# Support Vector Machines: Training with <br> Stochastic Gradient Descent 

Machine Learning

## Support vector machines

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


## SVM objective function



> A hyper-parameter that controls the tradeoff between a large margin and a small hinge-loss

## Outline: Training SVM by optimization

1. Review of convex functions and gradient descent
2. Stochastic gradient descent
3. Gradient descent vs stochastic gradient descent
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron

## Outline: Training SVM by optimization

1. Review of convex functions and gradient descent
2. Stochastic gradient descent
3. Gradient descent vs stochastic gradient descent
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron

## Solving the SVM optimization problem

$$
\min _{\mathbf{w}} \frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum_{i} \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

This function is convex in $\mathbf{w}$

## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have


## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have


## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have


## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have


## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have


## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have


## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have


## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have


## Recall: Convex functions

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in[0,1]$ we have

$$
f(\lambda u+(1-\lambda) v) \leq \lambda f(u)+(1-\lambda) f(v)
$$

From geometric perspective

Every tangent plane lies below the function


## Convex functions

$$
f(x)=-\mathscr{X}
$$

Linear functions


$$
f\left(x_{1}, x_{2}\right)=\frac{x_{1}^{2}}{a^{2}}+\frac{x_{2}^{2}}{b^{2}}
$$

$$
f(x)=x^{2}
$$


$f(x)=\max (0, x)$
max is convex


Some ways to show that a function is convex:

1. Using the definition of convexity
2. Showing that the second derivative is positive (for one dimensional functions)
3. Showing that the second derivative is positive semi-definite (for vector functions)

## Not all functions are convex



## Convex functions are convenient

A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in$ [0,1] we have

In general: Necessary condition for $x$ to be a minimum for the function $f$ is that the gradient $\nabla f(x)=0$

For convex functions, this is both necessary and sufficient

## Solving the SVM optimization problem

$$
\min _{\mathbf{w}} \frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum_{i} \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

This function is convex in w

- This is a quadratic optimization problem because the objective is quadratic
- Older methods: Used techniques from Quadratic Programming
- Very slow
- No constraints, can use gradient descent
- Still very slow!

We are trying to minimize

## Gradient descent

General strategy for minimizing a function $J(\mathbf{w})$

- Start with an initial guess for $\mathbf{w}$, say $\mathbf{w}^{0}$
- Iterate till convergence:
- Compute the gradient of the gradient of $J$ at $\mathbf{w}^{t}$
- Update $\mathbf{w}^{t}$ to get $\mathbf{w}^{t+1}$ by taking a step in the opposite direction of the gradient


Intuition: The gradient is the direction of steepest increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

## Gradient descent

General strategy for minimizing a function $J(\mathbf{w})$

- Start with an initial guess for $\mathbf{w}$, say $\mathbf{w}^{0}$
- Iterate till convergence:
- Compute the gradient of the gradient of $J$ at $\mathbf{w}^{t}$
- Update $\mathbf{w}^{t}$ to get $\mathbf{w}^{t+1}$ by taking a step in the opposite direction of the gradient


Intuition: The gradient is the direction of steepest increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

## Gradient descent

General strategy for minimizing a function $J(\mathbf{w})$

- Start with an initial guess for $\mathbf{w}$, say $\mathbf{w}^{0}$
- Iterate till convergence:
- Compute the gradient of the gradient of $J$ at $\mathbf{w}^{t}$
- Update $\mathbf{w}^{t}$ to get $\mathbf{w}^{t+1}$ by taking a step in the opposite direction of the gradient


Intuition: The gradient is the direction of steepest increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

## Gradient descent

General strategy for minimizing a function $J(\mathbf{w})$

- Start with an initial guess for $\mathbf{w}$, say $\mathbf{w}^{0}$
- Iterate till convergence:
- Compute the gradient of the gradient of $J$ at $\mathbf{w}^{t}$
- Update $\mathbf{w}^{t}$ to get $\mathbf{w}^{t+1}$ by taking a step in the opposite direction of the gradient


Intuition: The gradient is the direction of steepest increase in the function. To get to the minimum, go in the opposite direction

## Gradient descent for SVM

1. Initialize $\mathbf{w}^{0}$

$$
J(\mathbf{w})=\min _{\mathbf{w}} \frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum_{i}^{\text {We are trying to minimize }} \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

2. For $t=0,1,2, \ldots$.
3. Compute gradient of $J(\mathbf{w})$ at $\mathbf{w}^{t}$. Call it $\nabla J\left(\mathbf{w}^{t+1}\right)$
4. Update $w$ as follows:

$$
\mathbf{w}^{t+1} \leftarrow \mathbf{w}^{t}-r \nabla J\left(\mathbf{w}^{t}\right)
$$

$r$ : The learning rate .

