# Support vector machines

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

# SVM objective function

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)$$

Regularization term:

- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

Empirical Loss:

- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

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

# Support vector machines

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

# 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

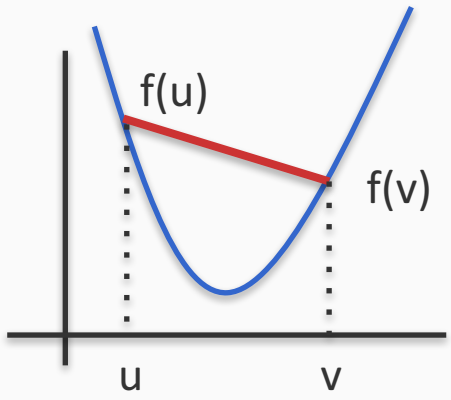
$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)$$

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

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



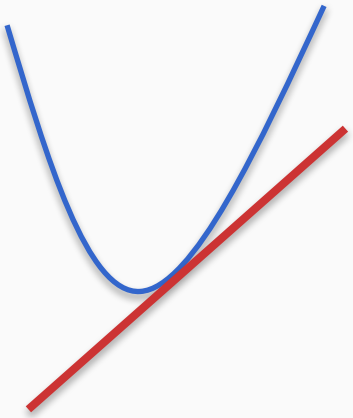
# Recall: Convex functions

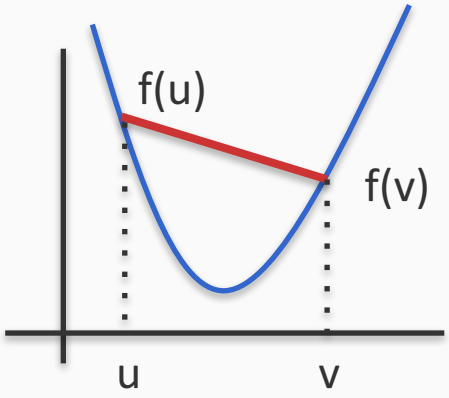
A function $f$ is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in [0,1]$ we have

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

From geometric perspective

Every tangent plane lies below the function

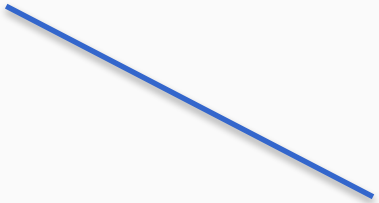




# Convex functions

$f(x) = -x$

Linear functions

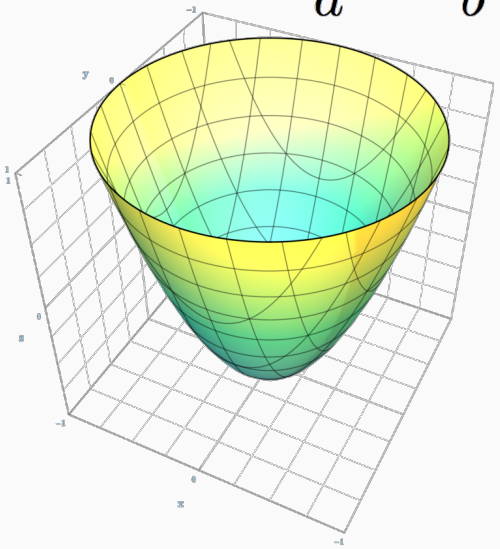


$f(x) = x^2$

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

*max* is convex

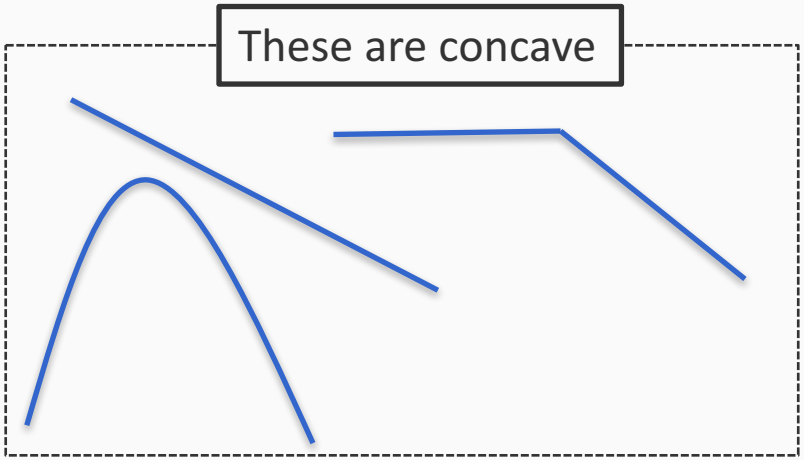
$$f(x_1, x_2) = \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2}$$



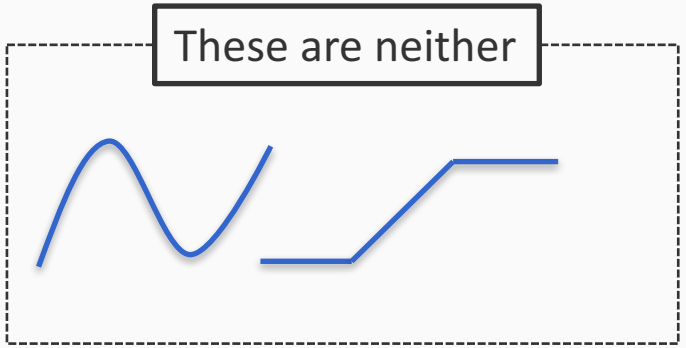
Some ways to show that a function is convex:

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



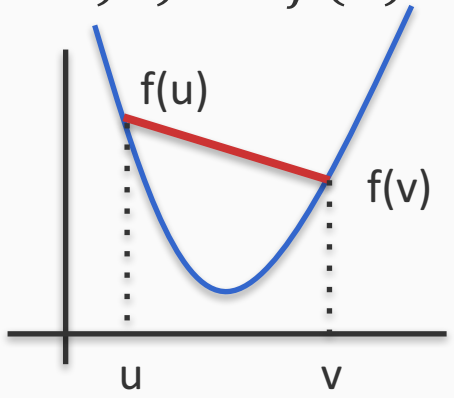

$$f(\lambda \boldsymbol{u} + (1-\lambda)\boldsymbol{v}) \geq \lambda f(\boldsymbol{u}) + (1-\lambda) f(\boldsymbol{v})$$



# 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

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




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

For convex functions, this is both necessary *and* sufficient

# Solving the SVM optimization problem

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)$$

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!

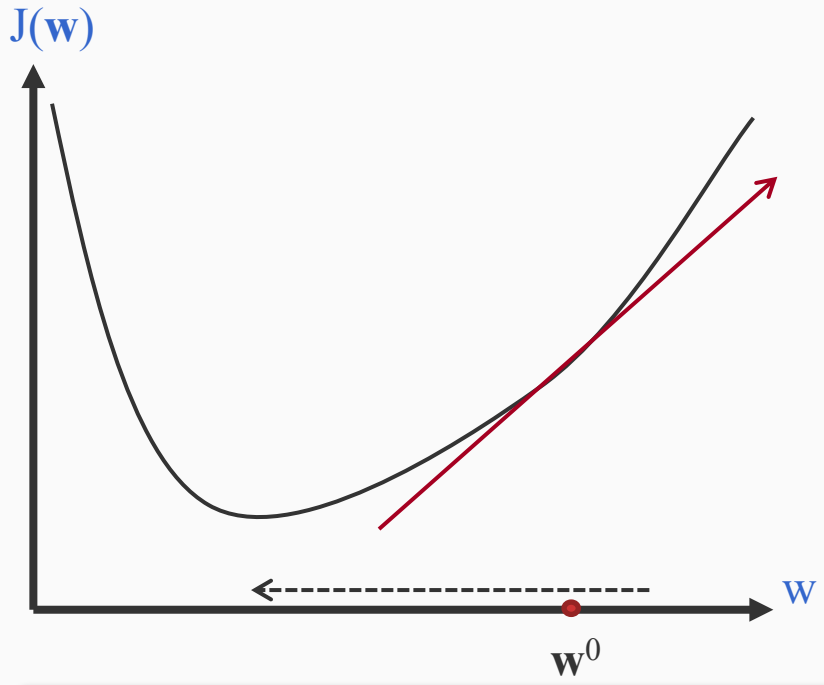
# Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

General strategy for minimizing a function J(**w**)

- Start with an initial guess for **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

