Support Vector Machine
Introduction
Hard Margin SVM: Primal Form

M. R. Hasan
Analogy based Learning
Readings

- Alpaydin: 10.3, 13.1, 13.2
- Murphy: 14.5.2.2
- Geron: chapter 5, appendix C
What We Will Cover

• Discriminative Approach: Binary Classification
• Support Vector Machine (SVM)
• Model: Inductive Hypothesis
• Various Models: Historical Perspective
• Algorithm: Unconstrained Optimization
• Primal & Dual Unconstrained Optimization
Generative vs. Discriminative

• There are **two different models** for solving classification problems:
  - Generative Model *(Approach 1)*
  - Discriminative Model *(Approach 2 & 3)*

Classification **goal**:
- Compute class probability $P(y \mid x)$ or
- A function that maps the features $(x)$ to the target class $(y)$
Generative vs. Discriminative

- **Generative Model (Approach 1)**
  - Estimate joint density \( p(X, Y) \)
  - Product rule: \( p(X, Y) = p(X \mid Y) \ p(Y) \)
  - First, assume some functional form for \( p(X \mid Y), p(Y) \)
  - Then, estimate \( p(X \mid Y) \) and \( p(Y) \)
  - Finally, use Bayes’ Rule to calculate posterior \( p(Y \mid X = x) \)
  - **Indirect computation** of class probability: \( P(Y \mid X) \)
  - Useful for generating samples, \( p(X) = \sum_y p(y) p(X \mid y) \)
Generative vs. Discriminative

• **Discriminative Model**
  - Estimate $P(Y \mid X)$ directly *(Approach 2)*
  - No need to measure $P(Y \mid X)$, instead learn the “discriminant” function $h(x)$ *(Approach 3)*
  - Direct but cannot obtain a sample of the data, because $p(X)$ is not available
Discriminative Approach

• We have discussed:
  • **Discriminative Model** (Approach 2: Logistic Regression).
  • Now we will discuss another discriminative model.

- The Approach 3 that uses **discriminant function** to solve classification problems.

<table>
<thead>
<tr>
<th>• Discriminative Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Estimate $P(Y \mid X)$ directly (<strong>Approach 2</strong>)</td>
</tr>
<tr>
<td>- Learn “discriminant” function $h(x)$ (<strong>Approach 3</strong>)</td>
</tr>
<tr>
<td>- Direct but cannot obtain a sample of the data, because $p(X)$ is not available</td>
</tr>
</tbody>
</table>
Linear Discrimination

• In this approach, we assume a model directly for the discriminant (decision boundary).

• We bypass the estimation of likelihoods or posteriors.
Linear Discrimination

• The discriminant-based approach makes an **assumption** on the **form** of the discriminant (decision boundary) between the classes.

• However, it **doesn’t make any assumption** (or require any knowledge) about the **densities**.

• For example, it doesn’t assume whether the density is Gaussian, or whether the inputs are correlated, and so forth.
Linear Discrimination

• Let’s consider a simple classification problem.
• Assume that we have a linearly separable dataset belonging to two classes: +(red) and -(blue)
• To separate these two classes we need to learn a decision boundary (hyperplane), i.e., a discriminant function.
Linear Discrimination

• The problem is: there are many possible decision boundaries for linearly separable data.
• Not just one, two, or three…
Linear Discrimination

• Indeed, there are many possible decision boundaries.
• How do we make the best separation possible?

Why does it even matter to find the best separation?
Linear Discrimination

• Consider the following **binary classification problem**.
• The plot shows the decision boundaries of **three possible linear classifiers** (red, magenta and green).

Why does it even matter to find the best separation?
Linear Discrimination

• The model whose decision boundary is represented by the green dashed line is so bad that it does not even separate the classes properly.
Linear Discrimination

• The other two models (red and magenta lines) work perfectly on this **training set**.

• But their **decision boundaries come so close** to the instances that these models will probably **not perform as well on new instances**, i.e., **test set**.
Linear Discrimination

- For **better generalization** we want a model (black decision boundary) as follows.
Linear Discrimination

• Its decision boundary not only separates the two classes but also stays as far away from the closest training instances as possible.

