the latter does not scale well with the size of the dataset.
- Scikit's `svm.SVC` class uses One-vs-One method for multi-class classification, while `svm.LinearSVC` uses One-vs-All.

Adapting a binary classifier for multi-class classification, cont.

- Scikit-Learn uses One-vs-All for most classification problems except for SVM because the latter does not scale well with the size of the dataset.
- Scikit's `svm.SVC` class uses One-vs-One method for multi-class classification, while `svm.LinearSVC` uses One-vs-All.



SVM quiz

Consider the following dataset:

feature1	feature2	class
2	0	good
0	1	bad
1	1	good

Compute the equation of the hard-margin SVM hyperplane for this dataset.

Solution to the quiz

We represent the two features by x_1, x_2. We want to find the hyperplane (a line in this case) of the form $a_1x_1 + a_2x_2 + a_0 = 0$ which maximizes $a_1^2 + a_2^2$, subject to the constraints of (2).
To simplify the solution note that we can assume that either $a_2 = 0$ or $a_2 = 1$, since we can divide by a_2 if it's not zero. Thus we have two cases $a_0 + a_1x_1 + x_2 = 0$ or $a_0 + a_1x_1 = 0$. We solve the problem for each situation separately and compare the results.
In the first case, we want to minimize $1 + a_1^2$ (equivalently $|a_1|$) subject to
$a_0 + 2a_1 + 0 \geq 1$,
$a_0 + 0 + 1 \leq -1$,
$a_0 + a_1 + 1 \geq 1$.
The region in $\mathbb{R}^2$ satisfying the above condition is bounded by the three lines given by when the above inequalities are qualities i.e
$a_0 + 2a_1 = 1$, $a_0 = -2$, $a_0 + a_1 = 0$. Drawing these lines we realize that the point subject to the above inequalities at which $\min |a_1|$ is achieved is $(-2, 2)$. This gives $\min ||\mathbf{a}||^2 = 1 + 2^2 = 5$, for the first case.

Solution to the quiz

In the second case, $||\mathbf{a}|| = a_1^2$.
The constraints in this case are:
$a_0 + 2a_1 \geq 1$,
$a_0 + 0 \leq -1$,
$a_0 + a_1 \geq 1$.
From the 2nd and 3rd inequalities we get $1 \leq a_0 + a_1 \leq a_1 - 1$ which implies $a_1 \geq 2$. Thus the smallest value of a_1 is 2, for which $a_0 = -1$. (This is achieved because (-1,2) satisfies the above inequalities. You can also draw the region in this case.) At this point $||\mathbf{a}||^2 = 2^2 = 4$. This is smaller than the first case and thus the solution is given by $-1 + 2x_1 = 0$.

Problems

1. Let $\{(\mathbf{x}_i, y_i)\}$ be a dataset such that $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. Set P, N consist of $\mathbf{x}_i$ for which y_i is positive, and negative respectively Suppose there is a linear function $f(\mathbf{x})$ such that for each i we have $sign(f(\mathbf{x}_i)) = sign(y_i)$. Then show that there is an SVM hyperplane for the dataset i.e. a hyperplane $H \subset \mathbb{R}^d$ such that $d(P, H) := \min_{\mathbf{x} \in H}\{d(\mathbf{x}, H)\} = \min_{\mathbf{y} \in N}\{d(\mathbf{y}, H)\} =: d(N, H)$ and moreover for any other hypersurface H' either $d(P, H') > d(P, H)$ or $d(N, H') > d(N, P)$.
2. With the same assumptions as above, let $\mathbf{x}_k, \mathbf{x}_l$ be such that $y_k y_l = -1$ and that $d(\mathbf{x}_k, \mathbf{x}_l) \leq d(\mathbf{x}_i, \mathbf{x}_j)$ for any $\mathbf{x}_i \in P, \mathbf{x}_j \in N$. Is it true that $\mathbf{x}_k, \mathbf{x}_l$ are support vectors for H?
3. Show that the loss function for soft-margin SVM is convex and then find subgradients for it, at point where it is not differentiable.
4. What is the soft-margin penalty for a point that lies on the boundary of the margin but on the wrong side?