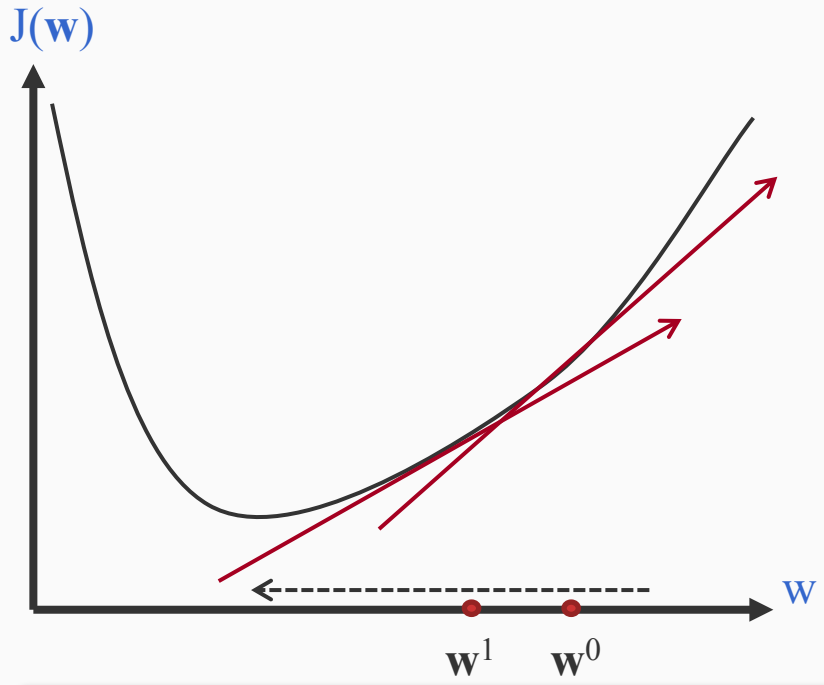
# Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

General strategy for minimizing a function J(**w**)

- Start with an initial guess for **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

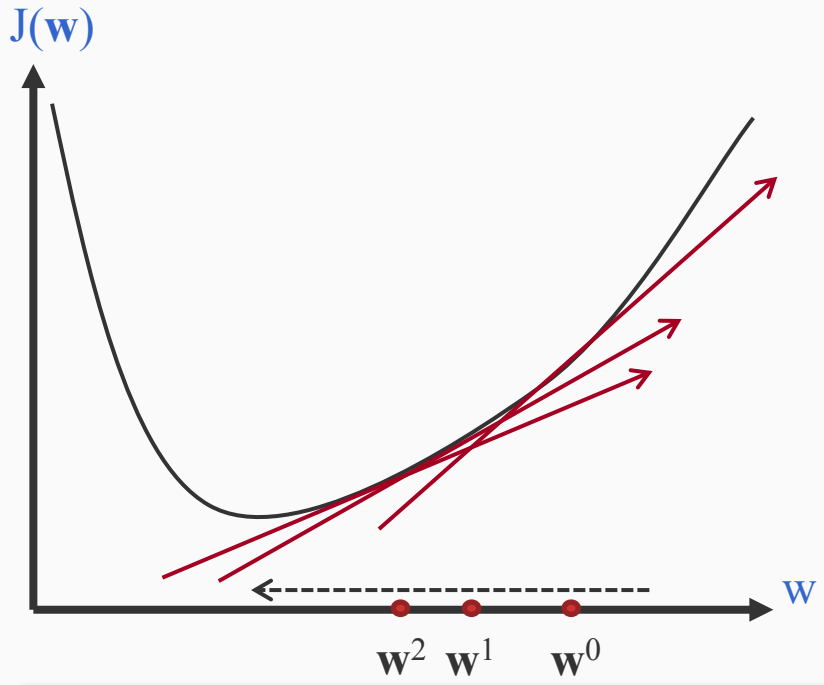
# Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

General strategy for minimizing a function J(**w**)

- Start with an initial guess for **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

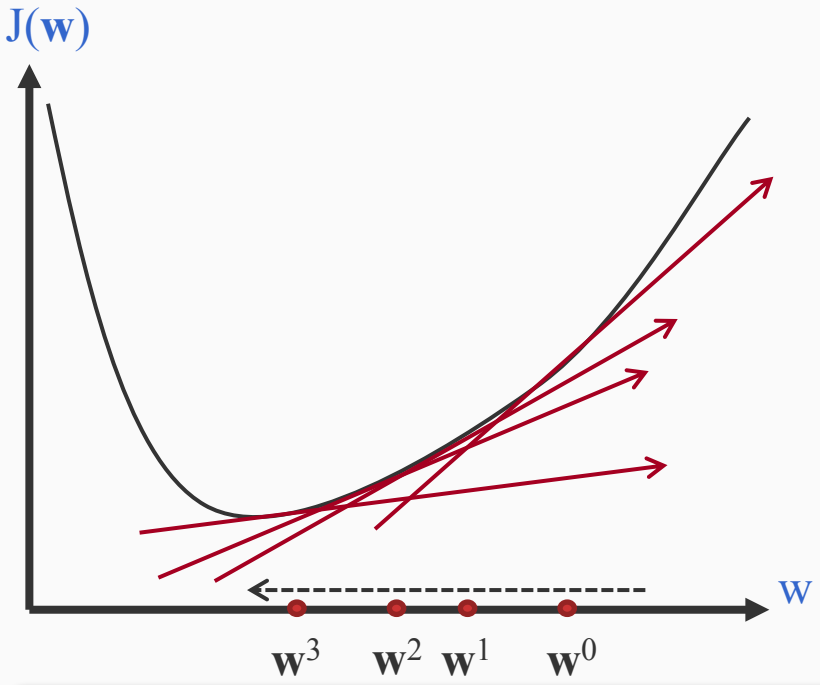
# Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

General strategy for minimizing a function J(**w**)

- Start with an initial guess for **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

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….
   1. Compute gradient of J($\mathbf{w}$) at $\mathbf{w}^t$. Call it $\nabla$ J($\mathbf{w}^t$)
   2. Update w as follows:

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

*r*: Called the learning rate .

# Outline: Training SVM by optimization

✓ 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

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….
   1. Compute gradient of J($\mathbf{w}$) at $\mathbf{w}^t$. Call it $\nabla$ J($\mathbf{w}^t$)

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

Slow, does not really scale

*r*: Called the learning rate

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

# Stochastic gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

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

3. 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 = $\{(\mathbf{x}_i, \mathrm{y}_i)\}$, $\mathbf{x} \in \Re^n$, $\mathrm{y} \in \{-1,1\}$

1. Initialize $\mathbf{w}^0 = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Pick a random example $(\mathbf{x}_i, \mathrm{y}_i)$ from the training set S

3. 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 = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w}^0 = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Pick a random example $(\mathbf{x}_i, y_i)$ from the training set S
   2. Treat $(\mathbf{x}_i, y_i)$ as a full dataset and take the derivative of the SVM objective at the current $\mathbf{w}^{t-1}$ to be $\nabla J^t(\mathbf{w}^{t-1})$
3. 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 = {($\mathbf{x}_i$, $y_i$)}, $\mathbf{x} \in \Re^n$, $y \in$ {-1,1}

1. Initialize $\mathbf{w}^0 = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Pick a random example ($\mathbf{x}_i$, $y_i$) from the training set S
   2. Treat ($\mathbf{x}_i$, $y_i$) as a full dataset and take the derivative of the SVM objective at the current $\mathbf{w}^{t-1}$ to be $\nabla J^t(\mathbf{w}^{t-1})$

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

3. 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 = {($\mathbf{x}_i$, $y_i$)}, $\mathbf{x} \in \Re^n$, $y \in$ {-1,1}

1. Initialize $\mathbf{w}^0 = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Pick a random example ($\mathbf{x}_i$, $y_i$) from the training set S
   2. Treat ($\mathbf{x}_i$, $y_i$) as a full dataset and take the derivative of the SVM objective at the current $\mathbf{w}^{t-1}$ to be $\nabla J^t(\mathbf{w}^{t-1})$

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

   3. Update: $\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \gamma_t \nabla J^t(\mathbf{w}^{t-1})$
3. 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 = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w}^0 = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Pick a random example $(\mathbf{x}_i, y_i)$ from the training set S
   2. Treat $(\mathbf{x}_i, y_i)$ as a full dataset and take the derivative of the SVM objective at the current $\mathbf{w}^{t-1}$ to be $\nabla J^t(\mathbf{w}^{t-1})$
   3. Update: $\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \gamma_t \nabla J^t(\mathbf{w}^{t-1})$
3. Return final $\mathbf{w}$