• Thus, for better generalizability the solution is to create a widest street between the two classes.
Linear Discrimination

- The “widest street” approach maximizes the separation between the two classes.
Linear Discrimination

**•** In other words, we choose a decision boundary that **has the largest margin** (dotted line).

**•** **Previous problem:** find a decision boundary (e.g., Logistic Regression)

**•** **Modified problem:** find **the** largest margin decision boundary
Linear Discrimination

• To solve the **largest margin classification** problem we will use one of the most powerful and versatile ML model.

• It’s called the **Support Vector Machine (SVM)**.
SVM & two Vladimiris (Lenin & Vapnik)!

Vapnik–Chervonenkis Theory
Computational Learning Theory, Vladimir Vapnik
Support Vector Machine

- SVM is a **discriminant-based** model.
- It is unlike what we did in a likelihood-based scheme that has **implicit parameters** in defining the likelihood densities.
- SVM uses a **different inductive bias**:

  Instead of making an assumption on the form of the class densities, SVM makes an assumption on the **form of the boundaries** separating classes.
Support Vector Machine

• **Vapnik** is the chief advocate of this discriminant-based approach.

• He argues that:

  *Estimating the class densities is a harder problem than estimating the class discriminants.*

• So, it does not make sense to solve a hard problem to solve an easier problem.

This is of course true only when the discriminant can be approximated by a **simple function**.
Support Vector Machine

- SVM is capable of performing **linear or nonlinear classification**, regression, and even **outlier detection**.
- It is one of the **most popular models** in Machine Learning.

SVMs are particularly well suited for classification of **complex** but **small- or medium-sized datasets**.
Support Vector Machine

• The **main idea** of the SVM is that the decision boundary is fully **determined (or supported) by the data points on the edge** of the street.
• These instances are called the **support vectors**.

---

First we will provide a **brief outline** of different SVM models.
Linearly Separable Data

No outlier:

Outlier:

Linearly Non-Separable Data
Linearly Separable Data

Vapnik & Chervonenkis, 1963

Outlier:

Soft Margin Classifier

Corinna Cortes & Vapnik, 1993

No outlier:

Hard Margin Classifier

\[ \min_{w,b} \frac{1}{2} \lVert w \rVert^2 \equiv \frac{1}{2} w^T w \]

subject to \( y_i (w^T \hat{x}_i + b) \geq 1 \forall i \)

\[ \min_{\hat{w},b} \frac{1}{2} \hat{w}^T \hat{w} + C \sum_{i=1}^{N} \xi_i \]

subject to \( y_i (\hat{w}^T \hat{x}_i + b) \geq 1 - \xi_i \forall i \)

\[ \xi_i \geq 0 \]

\[ \min_{\hat{w},b} \frac{1}{2} \hat{w}^T \hat{w} + C \sum_{i=1}^{N} h(y_i (\hat{w}^T \hat{x}_i + b)) \]

\[ h(z) = \max \{0, 1 - z\} \]
Linearly Non-Separable Data

**Feature Augmentation:**
Map data onto a higher-dimensional space

Increased dimension of the feature space increases the complexity!

As a solution use **Kernel SVM**

Solve the margin maximization problem on a high-dimensional feature space *without actually adding* the features (*kernel trick*).

Bernhard Boser, Isabelle Guyon and Vladimir Vapnik, 1992
Linearly Non-Separable Data

**Kernel SVM**

Solve the margin maximization problem on a high-dimensional feature space *without actually adding* the features (*kernel trick*).

**Polynomial Kernel**: new features are the polynomial and interaction terms of the existing features.

**Gaussian Radial Basis Function (RBF) Kernel**: new features are created based on the *similarity* with the existing features.

**Analogy-based learning**
Linearly Separable Data

Hard Margin Classifier

No outlier:

Vapnik & Chervonenkis, 1963

Linearly Non-Separable Data

Kernel SVM

Outlier:

Soft Margin Classifier

Corinna Cortes & Vapnik, 1993

Bernhard Boser, Isabelle Guyon and Vladimir Vapnik, 1992
Support Vector Machine

- For an **empirical understanding** of the elegant and beautiful world of Support Vector Machine classifiers go through the Jupyter notebooks in my Github repository “*Support Vector Machine Classifier-Beginner’s Survival Kit*”.
- These notebooks are complimentary to my slides.
SVM: Linear Classifier - Hard Margin

Vapnik & Chervonenkis, 1963

Linearly Separable Data
Support Vector Machine

• The main idea of the SVM is that the decision boundary is fully determined (or supported) by the data points on the edge of the street.
• These instances are called the support vectors.
• Notice that adding more training instances “off the street” will not affect the decision boundary at all!

