<table>
<thead>
<tr>
<th>Date</th>
<th>Day</th>
<th>no.</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 08</td>
<td>Mon</td>
<td>1</td>
<td>A Hands-On Introduction</td>
</tr>
<tr>
<td>Oct 10</td>
<td>Wed</td>
<td>–</td>
<td>self-study (Christoph traveling)</td>
</tr>
<tr>
<td>Oct 15</td>
<td>Mon</td>
<td>2</td>
<td>Bayesian Decision Theory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Generative Probabilistic Models</td>
</tr>
<tr>
<td>Oct 17</td>
<td>Wed</td>
<td>3</td>
<td>Discriminative Probabilistic Models</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Maximum Margin Classifiers</td>
</tr>
<tr>
<td>Oct 22</td>
<td>Mon</td>
<td>4</td>
<td>Generalized Linear Classifiers, Optimization</td>
</tr>
<tr>
<td>Oct 24</td>
<td>Wed</td>
<td>5</td>
<td>Evaluating Predictors; Model Selection</td>
</tr>
<tr>
<td>Oct 29</td>
<td>Mon</td>
<td>–</td>
<td>self-study (Christoph traveling)</td>
</tr>
<tr>
<td>Oct 31</td>
<td>Wed</td>
<td>6</td>
<td>Overfitting/Underfitting, Regularization</td>
</tr>
<tr>
<td>Nov 05</td>
<td>Mon</td>
<td>7</td>
<td>Learning Theory I: classical/Rademacher bounds</td>
</tr>
<tr>
<td>Nov 07</td>
<td>Wed</td>
<td>8</td>
<td>Learning Theory II: miscellaneous</td>
</tr>
<tr>
<td>Nov 12</td>
<td>Mon</td>
<td>9</td>
<td>Probabilistic Graphical Models I</td>
</tr>
<tr>
<td>Nov 14</td>
<td>Wed</td>
<td>10</td>
<td>Probabilistic Graphical Models II</td>
</tr>
<tr>
<td>Nov 19</td>
<td>Mon</td>
<td>11</td>
<td>Probabilistic Graphical Models III</td>
</tr>
<tr>
<td>Nov 21</td>
<td>Wed</td>
<td>12</td>
<td>Probabilistic Graphical Models IV</td>
</tr>
<tr>
<td>until</td>
<td></td>
<td></td>
<td>final project</td>
</tr>
<tr>
<td>Nov 25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What, if a linear classifier is really not a good choice?

Change the data representation, e.g. Cartesian $\rightarrow$ polar coordinates.
Nonlinear Classifiers

What, if a linear classifier is really not a good choice?

Change the data representation, e.g. Cartesian → polar coordinates
Definition (Max-margin Generalized Linear Classifier)

Let $C > 0$. Assume a necessarily linearly separable training set

$$\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}.$$ 

Let $\phi : \mathcal{X} \to \mathbb{R}^D$ be a feature map from $\mathcal{X}$ into a feature space $\mathbb{R}^D$. Then we can form a new training set

$$\mathcal{D}^\phi = \{(\phi(x^1), y^1), \ldots, (\phi(x^n), y^n)\} \subset \mathbb{R}^D \times \mathcal{Y}.$$ 

The maximum-(soft)-margin linear classifier in $\mathbb{R}^D$,

$$g(x) = \text{sign}[\langle w, \phi(x) \rangle_{\mathbb{R}^D} + b]$$

for $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$ is called \textbf{max-margin generalized linear classifier}.

It is still \textit{linear} w.r.t $w$, but (in general) nonlinear with respect to $x$. 
Example (Polar coordinates)

Left: dataset $\mathcal{D}$ for which no good linear classifier exists.
Right: dataset $\mathcal{D}^\phi$ for $\phi : \mathcal{X} \rightarrow \mathbb{R}^D$ with $\mathcal{X} = \mathbb{R}^2$ and $\mathbb{R}^D = \mathbb{R}^2$

$$
\phi(x, y) = (\sqrt{x^2 + y^2}, \arctan \frac{y}{x}) \quad \text{(and } \phi(0, 0) = (0, 0)\text{)}
$$
Example (Polar coordinates)