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

✓ Review of convex functions and gradient descent

✓ Stochastic gradient descent

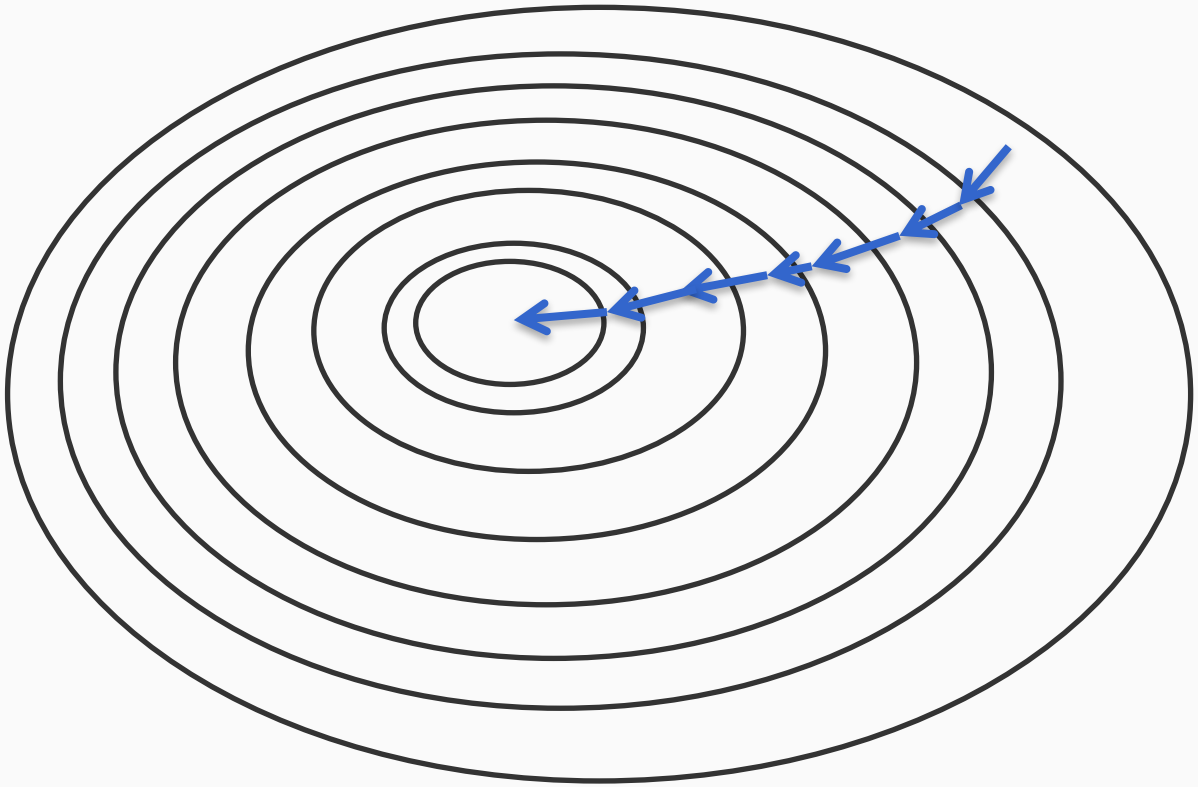
3. **Gradient descent vs stochastic gradient descent**