## Outline: Training SVM by optimization

$\checkmark$ Review of convex functions and gradient descent
2. Stochastic gradient descent
3. Gradient descent vs stochastic gradient descent
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron

## Gradient descent for SVM

1. Initialize $\mathbf{w}^{0}$

$$
J(\mathbf{w})=\min _{\mathbf{w}} \frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum_{i} \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

2. For $t=0,1,2, \ldots$.
3. Compute gradient of $J(\mathbf{w})$ at $\mathbf{w}^{t}$. Call it $\nabla J\left(\mathbf{w}^{t+1}\right)$

Gradient of the SVM objective requires summing over the entire training set

Slow, does not really scale

$$
J(\mathbf{w})=\frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum_{i} \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

## Stochastic gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}^{0}=0 \in \mathfrak{R}^{d}$
2. For epoch $=1$... T:
3. Return final $\mathbf{w}$

$$
J(\mathbf{w})=\frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

## Stochastic gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}^{0}=0 \in \mathfrak{R}^{d}$
2. For epoch $=1$... T:
3. Pick a random example $\left(\mathbf{x}_{i}, y_{i}\right)$ from the training set $S$
4. Return final $\mathbf{w}$

$$
J(\mathbf{w})=\frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

## Stochastic gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}^{0}=0 \in \mathfrak{R}^{d}$
2. For epoch $=1$... T:
3. Pick a random example $\left(\mathbf{x}_{i}, y_{i}\right)$ from the training set $S$
4. Treat $\left(\mathbf{x}_{i}, y_{i}\right)$ as a full dataset and take the derivative of the SVM objective $\hat{J}$ at the current $\mathbf{w}^{t-1}$. Call it $\nabla \hat{J}\left(\mathbf{w}^{\mathrm{t}-1}\right)$
5. Return final $\mathbf{w}$

$$
J(\mathbf{w})=\frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

## Stochastic gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}^{0}=0 \in \mathfrak{R}^{d}$
2. For epoch $=1$... T:
3. Pick a random example ( $\mathbf{x}_{i}, y_{i}$ ) from the training set $S$
4. Treat $\left(\mathbf{x}_{i}, y_{i}\right)$ as a full dataset and take the derivative of the SVM objective $\hat{J}$ at the current $\mathbf{w}^{t-1}$. Call it $\nabla \hat{J}\left(\mathbf{w}^{\mathrm{t}-1}\right)$

$$
\hat{J}(\mathbf{w})=\min _{\mathbf{w}} \frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

3. Return final $\mathbf{w}$

$$
J(\mathbf{w})=\frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

## Stochastic gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}^{0}=0 \in \mathfrak{R}^{d}$
2. For epoch $=1$... T:
3. Pick a random example ( $\mathbf{x}_{i}, y_{i}$ ) from the training set $S$
4. Treat $\left(\mathbf{x}_{i}, y_{i}\right)$ as a full dataset and take the derivative of the SVM objective $\hat{J}$ at the current $\mathbf{w}^{t-1}$. Call it $\nabla \hat{J}\left(\mathbf{w}^{\mathrm{t}-1}\right)$

$$
\hat{J}(\mathbf{w})=\min _{\mathbf{w}} \frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

3. Update: $\mathbf{w}^{t} \leftarrow \mathbf{w}^{t-1}-\gamma_{t} \nabla \mathbf{J}\left(\mathbf{w}^{\mathrm{t}-1}\right)$
4. Return final $\mathbf{w}$

$$
J(\mathbf{w})=\frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum_{i} \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