Left: dataset $\mathcal{D}$ for which no good linear classifier exists.
Right: dataset $\mathcal{D}^\phi$ for $\phi : \mathcal{X} \to \mathbb{R}^D$ with $\mathcal{X} = \mathbb{R}^2$ and $\mathbb{R}^D = \mathbb{R}^2$

\[
\phi(x, y) = (\sqrt{x^2 + y^2}, \arctan \frac{y}{x}) \quad \text{(and } \phi(0, 0) = (0, 0))
\]

Any classifier in $\mathbb{R}^D$ induces a classifier in $\mathcal{X}$.
Other popular feature mappings, $\phi$

**Example ($d$-th degree polynomials)**

$\phi : (x_1, \ldots, x_n) \mapsto (1, x_1, \ldots, x_n, x_1^2, \ldots, x_n^2, x_1x_2, \ldots, x_n^2, \ldots, x_d^n)$

Resulting classifier: $d$-th degree polynomial in $x$. $g(x) = \text{sign } f(x)$ with

$$f(x) = \langle w, \phi(x) \rangle = \sum_j w_j \phi(x)_j = \sum_i a_i x_i + \sum_{ij} b_{ij} x_i x_j + \ldots$$

**Example (Distance map)**

For a set of prototype $p_1, \ldots, p_N \in \mathcal{X}$:

$\phi : \vec{x} \mapsto \left( e^{-\|\vec{x} - \vec{p}_1\|^2}, \ldots, e^{-\|\vec{x} - \vec{p}_N\|^2} \right)$

Classifier: combine weights from close enough prototypes

$$g(x) = \text{sign} \langle w, \phi(x) \rangle = \text{sign} \sum_{i=1}^n a_i e^{-\|\vec{x} - \vec{p}_i\|^2}.$$
Example (Pre-trained deep network)

Imagine somebody trained a (deep) neural network on a large dataset, e.g. ImageNet for image classification.

Idea: use initial segment of network as feature extractor for other data:

min_{w \in \mathbb{R}^D, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi^i

subject to

y^i(\langle w, \phi(x^i) \rangle + b) \geq 1 - \xi^i, \quad \text{for } i = 1, \ldots, n,

\xi^i \geq 0. \quad \text{for } i = 1, \ldots, n.

How to solve numerically?

• off-the-shelf Quadratic Program (QP) solver
  only for small dimensions and training sets (a few hundred),

• variants of gradient descent,
  high dimensional data, large training sets (millions)

• by convex duality,
  for very high dimensional data and not so many examples \((d \gg n)\)
For simplification of notation, switch back to linear classifier:

\[
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi^i
\]

subject to

\[y^i(\langle w, x^i \rangle + b) \geq 1 - \xi^i, \quad \text{for } i = 1, \ldots, n,\]
\[\xi^i \geq 0. \quad \text{for } i = 1, \ldots, n.\]

How to solve numerically?

- off-the-shelf Quadratic Program (QP) solver only for small dimensions and training sets (a few hundred),
- variants of gradient descent,
  high dimensional data, large training sets (millions)
- by convex duality,
  for very high dimensional data and not so many examples \((d \gg n)\)
Subgradient-Based Optimization

\[ \min_{w \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi^i \]

subject to

\[ y^i(\langle w, x^i \rangle + b) \geq 1 - \xi^i, \quad \text{and} \quad \xi^i \geq 0, \quad \text{for } i = 1, \ldots, n. \]
Subgradient-Based Optimization

\[
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi^i
\]

subject to

\[y^i(\langle w, x^i \rangle + b) \geq 1 - \xi^i, \quad \text{and} \quad \xi^i \geq 0, \quad \text{for } i = 1, \ldots, n.\]

For any fixed \((w, b)\) we can find the optimal \(\xi^1, \ldots, \xi^n\):

\[\xi^i = \max\{ 0, 1 - y_i(\langle w, x_i \rangle + b) \}.\]
Subgradient-Based Optimization

\[
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi^i
\]

subject to

\[
y^i(\langle w, x^i \rangle + b) \geq 1 - \xi^i, \quad \text{and} \quad \xi^i \geq 0, \quad \text{for } i = 1, \ldots, n.
\]

For any fixed \((w, b)\) we can find the optimal \(\xi^1, \ldots, \xi^n:\)

\[
\xi^i = \max\{0, 1 - y_i(\langle w, x_i \rangle + b)\}.
\]

Plug into original problem:

\[
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i(\langle w, x_i \rangle + b)\}.
\]

"Hinge loss"
SVM Training in the Primal

\[
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i (\langle w, x_i \rangle + b)\}.
\]

- unconstrained optimization problem
- convex
  - \(\frac{1}{2}\|w\|^2\) is convex (differentiable with Hessian = \(\text{Id} \succeq 0\))
  - linear/affine functions are convex
  - pointwise \(\max\) over convex functions is convex.
  - sum of convex functions is convex.