4. Sub-derivatives of the hinge loss

5. Stochastic sub-gradient descent for SVM

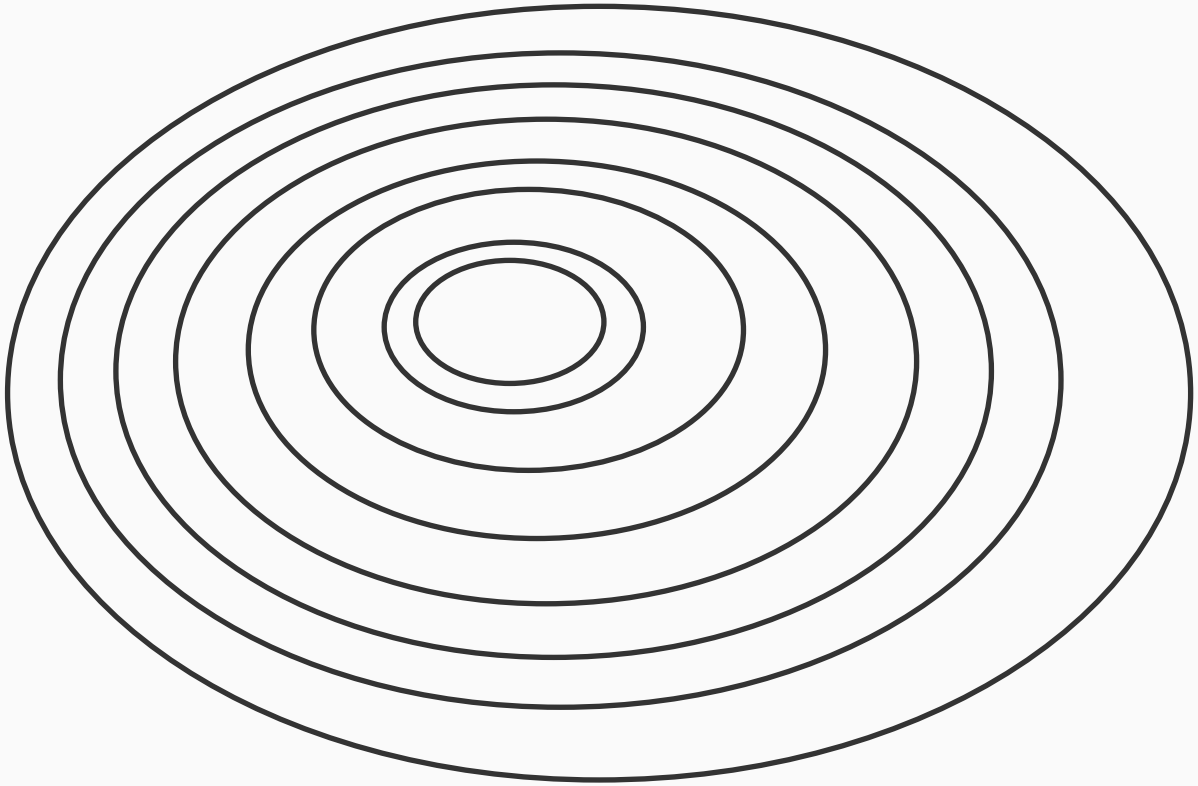
6. Comparison to perceptron

# Gradient Descent vs SGD



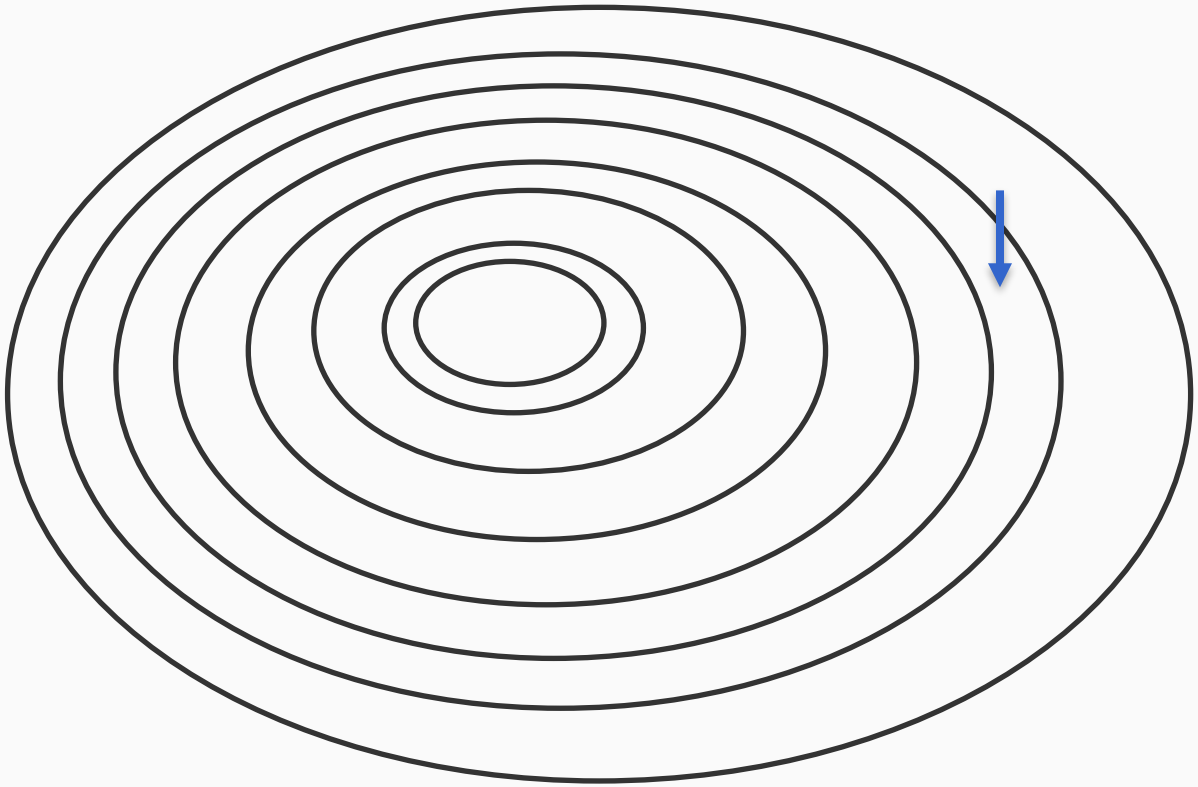
Gradient descent

# Gradient Descent vs SGD



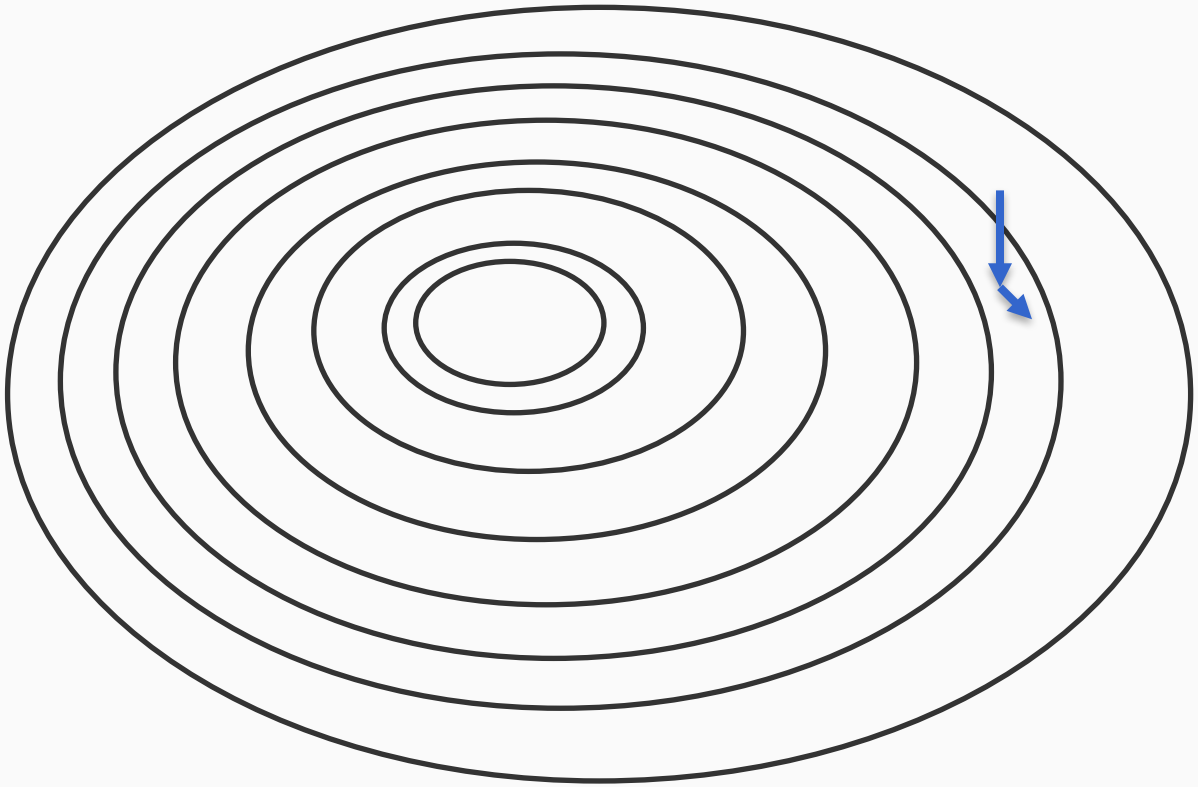
Stochastic Gradient descent