## Stochastic gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}^{0}=0 \in \mathfrak{R}^{d}$
2. For epoch $=1$... T:
3. Pick a random example $\left(\mathbf{x}_{i}, y_{i}\right)$ from the training set $S$
4. Treat $\left(\mathbf{x}_{i}, y_{i}\right)$ as a full dataset and take the derivative of the SVM objective $\hat{J}$ at the current $\mathbf{w}^{t-1}$. Call it $\nabla \hat{J}\left(\mathbf{w}^{\mathrm{t}-1}\right)$
5. Update: $\mathbf{w}^{t} \leftarrow \mathbf{w}^{t-1}-\gamma_{t} \nabla \mathrm{~J}\left(\mathbf{w}^{\mathrm{t}-1}\right)$

## 3. Return final w

This algorithm is guaranteed to converge to the minimum of $J$ if $\gamma_{t}$ is small enough. Why? The objective $J(\mathbf{w})$ is a convex function

## Outline: Training SVM by optimization

$\checkmark$ Review of convex functions and gradient descent
$\checkmark$ Stochastic gradient descent
3. Gradient descent vs stochastic gradient descent
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron

## Gradient Descent vs SGD



Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Gradient Descent vs SGD



Stochastic Gradient descent

## Outline: Training SVM by optimization

$\checkmark$ Review of convex functions and gradient descent
$\checkmark$ Stochastic gradient descent
$\checkmark$ Gradient descent vs stochastic gradient descent
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron

$$
J(\mathbf{w})=\min _{\mathbf{w}} \frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \sum_{i} \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}\right)
$$

## Stochastic gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \quad \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}^{0}=0 \in \mathfrak{R}^{d}$
2. For epoch $=1 \ldots \mathrm{~T}$ :
3. Pick a random example ( $\mathbf{x}_{i}, y_{i}$ ) from the training set $S$
4. Treat $\left(\mathbf{x}_{i}, y_{i}\right)$ as a full dataset and take the derivative of the SVM objective at the current $\mathbf{w}^{t-1}$ to be $\nabla \mathrm{J}\left(\mathbf{w}^{\mathrm{t}-1}\right)$
5. Update: $\mathbf{w}^{t} \leftarrow \mathbf{w}^{t-1}-\gamma_{t} \nabla \mathrm{~J}\left(\mathbf{w}^{\mathrm{t}-1}\right)$
6. Return final $\mathbf{w}$

## Hinge loss is not differentiable!

What is the derivative of the hinge loss with respect to $w$ ?

$$
J(\mathbf{w})=\min _{\mathbf{w}} \frac{1}{2} \mathbf{w}^{T} \mathbf{w}+c \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

## Detour: Sub-gradients

Generalization of gradients to non-differentiable functions
(Remember that every tangent is a hyperplane that lies below the function for convex functions)

Informally, a sub-tangent at a point is any hyperplane that lies below the function at the point.
A sub-gradient is the slope of that line

## Sub-gradients

Formally, a vector $g$ is a subgradient to $f$ at point $x$ if

$$
f(y) \geq f(x)+g^{T}(y-x) \quad \text { for all } y
$$



## Sub-gradients

Formally, a vector $g$ is a subgradient to $f$ at point $x$ if

$$
f(y) \geq f(x)+g^{T}(y-x) \quad \text { for all } y
$$

$f$ is differentiable at $x_{1}$
Tangent at this point
$f\left(x_{1}\right)+g_{1}^{T}\left(x-x_{1}\right)$
$\mathrm{g}_{1}$ is a gradient at $\mathrm{x}_{1}$


## Sub-gradients

Formally, a vector $g$ is a subgradient to $f$ at point x if

$$
f(y) \geq f(x)+g^{T}(y-x) \quad \text { for all } y
$$

$f$ is differentiable at $x_{1}$
Tangent at this point
$f\left(x_{1}\right)+g_{1}^{T}\left(x-x_{1}\right)$


## Sub-gradient of the SVM objective

$$
J^{t}(\mathbf{w})=\frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

General strategy: First solve the max and compute the gradient for each case

## Sub-gradient of the SVM objective

$$
J^{t}(\mathbf{w})=\frac{1}{2} \mathbf{w}^{T} \mathbf{w}+C \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)
$$

General strategy: First solve the max and compute the gradient for each case

$$
\nabla J^{t}= \begin{cases}\mathbf{w} & \text { if } \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)=0 \\ \mathbf{w}-C y_{i} \mathbf{x}_{i} & \text { otherwise }\end{cases}
$$