- \textit{not differentiable!}
SVM Training in the Primal

\[
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i(\langle w, x_i \rangle + b)\}.
\]

- unconstrained optimization problem
- convex
  - \( \frac{1}{2} \|w\|^2 \) is convex (differentiable with Hessian = \( \text{Id} \succeq 0 \))
  - linear/affine functions are convex
  - pointwise \( \max \) over convex functions is convex.
  - sum of convex functions is convex.
- not differentiable!

We can’t use gradient descent, since some points have no gradients!
**Definition:** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a **convex** function. A vector $v \in \mathbb{R}^d$ is called a **subgradient** of $f$ at $w_0$, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$
**Definition:** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a **convex** function. A vector $v \in \mathbb{R}^d$ is called a **subgradient** of $f$ at $w_0$, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$
**Definition:** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a **convex** function. A vector $v \in \mathbb{R}^d$ is called a **subgradient** of $f$ at $w_0$, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$
**Definition:** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. A vector $v \in \mathbb{R}^d$ is called a subgradient of $f$ at $w_0$, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$
**Definition:** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. A vector $v \in \mathbb{R}^d$ is called a subgradient of $f$ at $w_0$, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$ 

A general convex $f$ can have more than one subgradient at a position.

- We write $\nabla f(w_0)$ for the set of subgradients of $f$ at $w_0$,
- $v \in \nabla f(w_0)$ indicates that $v$ is a subgradient of $f$ at $w_0$. 

For differentiable $f$, the gradient $v = \nabla f(w_0)$ is the only subgradient.

If $f_1, \ldots, f_K$ are differentiable at $w_0$ and

$$f(w) = \max\{f_1(w), \ldots, f_K(w)\},$$

then $v = \nabla f_k(w_0)$ is a subgradient of $f$ at $w_0$, where $k$ is any index for which $f_k(w_0) = f(w_0)$.

Subgradients are only well defined for convex functions!
Illustration: Optimization using Gradients

\[ f(w_1, w_2) = (w_1)^2 + 2(w_2)^2 \quad \text{strictly convex, differentiable} \]
Illustration: Optimization using Gradients

\[ f(w_1, w_2) = (w_1)^2 + 2(w_2)^2 \]  

strictly convex, differentiable
Illustration: Optimization using Gradients

\[ f(w_1, w_2) = (w_1)^2 + 2(w_2)^2 \]

strictly convex, differentiable

Gradient of a differentiable function is a descent direction:

- for any \( w_t \) there exists an \( \eta \) such that
  \[ f(w_t + \eta v) < f(w_t) \]
$f(w_1, w_2) = (w_1)^2 + 2(w_2)^2$ strictly convex, differentiable

Gradient of a differentiable function is a descent direction:

- for any $w_t$ there exists an $\eta$ such that $f(w_t + \eta v) < f(w_t)$
Illustration: Optimization using Gradients

$$f(w_1, w_2) = (w_1)^2 + 2(w_2)^2$$  strictly convex, differentiable

Gradient of a differentiable function is a descent direction:

• for any $w_t$ there exists an $\eta$ such that
  $$f(w_t + \eta v) < f(w_t)$$
Illustration: Optimization using Gradients

\[ f(w_1, w_2) = (w_1)^2 + 2(w_2)^2 \]  
strictly convex, differentiable
Illustration: Optimization using Gradients

\[ f(w_1, w_2) = (w_1)^2 + 2(w_2)^2 \] strictly convex, differentiable

Gradient of a differentiable function is a descent direction:
- for any \( w_t \) there exists an \( \eta \) such that \( f(w_t + \eta v) < f(w_t) \)
Illustration: Optimization using Subgradients?

\[ f(w_1, w_2) = |w_1| + 2|w_2| \quad \text{convex, not differentiable} \]
Optimization using Subgradients?

\[ f(w_1, w_2) = |w_1| + 2|w_2| \]

convex, not differentiable

Subgradient might not be a descent direction:

- for \( w_t \) we might have \( f(w_t + \eta v) \geq f(w_t) \) for all \( \eta \in \mathbb{R} \)
- but: there is an \( \eta \) that brings us closer to the optimum,
  \[ \|w_{t+1} - w^*\| < \|w_t - w^*\| \]

(Proof: exercise...)
Illustration: Optimization using Subgradients?

\[ f(w_1, w_2) = |w_1| + 2|w_2| \]

convex, not differentiable

Subgradient might not be a descent direction:

- for \( w_t \) we might have \( f(w_t + \eta v) \geq f(w_t) \) for all \( \eta \in \mathbb{R} \)
- but: there is an \( \eta \) that brings us closer to the optimum, \( \|w_{t+1} - w^*\| < \|w_t - w^*\| \) (Proof: exercise...)
$f(w_1, w_2) = |w_1| + 2|w_2|$  convex, not differentiable

Subgradient might not be a descent direction:

• for $w_t$ we might have $f(w_t + \eta v) \geq f(w_t)$ for all $\eta \in \mathbb{R}$

• but: there is an $\eta$ that brings us closer to the optimum, $\|w_{t+1} - w^*\| < \|w_t - w^*\|$ (Proof: exercise...)
Illustration: Optimization using Subgradients?

\[ f(w_1, w_2) = |w_1| + 2|w_2| \quad \text{convex, not differentiable} \]
Illustration: Optimization using Subgradients?

\[ f(w_1, w_2) = |w_1| + 2|w_2| \] convex, not differentiable

\[ f(w_t + \eta v) \geq f(w_t) \quad \text{for all } \eta \in \mathbb{R} \]

but: there is an \( \eta \) that brings us closer to the optimum,

\[ \| w_{t+1} - w^* \| < \| w_t - w^* \| \quad (\text{Proof: exercise...}) \]
Illustration: Optimization using Subgradients?

\[ f(w_1, w_2) = |w_1| + 2|w_2| \quad \text{convex, not differentiable} \]

Subgradient might not be a not a descent direction:

- For \( w_t \) we might have \( f(w_t + \eta v) \geq f(w_t) \) for all \( \eta \in \mathbb{R} \)
Illustration: Optimization using Subgradients?

\[ f(w_1, w_2) = |w_1| + 2|w_2| \quad \text{convex, not differentiable} \]

Subgradient might not be a \textbf{not a descent direction}:

- for \( w_t \) we might have \( f(w_t + \eta v) \geq f(w_t) \) for all \( \eta \in \mathbb{R} \)
- but: there is an \( \eta \) that brings us closer to the optimum,
  \[ \|w_{t+1} - w^*\| < \|w_t - w^*\| \quad \text{(Proof: exercise...)} \]
**Subgradient Method (not Descent!**)  

**input** step sizes $\eta_1, \eta_2, \ldots$

1: $w_1 \leftarrow 0$

2: for $t = 1, \ldots, T$ do

3:  $v \leftarrow$ a subgradient of $\mathcal{L}$ at $w_t$

4:  $w_{t+1} \leftarrow w_t - \eta_t v$

5: end for

**output** $w_t$ with smallest values $\mathcal{L}(w_t)$ for $t = 1, \ldots, T$
Subgradient Method (not Descent!)

**input** step sizes \( \eta_1, \eta_2, \ldots \)

1: \( w_1 \leftarrow 0 \)

2: **for** \( t = 1, \ldots, T \) **do**

3: \( v \leftarrow \) a subgradient of \( \mathcal{L} \) at \( w_t \)

4: \( w_{t+1} \leftarrow w_t - \eta_t v \)

5: **end for**

**output** \( w_t \) with smallest values \( \mathcal{L}(w_t) \) for \( t = 1, \ldots, T \)

Stepsize rules: how to choose \( \eta_1, \eta_2, \ldots, ? \)

- \( \eta_t = \eta \) constant: will get us (only) close to the optimum
- decrease slowly, but not too slowly: converges to optimum

\[
\sum_{t=1}^{\infty} \eta_t = \infty \quad \sum_{t=1}^{\infty} (\eta_t)^2 < \infty \quad \text{e.g. } \eta_t = \frac{\eta}{t + t_0}
\]

How to choose overall \( \eta \)? trial-and-error
- Try different values, see which one decreases the objective (fastest)
Many objective functions in ML contain a sum over all training examples:

\[
L_{\text{LogReg}}(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i(\langle w, x_i \rangle + b))),
\]

\[
L_{\text{SVM}}(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i(\langle w, x_i \rangle + b)\}.
\]

Computing the gradient or subgradient scales like \(O(nd)\),
- \(d\) is the dimensionality of the data
- \(n\) is the number of training examples.