# Gradient Descent vs SGD



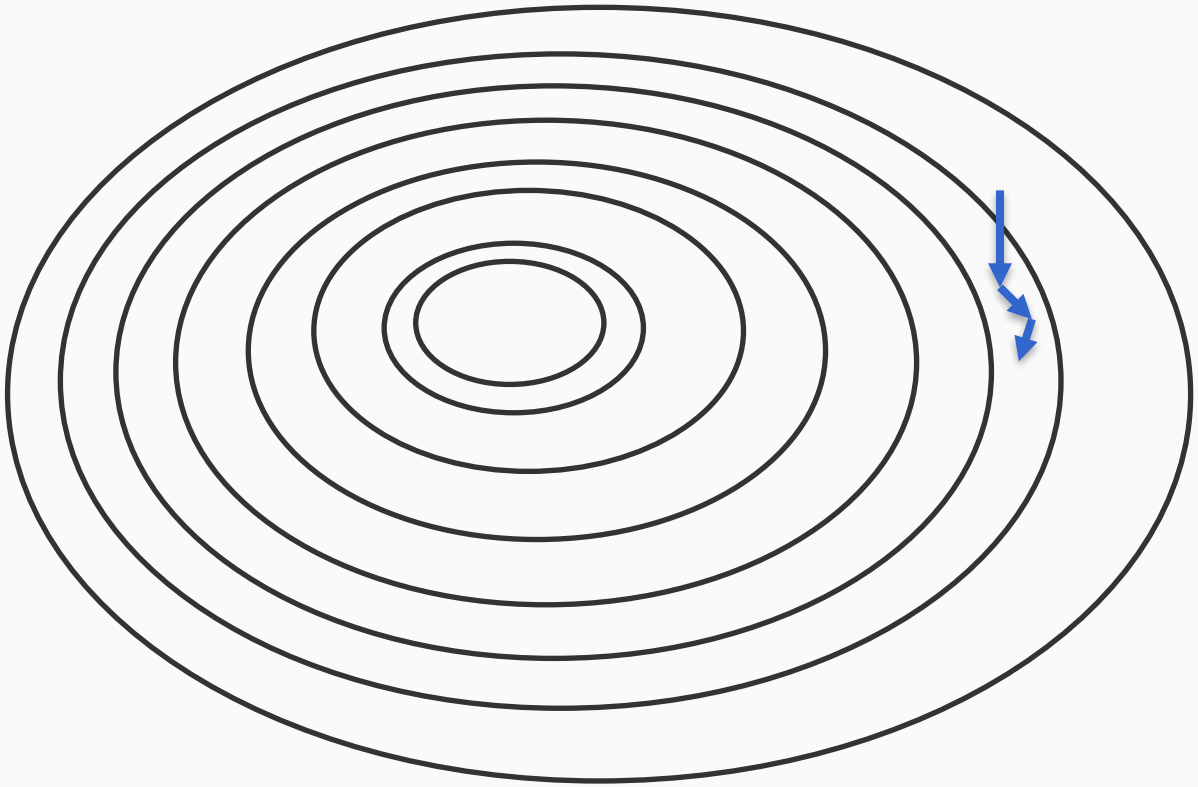
Stochastic Gradient descent

# Gradient Descent vs SGD



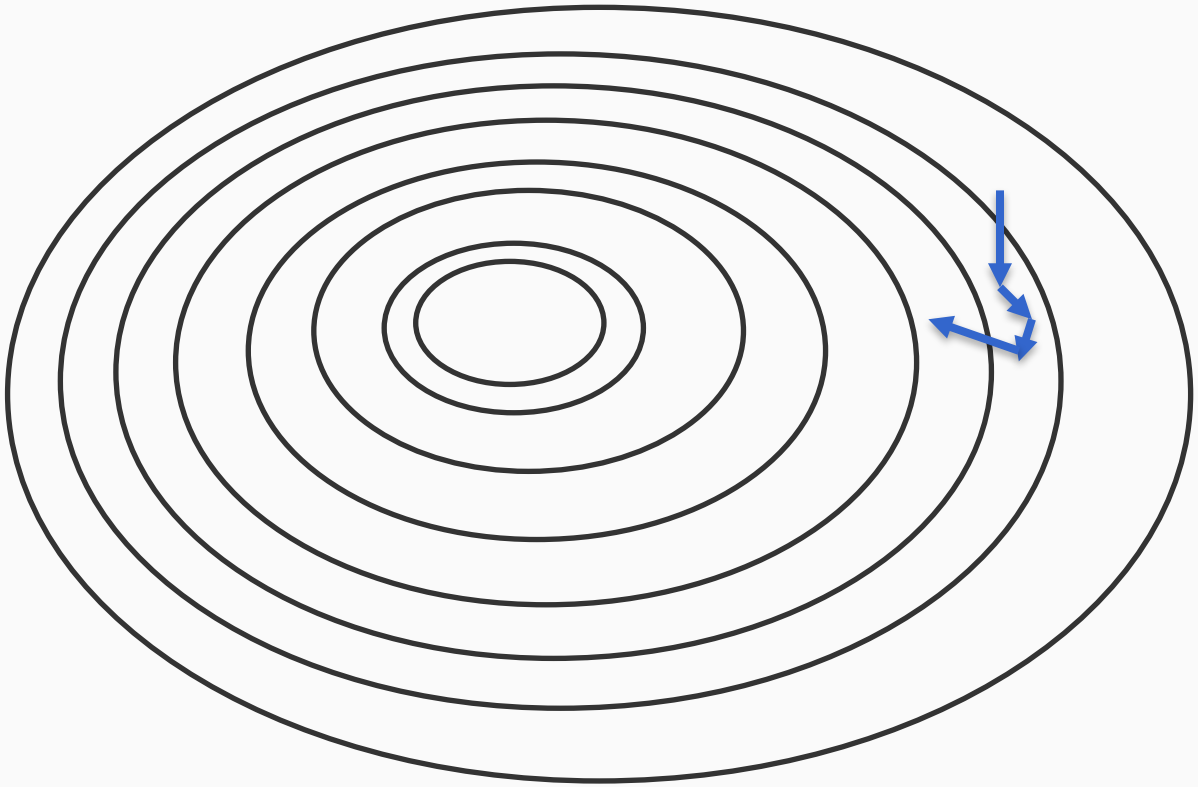
Stochastic Gradient descent

# Gradient Descent vs SGD



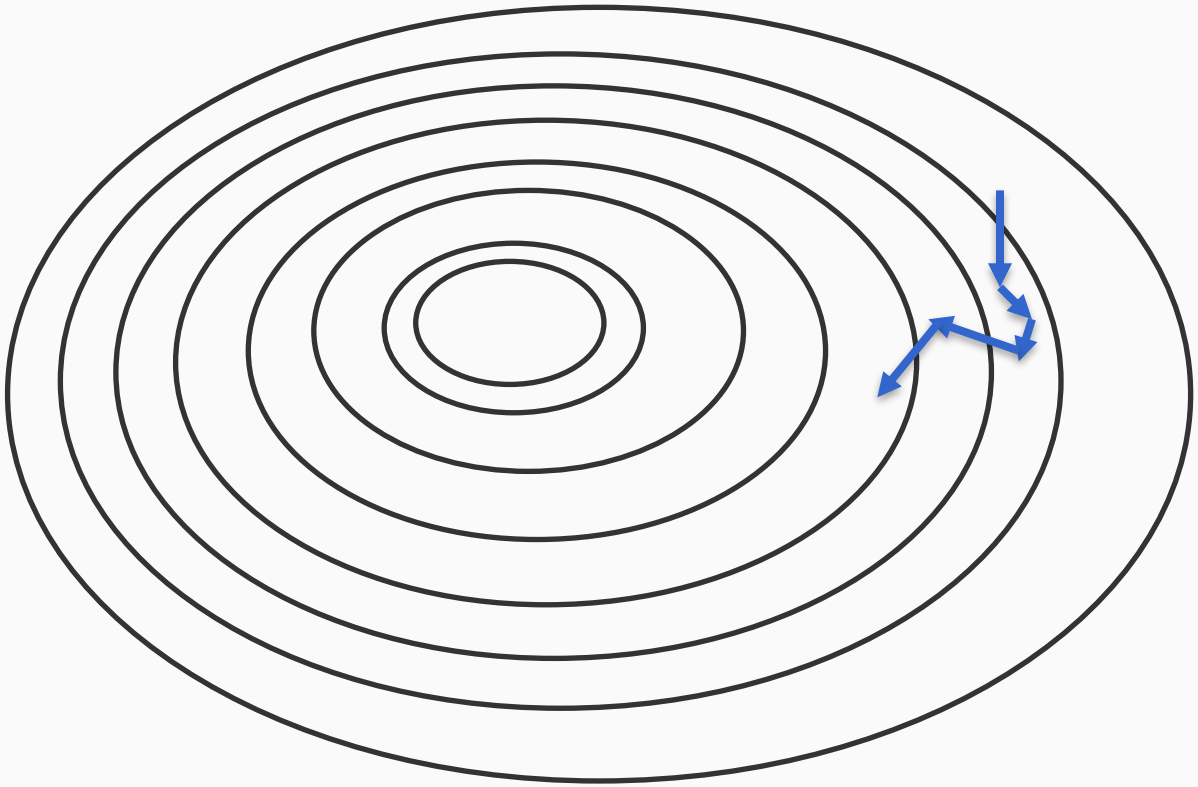
Stochastic Gradient descent

# Gradient Descent vs SGD



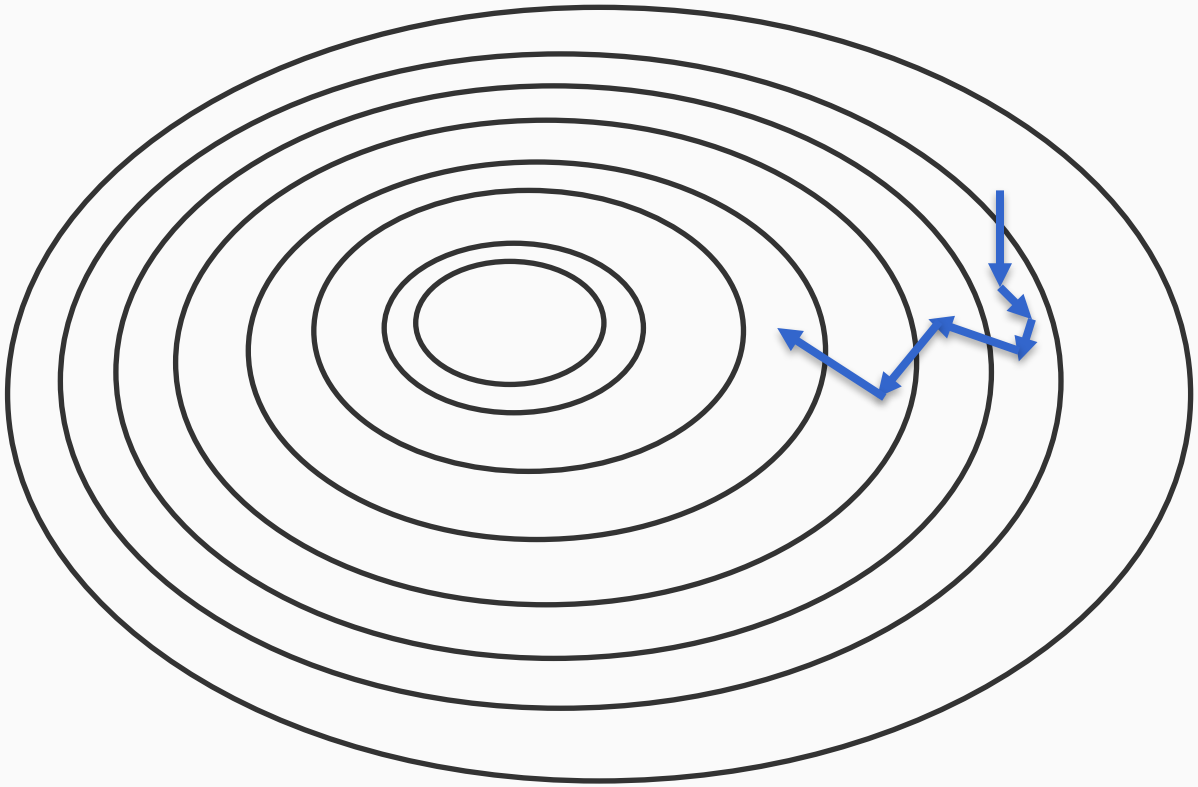