We will first discuss linear SVM classification.
Linear SVM Classification

- There would be **two possible scenarios** in linear SVM classification.
- In our decision rule, we can strictly impose that **all instances be off the street and on the right side**.
- This is called **hard margin** classification.
- The flexible version of it is called **soft margin** classification.

![Graph showing hard and soft margin SVM](image)

We will first discuss **hard margin linear SVM classification**.
Hard Margin Linear SVM Classification

• To learn the large margin decision boundary for classification, we need to create a decision rule.

Let’s define the decision rule first.

We will construct the decision rule in steps, starting from an intuitive one.
Hard Margin Linear SVM Classification

- Consider the following linearly separable dataset.
- We have a **two-class problem** and there are two features $x_1$ and $x_2$.

To find the linear decision boundary, first find a **vector** $w$ that is perpendicular.

We don’t know anything about the **length of** $w$ yet.
Hard Margin Linear SVM Classification

• Why did we choose vector $w$ to be perpendicular to the decision boundary?
• Because it will be used by each data point to **measure the perpendicular distance from the decision boundary**.
Hard Margin Linear SVM Classification

• Let’s say that we have an unknown point “x”, and we want to decide on which side of the street it belongs to.
• To answer this question, we can take a projection of x on w.

We get a scalar value (from $w^Tx$) that will be proportional to the length along the direction of w.

Based on that length we can decide whether x is + or -.
Hard Margin Linear SVM Classification

• In other words, we want to know whether the dot product of \( x \) and \( w \) is \textbf{greater than some constant} \( c \): \( w^T x \geq c \)

• If so, then the \( x \) belong to class “+”

\[
\overrightarrow{w}^T \vec{x} \geq c
\]

This is our Decision Rule
Hard Margin Linear SVM Classification

- Let’s **modify the decision rule** by separately writing the intercept/bias/offset term.

\[ \vec{w}^T \vec{x} + b \geq c \]

*Without loss of generality*

\[ \vec{w}^T \vec{x} + b \geq 0 \quad \text{Then, class} = + \]

Here “b” is the intercept/bias that we write separately in SVM

This is our (new) Decision Rule
Hard Margin Linear SVM Classification

• We don’t know anything about $w$ (or $b$) except that $w$ is perpendicular to the decision boundary.

• Note that there could be many $w$’s and $b$’s (of different length) that are perpendicular to the decision boundary.

Our goal is to learn the vector $w$ and $b$ to make decisions about the data points.

Without loss of generality

$\overrightarrow{w}.\overrightarrow{x} + b \geq 0 \quad \text{Then, class} = +$
Hard Margin Linear SVM Classification

- So, to achieve our goal of learning the parameters $w$ and $b$, we can **formulate this classification problem** as a **constrained optimization problem**.

\[
\begin{align*}
\max_{\tilde{w}, b} & \quad \text{Margin} \\
\text{subject to} & \quad \text{Correct Classification on training data}
\end{align*}
\]

\[
\begin{align*}
\max_{\tilde{w}, b} & \quad \text{Margin} \\
\text{subject to} & \quad \tilde{w}^T.\tilde{x} + b \geq 0 \quad \text{Then, class} = +
\end{align*}
\]
SVM Algorithm: Constrained Optimization

\[
\max_{\mathbf{w}, b} \text{Margin} \\
\text{subject to } \text{Correct Classification on training data}
\]

\[
\max_{\mathbf{w}, b} \text{Margin} \\
\text{subject to } \mathbf{w}^T \mathbf{x} + b \geq 0 \text{ Then, class } = +
\]
Hard Margin Linear SVM Classification

- We use $y$ to denote the label of two classes.
- For convenience: two classes are labeled as $+1$ (for positive samples) and $-1$ (for negative samples).
- Then, our decision rule is given by:

$$\bar{w}^T \hat{x}_i + b \geq 0 \quad \forall y_i = +1$$

$$\bar{w}^T \hat{x}_i + b < 0 \quad \forall y_i = -1$$

We may combine these two rules:

$$y_i(\bar{w}^T \hat{x}_i + b) \geq 0 \quad \forall i$$

Optimization Problem

$$\max_{\bar{w}, b} \text{Margin}$$

subject to

$$y_i(\bar{w}^T \hat{x}_i + b) \geq 0 \quad \forall i$$
Hard Margin Linear SVM Classification