Both \(d\) and \(n\) can be big (millions). What can we do?
- we’ll not get rid of \(O(d)\), since \(w \in \mathbb{R}^d\),
- but we can get rid of the scaling with \(O(n)\) for each update!
Let \( f(w) = \sum_i f_i(w) \), with convex, differentiable \( f_1, \ldots, f_n \).

**Stochastic Gradient Descent**

**input** step sizes \( \eta_1, \eta_2, \ldots \)

1: \( w_1 \leftarrow 0 \)

2: **for** \( t = 1, \ldots, T \) **do**

3: \( i \leftarrow \text{random index in } 1, 2, \ldots, n \)

4: \( v \leftarrow n \nabla f_i(w_t) \)

5: \( w_{t+1} \leftarrow w_t - \eta_t v \)

6: **end for**

**output** \( w_T \), or average \( \frac{1}{T-T_0} \sum_{t=T_0}^T w_t \)

- Each iteration takes only \( O(d) \),
- Gradient is "wrong" in each step, but **correct in expectation**.
- No line search, since evaluating \( f(w - \eta v) \) would be \( O(nd) \),
- Objective does not decrease in every step,
- Converges to optimum if \( \eta_t \) is square summable, but not summable.
Let \( f(w) = \sum_i f_i(w) \), with convex \( f_1, \ldots, f_n \).

**Stochastic Subgradient Method**

**input** step sizes \( \eta_1, \eta_2, \ldots \)

1: \( w_1 \leftarrow 0 \)
2: **for** \( t = 1, \ldots, T \) **do**
3: \( i \leftarrow \) random index in \( 1, 2, \ldots, n \)
4: \( v \leftarrow n\bar{v} \) for \( \bar{v} \in \nabla f_i(w_t) \)
5: \( w_{t+1} \leftarrow w_t - \eta_t v \)
6: **end for**

**output** \( w_T \), or average \( \frac{1}{T-T_0} \sum_{t=T_0}^{T} w_t \)

- Each iteration takes only \( O(d) \),
- Converges to optimum if \( \eta_t \) is square summable, but not summable.
- Even better: pick not completely at random but go in epochs: randomly shuffle dataset, go through all examples, reshuffle, etc.
Stochastic Primal SVMs Training

\[
\mathcal{L}_{SVM}(w, b) = \sum_{i=1}^{n} \left( \frac{1}{2n} \|w\|^2 + C \max\{ 0, 1 - y_i(\langle w, x_i \rangle + b) \} \right).
\]

**input** step sizes \( \eta_1, \eta_2, \ldots \) or step size rule, such as \( \eta_t = \frac{\eta}{t+t_0} \)

1: \((w_1, b_1) \leftarrow (0, 0)\)
2: **for** \( t = 1, \ldots, T \) **do**
3: \( \text{pick } (x, y) \text{ from } \mathcal{D} \text{ (randomly, or in epochs)} \)
4: \( \text{if } y\langle x, w \rangle + b \geq 1 \text{ then} \)
5: \( w_{t+1} \leftarrow (1 - \eta_t)w_t \)
6: \( \text{else} \)
7: \( w_{t+1} \leftarrow (1 - \eta_t)w_t + nC \eta_t y x \)
8: \( b_{t+1} \leftarrow \eta_t nC y \)
9: \( \text{end if} \)
10: **end for**

**output** \( w_T \), or average \( \frac{1}{T-T_0} \sum_{t=T_0}^{T} w_t \)

Widely used for SVM training, but setting stepsizes can be painful.
SVM Optimization by Dualization

Back to the original formulation

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi^i$$

subject to, for $i = 1, \ldots, n,$

$$y^i(\langle w, x^i \rangle + b) \geq 1 - \xi^i, \quad \text{and} \quad \xi^i \geq 0.$$ 

Convex optimization problem: we can study its dual problem.
General Principle of Dualization

Assume a constrained optimization problem:

$$\min_{\theta \in \Theta \subset \mathbb{R}^K} f(\theta)$$

subject to

$$g_1(\theta) \leq 0, \quad g_2(\theta) \leq 0, \quad \ldots, \quad g_k(\theta) \leq 0.$$
General Principle of Dualization