Stochastic Gradient descent

# Gradient Descent vs SGD



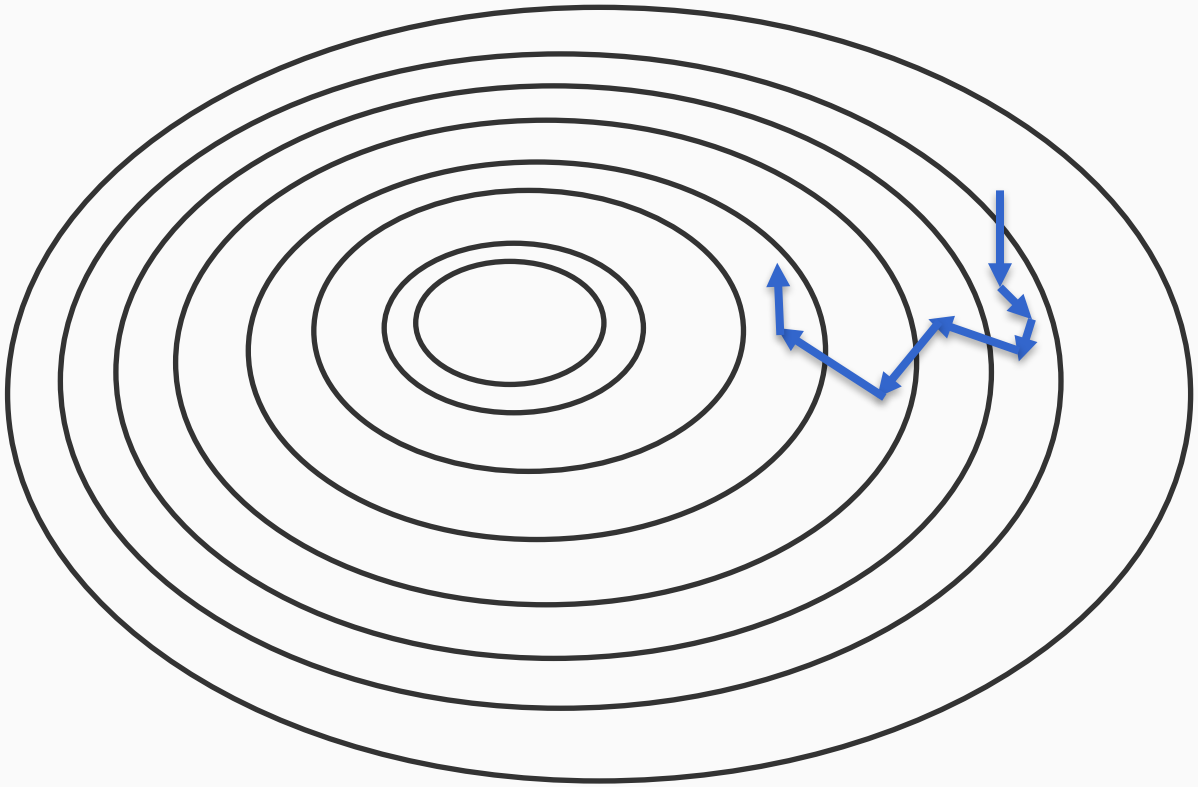
Stochastic Gradient descent

# Gradient Descent vs SGD



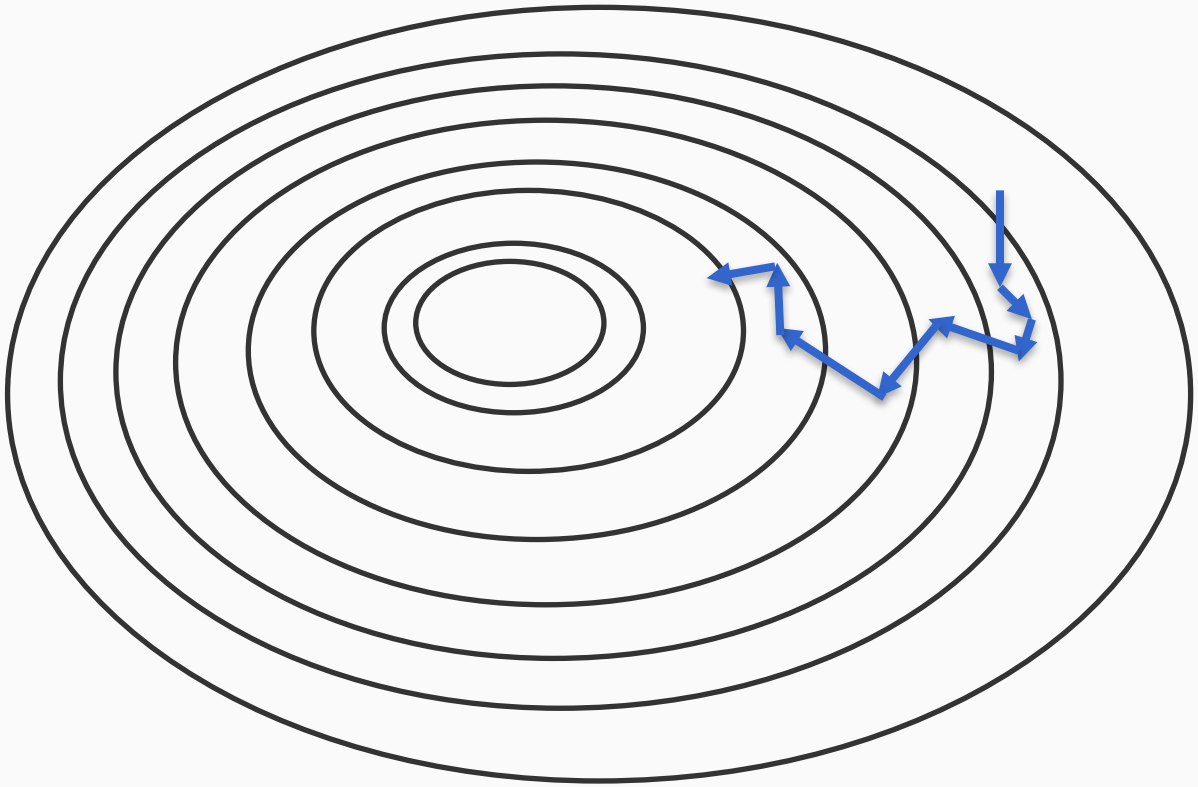
Stochastic Gradient descent

# Gradient Descent vs SGD



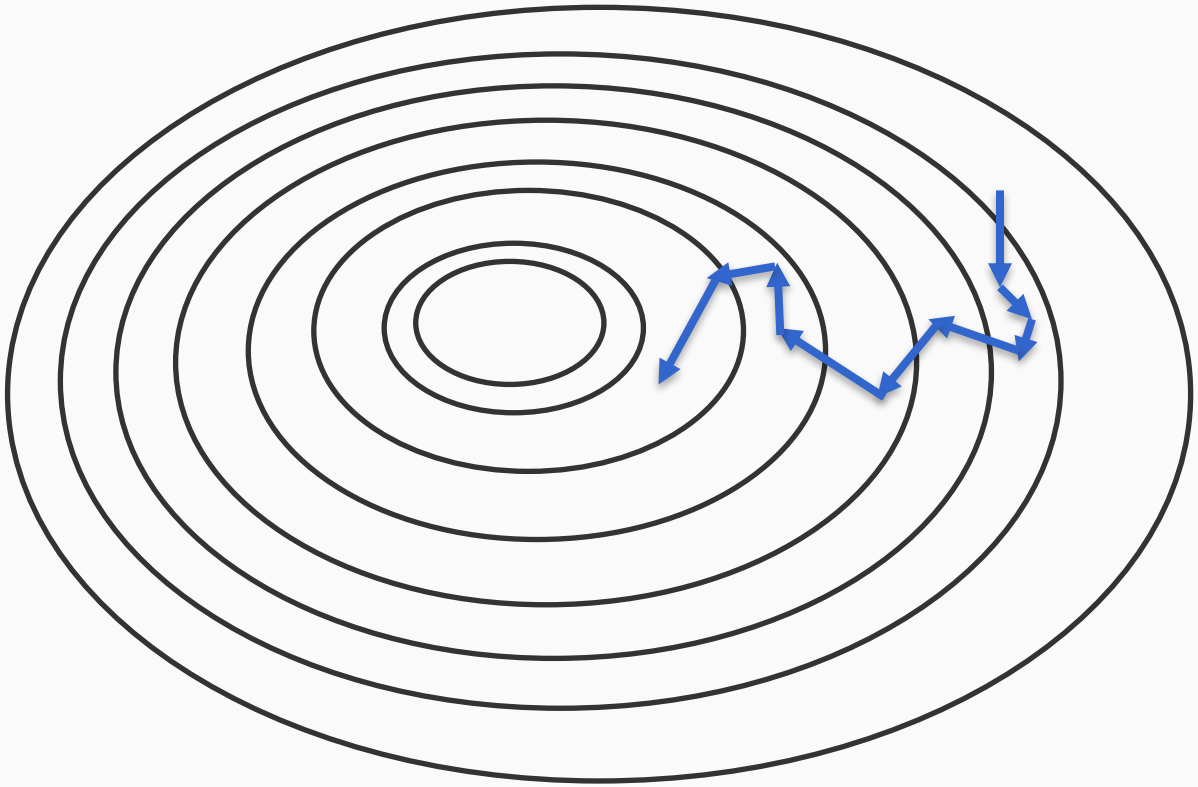
Stochastic Gradient descent