## Outline: Training SVM by optimization

$\checkmark$ Review of convex functions and gradient descent
$\checkmark$ Stochastic gradient descent
$\checkmark$ Gradient descent vs stochastic gradient descent
$\checkmark$ Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron

## Stochastic sub-gradient descent for SVM

$$
\nabla J^{t}= \begin{cases}\mathbf{w} & \text { if } \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)=0 \\ \mathbf{w}-C_{y_{i}} \mathbf{x}_{i} & \text { otherwise }\end{cases}
$$

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \quad \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}=0 \in \mathfrak{R}^{d}$
2. Return $\mathbf{w}$

## Stochastic sub-gradient descent for SVM

$$
\nabla \cdot J^{t}= \begin{cases}\mathbf{w} & \text { if } \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)=0 \\ \mathbf{w}-C y_{i} \mathbf{x}_{i} & \text { otherwise }\end{cases}
$$

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \quad \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}=0 \in \mathfrak{R}^{d}$
2. For epoch = 1 ... T:
3. Return $\mathbf{w}$

## Stochastic sub-gradient descent for SVM

$$
\nabla J^{t}= \begin{cases}\mathbf{w} & \text { if } \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)=0 \\ \mathbf{w}-C_{y_{i}} \mathbf{x}_{i} & \text { otherwise }\end{cases}
$$

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \quad \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}=0 \in \mathfrak{R}^{d}$
2. For epoch = 1 ... T:

For each training example $\left(\mathbf{x}_{i}, y_{i}\right) \in S$ :

$$
\text { Update } \mathbf{w} \leftarrow \mathbf{w}-\gamma_{t} \nabla J
$$

3. Return $\mathbf{w}$

## Stochastic sub-gradient descent for SVM

$$
\nabla J^{t}= \begin{cases}\mathbf{w} & \text { if } \max \left(0,1-y_{i} \mathbf{w}^{T} \mathbf{x}_{i}\right)=0 \\ \mathbf{w}-C_{y_{i} \mathbf{x}_{i}} & \text { otherwise }\end{cases}
$$

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \quad \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}=0 \in \mathfrak{R}^{d}$
2. For epoch = 1 ... T:

For each training example $\left(\mathbf{x}_{i}, y_{i}\right) \in S$ :

$$
\begin{aligned}
& \text { If } y_{i} \mathbf{w}^{T} \mathbf{x}_{i} \leq 1: \\
& \quad \mathbf{w} \leftarrow\left(1-\gamma_{t}\right) \mathbf{w}+\gamma_{t} C y_{i} \mathbf{x}_{i}
\end{aligned}
$$

else:

$$
\mathbf{w} \leftarrow\left(1-\gamma_{t}\right) \mathbf{w}
$$

3. Return $\mathbf{w}$

## Stochastic sub-gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \quad \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}=0 \in \Re^{d}$
2. For epoch = 1 ... T:

For each training example $\left(\mathbf{x}_{i}, y_{i}\right) \in S$ :
$\gamma_{t}$ : learning rate, many tweaks possible

$$
\text { If } y_{i} \mathbf{w}^{T} \mathbf{x}_{i} \leq 1 \text { : }
$$

$$
\mathbf{w} \leftarrow\left(1-\gamma_{t}\right) \mathbf{w}+\gamma_{t} C y_{i} \mathbf{x}_{i}
$$

else:

$$
\mathbf{w} \leftarrow\left(1-\gamma_{t}\right) \mathbf{w}
$$

3. Return $\mathbf{w}$

## Stochastic sub-gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \quad \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}=0 \in \mathfrak{R}^{d}$
2. For epoch = 1 ... T:

For each training example $\left(\mathbf{x}_{i}, y_{i}\right) \in S$ :

```
\(\gamma_{t}\) : learning rate, many tweaks possible
```

$$
\text { If } \begin{aligned}
y_{i} \mathbf{w}^{T} \mathbf{x}_{i} & \leq 1: \\
\quad \mathbf{w} & \leftarrow\left(1-\gamma_{t}\right) \mathbf{w}+\gamma_{t} C y_{i} \mathbf{x}_{i}
\end{aligned}
$$

else:

$$
\mathbf{w} \leftarrow\left(1-\gamma_{t}\right) \mathbf{w}
$$

3. Return $\mathbf{w}$

## Convergence and learning rates

With enough iterations, it will converge in expectation

Provided the step sizes are "square summable, but not summable"