Assume a constrained optimization problem:

\[
\min_{\theta \in \Theta \subset \mathbb{R}^K} f(\theta)
\]

subject to

\[
g_1(\theta) \leq 0, \quad g_2(\theta) \leq 0, \quad \ldots, \quad g_k(\theta) \leq 0.
\]

We define the **Lagrangian**, that combines objective and constraints

\[
\mathcal{L}(\theta, \alpha) = f(\theta) + \alpha_1 g_1(\theta) + \cdots + \alpha_k g_k(\theta)
\]

with **Lagrange multipliers**, \(\alpha_1, \ldots, \alpha_k \geq 0\). Note:

\[
\max_{\alpha_1 \geq 0, \ldots, \alpha_k \geq 0} \mathcal{L}(\theta, \alpha) = \begin{cases}  
 f(\theta) & \text{if } g_1(\theta) \leq 0, \ g_2(\theta) \leq 0, \ \ldots, \ g_k(\theta) \leq 0 \\
 \infty & \text{otherwise} \end{cases}
\]

Any optimal solution, \(\theta\), for \(\min_{\theta \in \Theta} \max_{\alpha \geq 0} \mathcal{L}(\theta, \alpha)\) is also optimal for the original constrained problem.
Theorem (Special Case of Slater’s Condition)

If $f$ is convex, $g_1, \ldots, g_k$ are affine functions, and there exists at least one point $\theta \in \text{relint}(\Theta)$ that is feasible (i.e. $g_i(\theta) \leq 0$ for $i = 1, \ldots, k$). Then

$$\min_{\theta \in \Theta} \max_{\alpha \geq 0} \mathcal{L}(\theta, \alpha) = \max_{\alpha \geq 0} \min_{\theta \in \Theta} \mathcal{L}(\theta, \alpha)$$
Theorem (Special Case of Slater’s Condition)

If $f$ is convex, $g_1, \ldots, g_k$ are affine functions, and there exists at least one point $\theta \in \text{relint}(\Theta)$ that is feasible (i.e. $g_i(\theta) \leq 0$ for $i = 1, \ldots, k$). Then

$$\min_{\theta \in \Theta} \max_{\alpha \geq 0} \mathcal{L}(\theta, \alpha) = \max_{\alpha \geq 0} \min_{\theta \in \Theta} \mathcal{L}(\theta, \alpha)$$

Call $f(\theta)$ the **primal** and $h(\alpha) = \min_{\theta \in \Theta} \mathcal{L}(\theta, \alpha)$ be the **dual function**.

The theorem states that minimizing the primal $f(\theta)$ (with constraints given by the $g_k$) is equivalent to maximizing its dual $h(\alpha)$ (with $\alpha \geq 0$).

$$\min_{\theta \in \mathbb{R}^K} f(\theta) = \max_{\alpha \in \mathbb{R}_+^k} h(\alpha)$$
Dualizing of the SVM optimization problem

The SVM optimization problem fulfills the conditions of the theorem.

\[
\begin{align*}
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\
\text{subject to, for } & \quad i = 1, \ldots, n,
\end{align*}
\]

\[y^i(\langle w, x^i \rangle + b) \geq 1 - \xi^i, \quad \text{and} \quad \xi^i \geq 0.\]

We can compute its minimal value as \(\max_{\alpha \geq 0, \beta \geq 0} h(\alpha, \beta)\) with

\[h(\alpha, \beta) = \min_{(w,b)} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - \xi_i - y^i(\langle w, x^i \rangle + b)) - \sum_i \beta_i \xi_i\]

(Blackboard...)
Dualizing of the SVM optimization problem

In a minimum w.r.t. \((w, b)\):

\[
0 = \frac{\partial}{\partial w} \mathcal{L}(w, b, \xi, \alpha, \beta) = w - \sum_i \alpha_i y_i x^i \quad \Rightarrow \quad w = \sum_i \alpha_i y_i x^i
\]

\[
0 = \frac{\partial}{\partial b} \mathcal{L}(w, b, \xi, \alpha, \beta) = \sum_i \alpha_i y_i
\]

\[
0 = \frac{\partial}{\partial \xi_i} \mathcal{L}(w, b, \xi, \alpha, \beta) = C - \alpha_i - \beta_i
\]

Insert new constraints into objective:

\[
\max_{\alpha \geq 0} \quad \frac{1}{2} \left\| \sum_i \alpha_i y_i x^i \right\|^2 + \sum_i \alpha_i - \sum_i \alpha_i y_i \left\langle \sum_j \alpha_j y_j x^j, x^i \right\rangle
\]
SVM Dual Optimization Problem

\[
\max_{\alpha \geq 0} \quad -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x^i, x^j \rangle + \sum_i \alpha_i
\]

subject to \( \sum_i \alpha_i y_i = 0 \) and \( 0 \leq \alpha_i \leq C \), for \( i = 1, \ldots, n \).