# Gradient Descent vs SGD



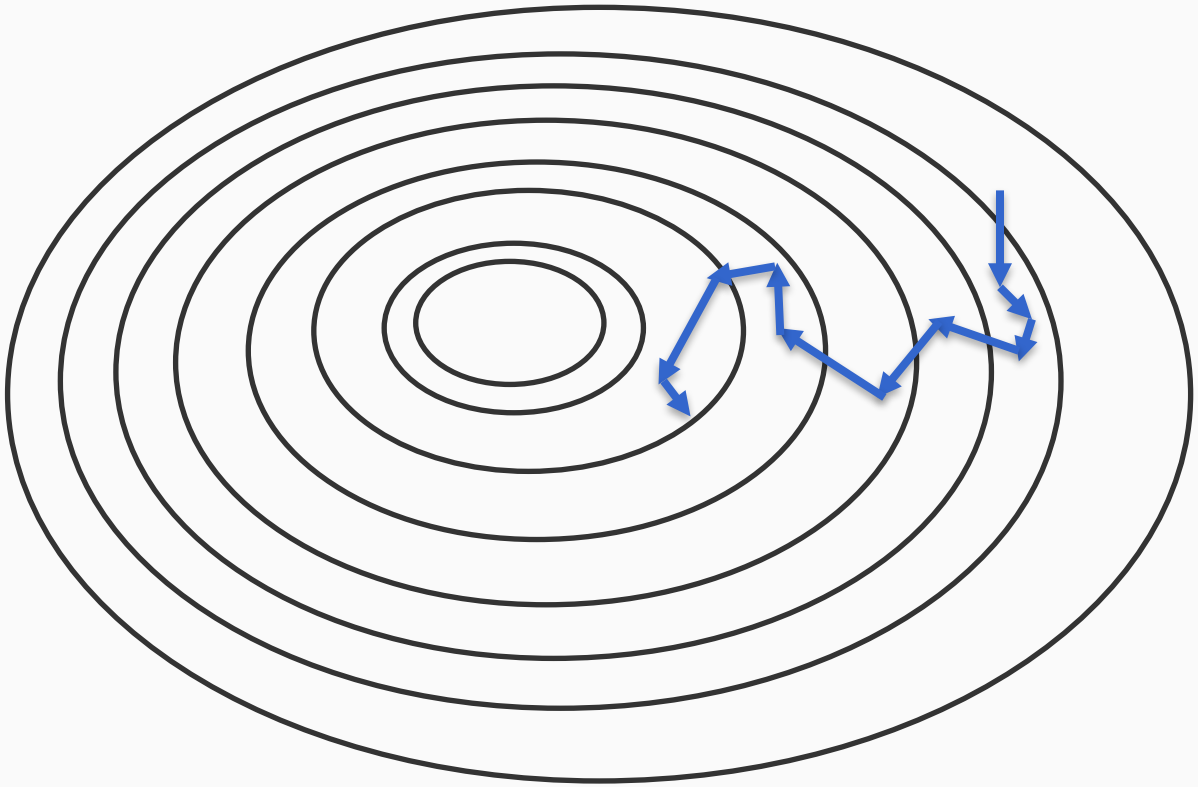
Stochastic Gradient descent

# Gradient Descent vs SGD



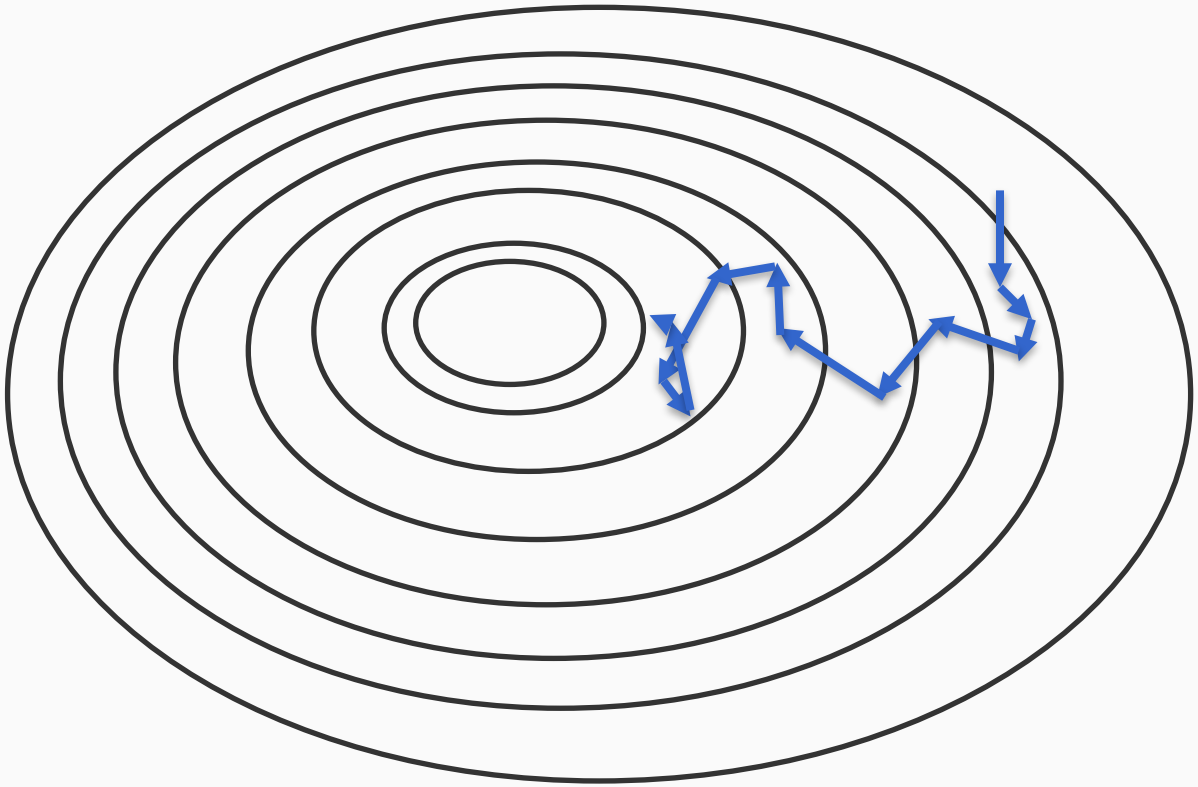
Stochastic Gradient descent

# Gradient Descent vs SGD



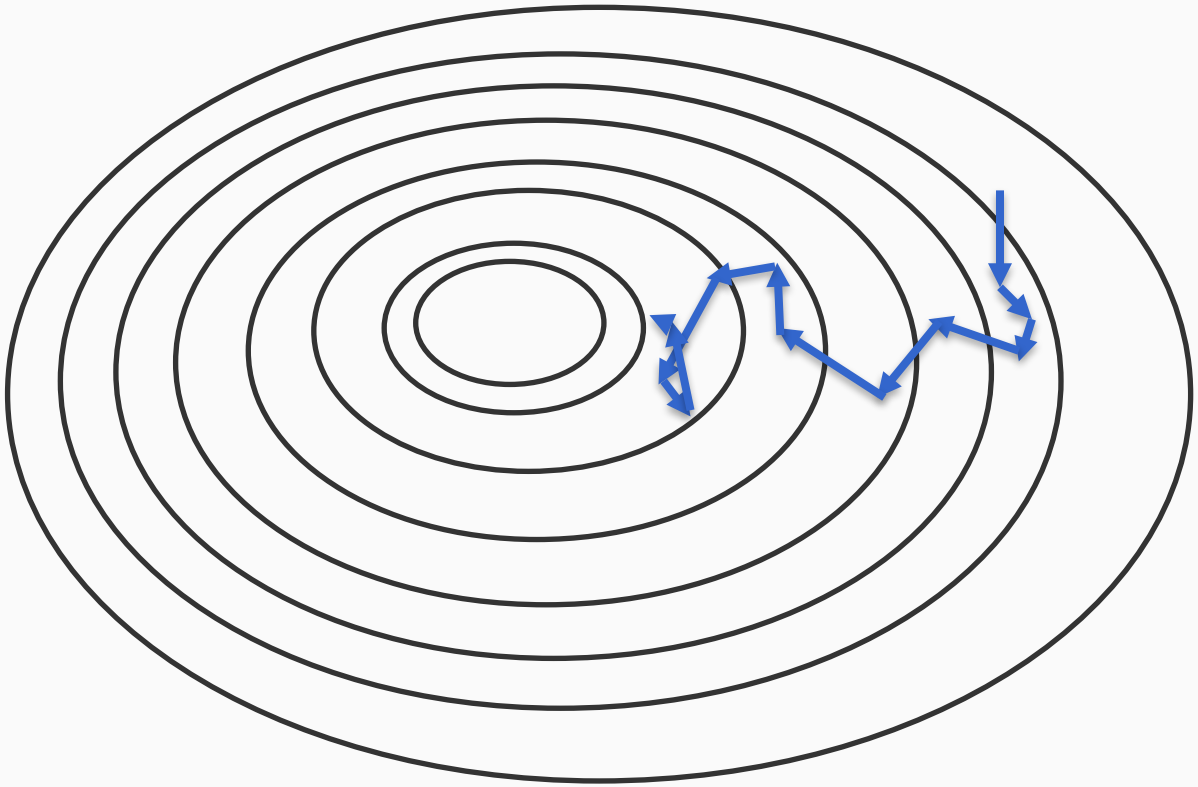
Stochastic Gradient descent

# Gradient Descent vs SGD



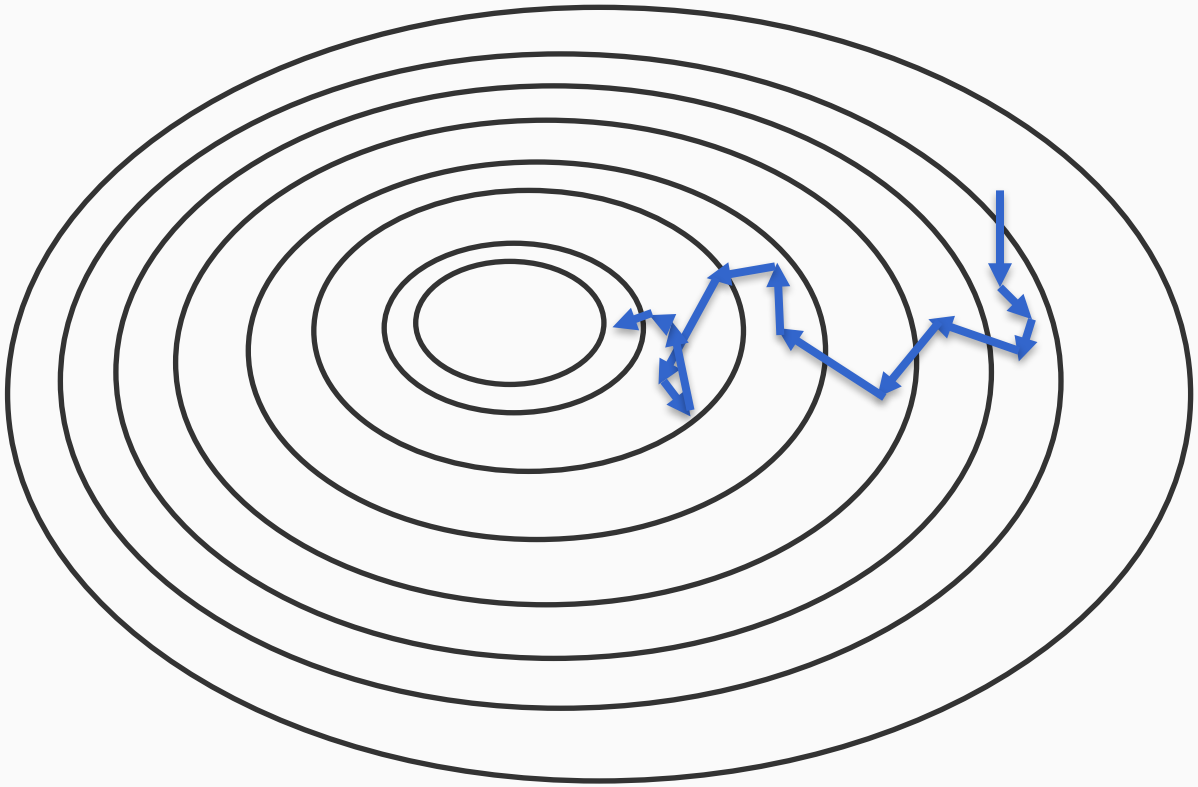