• So far, we presented our decision rule based on the **decision boundary**.
• At the decision boundary (dotted line): \( w^T x + b = 0 \)
• For maximizing the margin, we need to **define the margin** mathematically.

\[
\begin{align*}
\text{max}_{\vec{w}, b} & \quad \text{Margin} \\
\text{subject to} & \quad y_i (\vec{w}^T \vec{x}_i + b) \geq 0 \quad \forall i
\end{align*}
\]
Hard Margin Linear SVM Classification

• The two opposite margins are some constant “r” distance away from the boundary.
• So, an unknown point “x” is +/-, if it’s +r/-r distance away from the decision boundary.

Note, “r” is arbitrary
Hard Margin Linear SVM Classification

• We can always **scale both** \( w \) **and** \( b \) **to get an arbitrary** \( r \).
• So for convenience, let’s fix it as: \( r = +/-1 \)
Hard Margin Linear SVM Classification

Previous decision rule (no margin):

\[ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0 \quad \forall i \]

New Decision Rule (margin added):

\[ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \forall i \]

On the margin:

\[ y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1 \quad \forall i \]
Hard Margin Linear SVM Classification

• Thus, the optimization problem (margin maximization problem) can be formulated as follows.

\[
\begin{align*}
\max_{w,b} & \quad \text{Margin} \\
\text{subject to} & \quad y_i(w^T \tilde{x}_i + b) \geq 1 \quad \forall i
\end{align*}
\]

Decision Rule:

\[
y_i(w^T \tilde{x}_i + b) \geq 1 \quad \forall i
\]
Hard Margin Linear SVM Classification

• The **margin maximization problem** poses the following question: how do we **maximize the width** of the boundary?

• First, let’s mathematically define the **margin** (**width of the street**).
Hard Margin Linear SVM Classification

- Consider two points (+ and -) **on the two decision boundaries**.
- These are represented by vectors $x^+$ and $x^-$.
- The **distance** between $x^+$ and $x^-$ is: $(x^+ - x^-)$. 
Hard Margin Linear SVM Classification

• Then, the **width of the street** is given by the **dot product between** \((x^+ - x^-)\) and a **unit normal vector** to the decision boundary.

\[
\text{Width} = (\hat{x}^+ - \hat{x}^-). \left( \frac{\vec{w}}{||\vec{w}||} \right)
\]

\[
\text{Width} = \frac{\vec{w}^T \hat{x}^+ - \vec{w}^T \hat{x}^-}{||\vec{w}||}
\]
Hard Margin Linear SVM

- Since $x^+$ and $x^-$ are on the margin, we can use the following relation to derive a convenient expression for width in terms of $w$.

\[ y_i (\mathbf{w}^T \hat{x}_i + b ) = 1 \quad \forall i \]

Hence:

- For $\hat{x}^+$, $y = 1$: $\mathbf{w}^T \hat{x}^+ = 1 - b$
- For $\hat{x}^-$, $y = -1$: $\mathbf{w}^T \hat{x}^- = -1 - b$

\[
\text{Width} = (\hat{x}^+ - \hat{x}^-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}
\]

\[
\text{Width} = \frac{\mathbf{w}^T \hat{x}^+ - \mathbf{w}^T \hat{x}^-}{\|\mathbf{w}\|} = \frac{1 - b + 1 + b}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}
\]

Margin = $\frac{2}{\|w\|}$
Hard Margin Linear SVM

• We state the large margin maximization problem as:

$$\max_{w,b} \frac{2}{||w||}$$

subject to  \( y_i (\overrightarrow{w}^T \overrightarrow{x_i} + b) \geq 1 \)

Here \( \frac{b}{||w||} \) determines the offset of the hyperplane from the origin along the normal vector \( w \).
Hard Margin Linear SVM

\[
\max_{\mathbf{w}, b} \frac{2}{\|\mathbf{w}\|}
\]

subject to \( y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \)

- For convenience we recast the maximization problem as a minimization problem as follows:

\[
\max_{\mathbf{w}, b} \frac{2}{\|\mathbf{w}\|} = \max_{\mathbf{w}, b} \frac{1}{\|\mathbf{w}\|}
\]

\(\|\mathbf{w}\|\) is not differentiable at \(\mathbf{w} = 0\).