- Examples \( x^i \) with \( \alpha_i \neq 0 \) are called support vectors.
- From the coefficients \( \alpha_1, \ldots, \alpha_n \) we can recover the optimal \( w \):
  \[
  w = \sum_i \alpha_i y^i x^i
  \]
  \[
  b = 1 - y^i \langle x^i, w \rangle \quad \text{for any } i \text{ with } 0 < \alpha_i < C
  \]
  (more complex rule for \( b \) if no such \( i \) exists).
- The prediction rule becomes
  \[
  g(x) = \text{sign} \left( \langle w, x \rangle + b \right) = \text{sign} \left( \sum_{i=1}^n \alpha_i y_i \langle x_i, x \rangle + b \right)
  \]
SVM Dual Optimization Problem

\[
\begin{align*}
\max_{\alpha \geq 0} & \quad -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x^i, x^j \rangle + \sum_i \alpha_i \\
\text{subject to} & \quad \sum_i \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, n.
\end{align*}
\]

Why solve the dual optimization problem?

- fewer unknowns: $\alpha \in \mathbb{R}^n$ instead of $(w, b, \xi) \in \mathbb{R}^{d+1+n}$
- less storage when $d \gg n$:
  \[
  (\langle x^i, x^j \rangle)_{i,j} \in \mathbb{R}^{n \times n} \text{ instead of } (x^1, \ldots, x^n) \in \mathbb{R}^{n \times d}
  \]
- Kernelization (not in this course)
For optimization, the *bias term* is an annoyance

- In primal optimization, it often requires a different stepsize.
- In dual optimization, sometimes not straight-forward to recover.
- It couples the dual variables by an equality constraint: $\sum_i \alpha_i y_i = 0$.

We can get rid of the bias by the **augmentation trick**.

**Original:**

- $f(x) = \langle w, x \rangle_{\mathbb{R}^d} + b$, with $w \in \mathbb{R}^d, b \in \mathbb{R}$.

**New augmented:**

- **linear:** $f(x) = \langle \tilde{w}, \tilde{x} \rangle_{\mathbb{R}^{d+1}}$, with $\tilde{w} = (w, b)$, $\tilde{x} = (x, 1)$.
- **generalized:** $f(x) = \langle \tilde{w}, \tilde{\phi}(x) \rangle_{\tilde{\mathcal{H}}}$ with $\tilde{w} = (w, b)$, $\tilde{\phi}(x) = (\phi(x), 1)$. 
SVM without bias term – primal optimization problem

\[
\begin{align*}
\min_{w \in \mathbb{R}^d, \xi \in \mathbb{R}^n} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi^i \\
\text{subject to, for } i = 1, \ldots, n, & \quad y^i \langle w, x^i \rangle \geq 1 - \xi^i, \quad \text{and} \quad \xi^i \geq 0.
\end{align*}
\]

Difference: no \( b \) variable to optimize over
SVMs Without Bias Term – Optimization

**SVM without bias term – primal optimization problem**

\[
\min_{w \in \mathbb{R}^d, \xi \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi^i
\]

subject to, for \(i = 1, \ldots, n\),

\[y^i \langle w, x^i \rangle \geq 1 - \xi^i, \quad \text{and} \quad \xi^i \geq 0.\]

Difference: no \(b\) variable to optimize over

**SVM without bias term – dual optimization problem**

\[
\max_{\alpha} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^i y^j \langle x^i, x^j \rangle + \sum_i \alpha_i
\]

subject to, \(0 \leq \alpha_i \leq C\), for \(i = 1, \ldots, n\).