- Step sizes $\gamma_{t}$ are positive
- Sum of squares of step sizes over $t=1$ to 1 is not infinite
- Sum of step sizes over $t=1$ to 1 is infinity
- Some examples: $\gamma_{t}=\frac{\gamma_{0}}{1+\frac{\gamma_{0} t}{C}}$ or $\gamma_{t}=\frac{\gamma_{0}}{1+t}$


## Convergence and learning rates

- Number of iterations to get to accuracy within $\epsilon$
- For strongly convex functions, N examples, d dimensional:
- Gradient descent: $O\left(N d \ln \frac{1}{\epsilon}\right)$
- Stochastic gradient descent: $O\left(\frac{d}{\epsilon}\right)$
- More subtleties involved, but SGD is generally preferable when the data size is huge


## Convergence and learning rates

- Number of iterations to get to accuracy within $\epsilon$
- For strongly convex functions, N examples, d dimensional:
- Gradient descent: $O\left(N d \ln \frac{1}{\epsilon}\right)$
- Stochastic gradient descent: $O\left(\frac{d}{\epsilon}\right)$
- More subtleties involved, but SGD is generally preferable when the data size is huge
- Recently, many variants that are based on this general strategy
- Examples: Adagrad, momentum, Nesterov's accelerated gradient, Adam, RMSProp, etc...


## Outline: Training SVM by optimization

$\checkmark$ Review of convex functions and gradient descent
$\checkmark$ Stochastic gradient descent
$\checkmark$ Gradient descent vs stochastic gradient descent
$\checkmark$ Sub-derivatives of the hinge loss
$\checkmark$ Stochastic sub-gradient descent for SVM
6. Comparison to perceptron

## Stochastic sub-gradient descent for SVM

Given a training set $S=\left\{\left(\mathbf{x}_{i}, y_{i}\right)\right\}, \quad \mathbf{x} \in \mathfrak{R}^{d}, y \in\{-1,1\}$

1. Initialize $\mathbf{w}=0 \in \Re^{d}$
2. For epoch = 1 ... T:

For each training example $\left(\mathbf{x}_{i}, y_{i}\right) \in S$ :

$$
\text { If } \begin{aligned}
& y_{i} \mathbf{w}^{T} \mathbf{x}_{i} \leq 1: \\
& \mathbf{w} \leftarrow\left(1-\gamma_{t}\right) \mathbf{w}+\gamma_{t} C y_{i} \mathbf{x}_{i}
\end{aligned}
$$

else:

$$
\mathbf{w} \leftarrow\left(1-\gamma_{t}\right) \mathbf{w}
$$

3. Return $\mathbf{w}$

Compare with the Perceptron update:
If $y_{i} \mathbf{W}^{T} \mathbf{x}_{i} \leq 0$, update $\mathbf{w} \leftarrow \mathbf{w}+\gamma_{t} y_{i} \mathbf{x}_{i}$

## Perceptron vs. SVM

- Perceptron: Stochastic sub-gradient descent for a different loss
- No regularization though

$$
L_{\text {Perceptron }}(y, \mathbf{x}, \mathbf{w})=\max \left(0,-y \mathbf{w}^{T} \mathbf{x}\right)
$$

- SVM optimizes the hinge loss
- With regularization

$$
L_{\text {Hinge }}(y, \mathbf{x}, \mathbf{w})=\max \left(0,1-y \mathbf{w}^{T} \mathbf{x}\right)
$$

## SVM summary from optimization perspective

- Minimize regularized hinge loss
- Solve using stochastic gradient descent
- Very fast, run time does not depend on number of examples
- Compare with Perceptron algorithm: Perceptron does not maximize margin width
- Perceptron variants can force a margin
- Convergence criterion is an issue; can be too aggressive in the beginning and get to a reasonably good solution fast; but convergence is slow for very accurate weight vector
- Other successful optimization algorithms exist
- Eg: Dual coordinate descent, implemented in liblinear