Optimization algorithms work much better on differentiable functions.

Notice that \(\frac{1}{2}\|\mathbf{w}\|^2\) has a nice and simple derivative (it is just \(\mathbf{w}\)).
Hard Margin Linear SVM Classification

• Finally, we have derived a constrained minimization problem:

\[
\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \equiv \frac{1}{2} \mathbf{w}^T \mathbf{w}
\]

subject to \( y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \ \forall i \)

This form of the constrained optimization problem is known as the primal form.

Notice that \( \mathbf{w}^T \mathbf{w} \) is quadratic, hence a convex function.

How is \( \mathbf{w}^T \mathbf{w} \) convex?
Hard Margin Linear SVM Classification

• $w^Tw$ is convex because:
  - The **identity matrix** sitting in between $w^T$ and $w$
  - The **eigenvalues of 2nd-order derivative** (Hessian) of quadratic function is $\geq 0$.
• Hence, it is a **positive semidefinite matrix**.
• Therefore, $w^Tw$ is convex function.

\[
\vec{w}^T\vec{w} = [\vec{w}^T \quad b^T] \begin{bmatrix}
\mathbb{I}_{d \times d} & 0 \\
0 & 0_{(d+1) \times (d+1)}
\end{bmatrix} \begin{bmatrix}
\vec{w} \\
b
\end{bmatrix}
\]

\[
\text{Hessian} = \frac{1}{2} \begin{bmatrix}
\mathbb{I}_{d \times d} & 0 \\
0 & 0
\end{bmatrix} \succeq 0
\]
Hard Margin Linear SVM Classification

• Note that we are trying to minimize $\frac{1}{2}w^Tw$, which is a **quadratic function**, subject to a **linear constraint**.

• It is essentially a **quadratic programming (QP) optimization**.

$$
\min_{w,b} \frac{1}{2} \|w\|^2 \equiv \frac{1}{2} w^T w
$$

subject to \( y_i (w^T x_i + b) \geq 1 \ \forall i \)

**Quadratic programming (QP)** is the process of **solving a (linearly constrained) quadratic optimization problem**.

If the objective function is convex, then any local minimum found is also the sole **global minimum**.
Hard Margin Linear SVM Classification

- Notice here we have \( d \) variables (\( w \)) and \( N \) constraints.
- We can use any standard QP solver to find an optimal solution for the primal problem.

\[
\begin{align*}
\min_{\bar{w}, b} & \quad \frac{1}{2} \left\| \bar{w} \right\|^2 \\
\text{subject to} & \quad y_i (\bar{w}^T \bar{x}_i + b) \geq 1 \quad \forall i
\end{align*}
\]

No. of variables (\( w \)): \( d \)

No. of constraints: \( N \)
Hard Margin Linear SVM Classification

• The main problem with the \textbf{primal} constrained minimization problem is that its \textit{complexity depends on} \(d\).
• For a \textbf{high-dimensional} dataset, the solution for this optimization problem would be \textit{expensive}.
• This expense will become \textit{prohibitively high} for a \textbf{nonlinear} dataset.

\[
\min_{\vec{w},b} \frac{1}{2} \|\vec{w}\|^2 \equiv \frac{1}{2} \vec{w}^T \vec{w} \\
\text{subject to} \quad y_i(\vec{w}^T \vec{x}_i + b) \geq 1 \forall i
\]

No. of variables (w): \(d\)
No. of constraints: \(N\)
Hard Margin Linear SVM Classification

- Consider a **nonlinear classification problem** that doesn’t admit a linear decision boundary.
- How do we make this data linearly separable?
- We **map the problem to a new space** that has many more dimensions than the original space.
Hard Margin Linear SVM Classification

• Thus, by **augmenting the feature space** (by using a basis function) we transform the original problem into a **linearly separable problem**.
Hard Margin Linear SVM Classification

• But this results into an explosion of the feature space that adds significant complexity to the model.
• In such a case, we are interested in a method whose complexity does not depend on the input dimensionality.

\[
\begin{align*}
\min_{\mathbf{w}, \mathbf{b}} \frac{1}{2} \|\mathbf{w}\|^2 & \equiv \frac{1}{2} \mathbf{w}^T \mathbf{w} \\
\text{subject to} \quad y_i (\mathbf{w}^T \mathbf{x}_i + \mathbf{b}) & \geq 1 \quad \forall i
\end{align*}
\]
Hard Margin Linear SVM Classification

• There is a technique that enables us to convert the **primal** optimization problem to a form whose complexity does not depend on the **dimension** $d$ of the data.