Difference to variant with bias term: no constraint \(\sum_i y_i \alpha_i = 0\).
Linear SVM Optimization in the Dual

Stochastic Coordinate Dual Ascent

\[ \begin{align*}
\alpha & \leftarrow 0. \\
\text{for } t = 1, \ldots, T & \text{ do} \\
& i \leftarrow \text{random index (uniformly random or in epochs)} \\
& \text{solve QP w.r.t. } \alpha_i \text{ with all } \alpha_j \text{ for } j \neq i \text{ fixed.} \\
\text{end for} \\
\text{return } \alpha
\end{align*} \]

Properties:
- converges monotonically to global optimum
- each subproblem has smallest possible size: 1-dimensional

Open Problem:
- how to make each step efficient?
What’s the complexity of the update step? Derive an explicit expression:

Original problem: \( \max_{\alpha \in [0, C]^n} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x^i, x^j \rangle + \sum_i \alpha_i \)
What’s the complexity of the update step? Derive an explicit expression:

Original problem: \( \max_{\alpha \in [0,C]^n} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^i y^j \langle x^i, x^j \rangle + \sum_i \alpha_i \)

When all \( \alpha_j \) except \( \alpha_i \) are fixed: \( \max_{\alpha_i \in [0,C]} F(\alpha_i) \), with

\[
F(\alpha_i) = -\frac{1}{2} \alpha_i^2 \langle x^i, x^i \rangle + \alpha_i \left( 1 - y^i \sum_{j \neq i} \alpha_j y^j \langle x^i, x^j \rangle \right) + \text{const.}
\]

\[
\frac{\partial}{\partial \alpha_i} F(\alpha_i) = -\alpha_i \|x^i\|^2 + \left( 1 - y^i \sum_{j \neq i} \alpha_j y^j \langle x^i, x^j \rangle \right) + \text{const.}
\]

\[
\alpha_{i,\text{new}} = \alpha_i + \frac{1 - y^i \sum_{j=1}^n \alpha_j y^j \langle x^i, x^j \rangle}{\|x^i\|^2}, \quad \alpha_i = \begin{cases} 0 & \text{if } \alpha_{i,\text{new}} < 0, \\ C & \text{if } \alpha_{i,\text{new}} > C, \\ \alpha_{i,\text{new}} & \text{otherwise.} \end{cases}
\]

\( (\alpha_i \text{ show up, because sum range is } j = 1, \ldots, n, \text{ not } j \neq i) \)

- complexity of each update: \( n \) inner products = \( O(nd) \)
- if we pre-compute and store all \( \langle x_i, x_j \rangle \): \( O(n) \) with \( O(n^2) \) storage
For $n \gg d$, we can improve using the linearity of $\langle \cdot , \cdot \rangle$:

$$
\alpha_i^{\text{new}} = \alpha_i + \frac{1 - y^i \sum_j \alpha_j y^j \langle x^i, x^j \rangle}{\|x^i\|^2}
= \alpha_i + \frac{1 - y^i \langle x^i, \sum_j \alpha_j y^j x^j \rangle}{\|x^i\|^2}
$$

remember $w = \sum_j \alpha_j y^j x^j$. If we keep $w$ stored explicitly:

$$
= \alpha_i + \frac{1 - y^i \langle w, x^i \rangle}{\|x^i\|^2},
$$

- each update: $O(d)$, independent of $n$
  - $\langle w, x^i \rangle$ takes $O(d)$ for explicit $w \in \mathbb{R}^d$
  - taking care that $w$ stays up-to-date: also $O(d)$

$$
w^{\text{new}} = w^{\text{old}} + (\alpha_i^{\text{new}} - \alpha_i^{\text{old}}) y^i x^i$$
SCDA for (Generalized) Linear SVMs [Hsieh, 2008]

initialize $\alpha \leftarrow 0$, $w \leftarrow 0$

for $t = 1, \ldots, T$ do
  $i \leftarrow$ random index (uniformly random or in epochs)
  $\delta \leftarrow \frac{1-y^i \langle w, x^i \rangle}{\|x^i\|^2}$
  $\bar{\alpha} \leftarrow \begin{cases} 
  0, & \text{if } \alpha_i + \delta < 0, \\
  C, & \text{if } \alpha_i + \delta > C, \\
  \alpha_i + \delta, & \text{otherwise.}
  \end{cases}$
  $w \leftarrow w + (\bar{\alpha} - \alpha_i) y^i x^i$
  $\alpha_i \leftarrow \bar{\alpha}$
end for

return $\alpha$, $w$

Properties:

- converges monotonically to global optimum
- complexity of each step is independent of $n$
- resembles stochastic gradient method, but step size is automatic