• We can actually convert the optimization problem into a **dual form** whose complexity depends on the **number of training instances** $N$.

In short, we can convert the problem from its **primal form** (complexity depends on $d$) to a **dual form** (complexity depends on $N$)

\[
\begin{align*}
\min_{\hat{w},b} & \quad \frac{1}{2} \|\hat{w}\|^2 \equiv \frac{1}{2} \hat{w}^T \hat{w} \\
\text{subject to} & \quad y_i (\hat{w}^T \hat{x}_i + b) \geq 1 \ \forall i
\end{align*}
\]
Hard Margin Linear SVM Classification

Primal form (complexity depends on $d$)

$$
\min_{\vec{w}, b} \frac{1}{2} \|\vec{w}\|^2 \equiv \frac{1}{2} \vec{w}^T \vec{w}
$$

subject to  \( y_i(\vec{w}^T \vec{x}_i + b) \geq 1 \ \forall i \)

Dual form (complexity depends on $N$)

$$
\max_{\alpha_1, \ldots, \alpha_N} \vec{\alpha}^T \vec{1} - \frac{1}{2} \vec{\alpha}^T H \vec{\alpha}
$$

subject to  \( \sum_{i=1}^{N} \alpha_i y_i = 0 \)

$$
\vec{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \vec{x}_i
\quad \alpha_i \geq 0
$$

where \( H_{ij} = y_i y_j \vec{x}_i^T \vec{x}_j \)
Hard Margin Linear SVM Classification

• The dual problem would benefit us in two ways.

**Benefit 1:** Its complexity depends on N (not on d)

\[
\max_{\alpha_1, \ldots, \alpha_N} \sum_{i=1}^{N} \alpha_i y_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \]

subject to \( \sum_{i=1}^{N} \alpha_i y_i = 0 \)

\( \alpha_i \geq 0 \)

\[ w^* = \sum_{i=1}^{N} \alpha_i y_i x_i \]

where \( H_{ij} = y_i y_j \langle x_i, x_j \rangle \)

**Benefit 2:** It will allow us to solve a high-dimensional optimization problem without even projecting the data in high-dimension (for nonlinear case)

This will be possible by a “magical” technique known as Kernel trick!
Thus, although we can directly solve the **primal problem** (by using CVXOPT), we are interested to **convert it into its dual form**.

### C V X O P T

CVXOPT is a free software package for convex optimization based on the Python programming language. It can be used with the interactive Python interpreter, on the command line by executing Python scripts, or integrated in other software via Python extension modules. Its main purpose is to make the development of software for convex optimization applications straightforward by building on Python's extensive standard library and on the strengths of Python as a high-level programming language.
Hard Margin Linear SVM Classification

\[
\begin{align*}
\min_{\vec{w}, b} & \quad \frac{1}{2} \|\vec{w}\|^2 = \frac{1}{2} \vec{w}^T \vec{w} \\
\text{subject to} & \quad y_i(\vec{w}^T \vec{x}_i + b) \geq 1 \quad \forall i
\end{align*}
\]

How do we transform the primal problem into its dual form?

We can derive the dual form by analytically solving the primal problem.

Bird’s-eye view of Scikit-Learn implementations of the SVM primal & dual algorithms:

\[
\begin{align*}
\max_{\alpha_1, \ldots, \alpha_N} & \quad \vec{a}^T \vec{1} - \frac{1}{2} \vec{a}^T H \vec{a} \\
\text{subject to} & \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& \quad \vec{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \vec{x}_i \\
& \quad \alpha_i \geq 0
\end{align*}
\]

where \( H_{ij} = y_i y_j \vec{x}_i^T \vec{x}_j \)