Stochastic Gradient descent

# Gradient Descent vs SGD



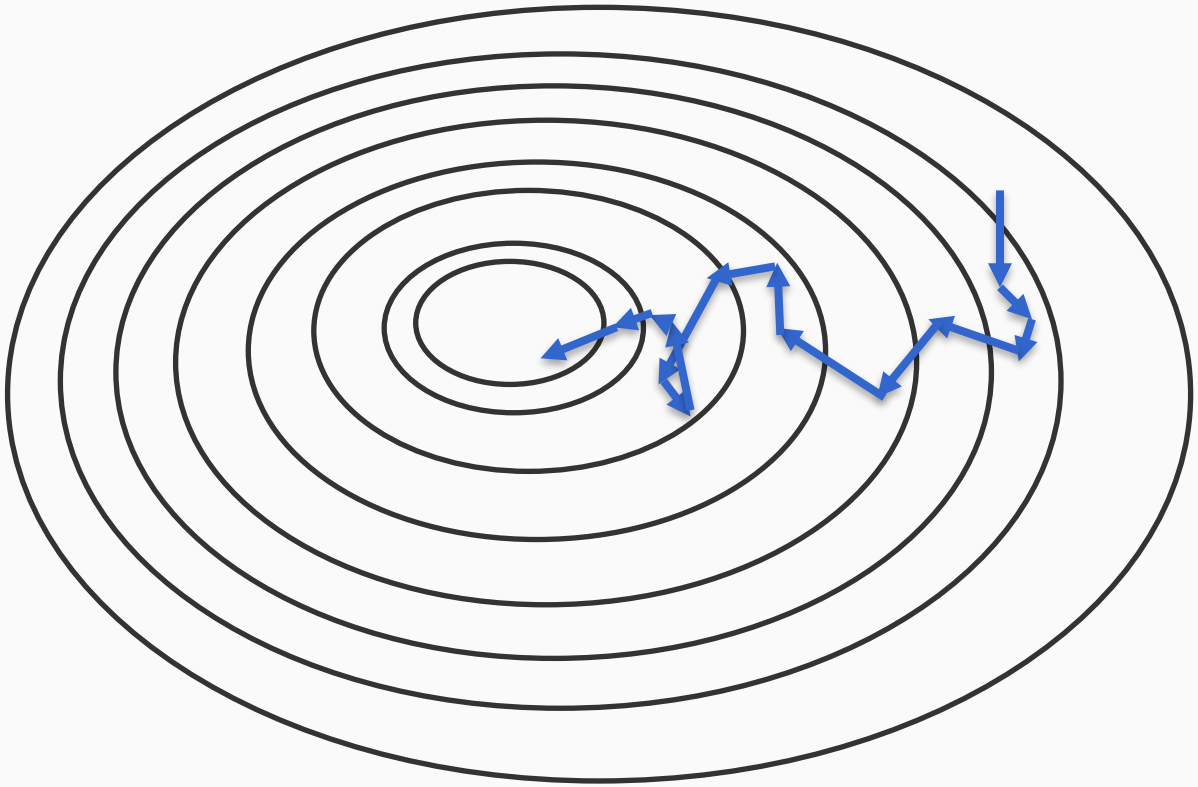
Stochastic Gradient descent

# Gradient Descent vs SGD



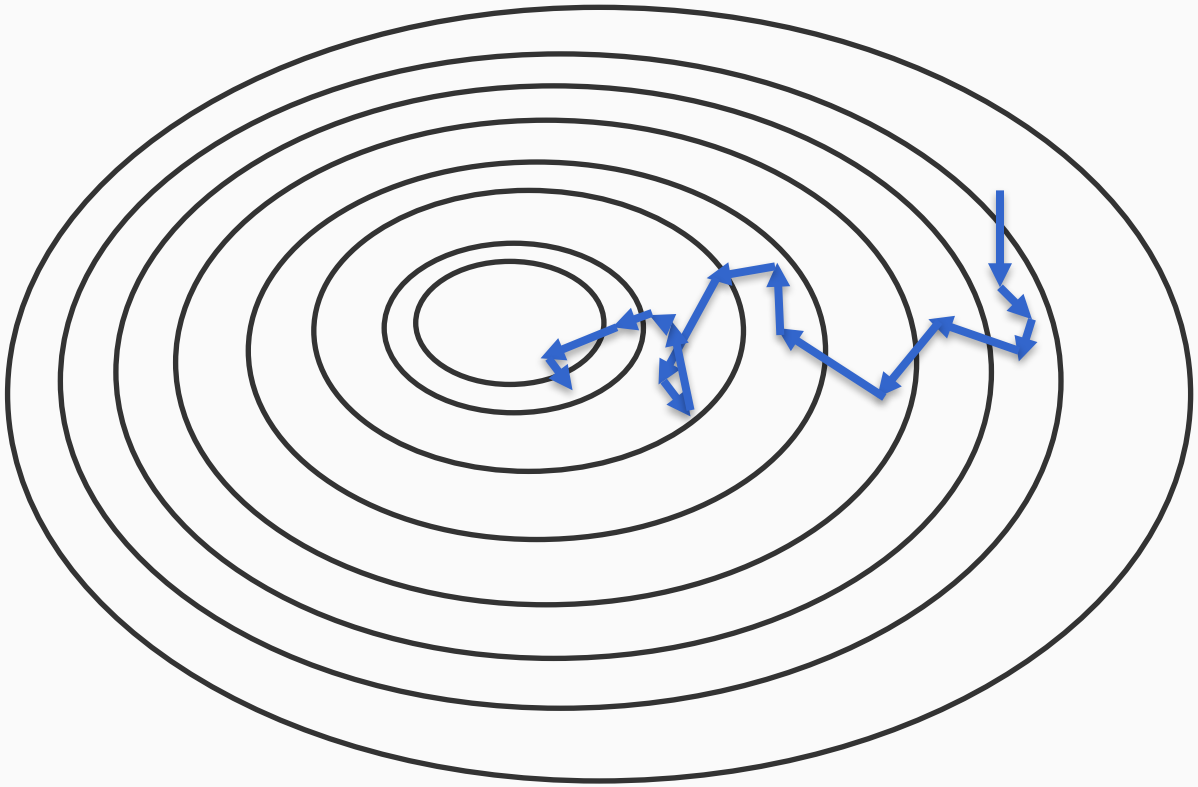
Stochastic Gradient descent

# Gradient Descent vs SGD



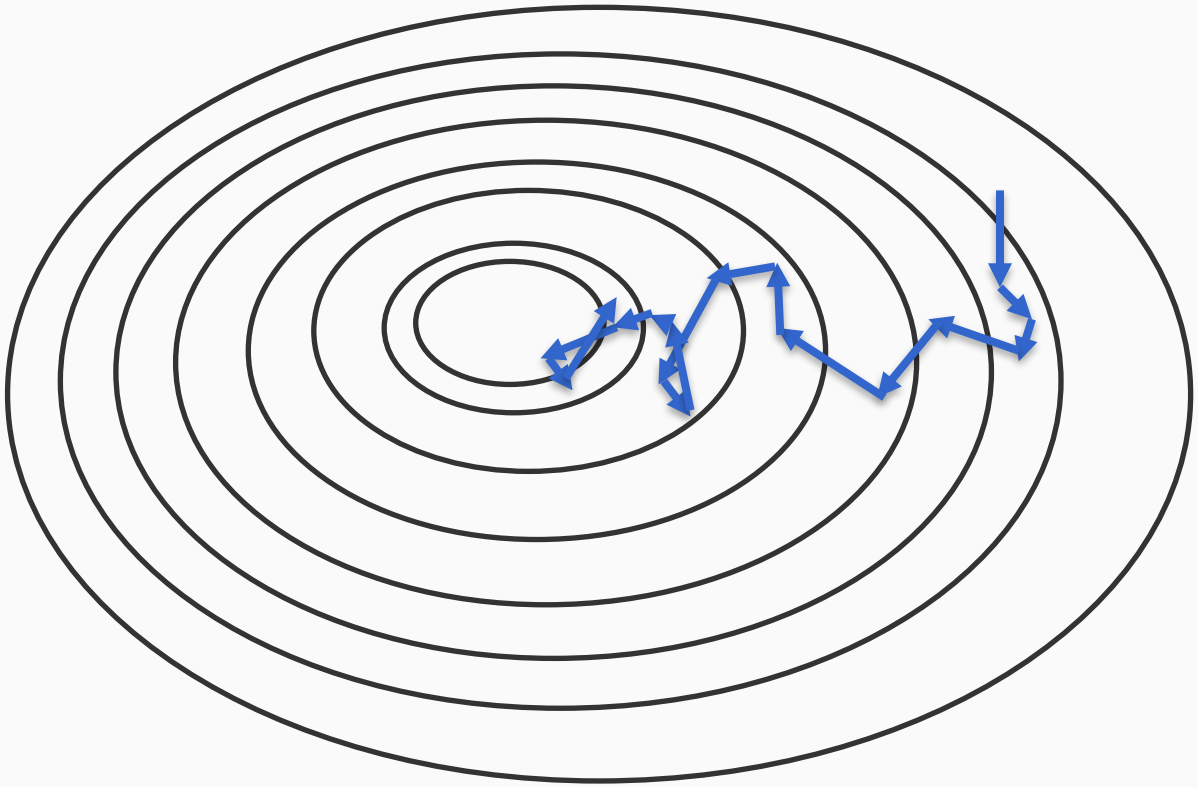
Stochastic Gradient descent

# Gradient Descent vs SGD



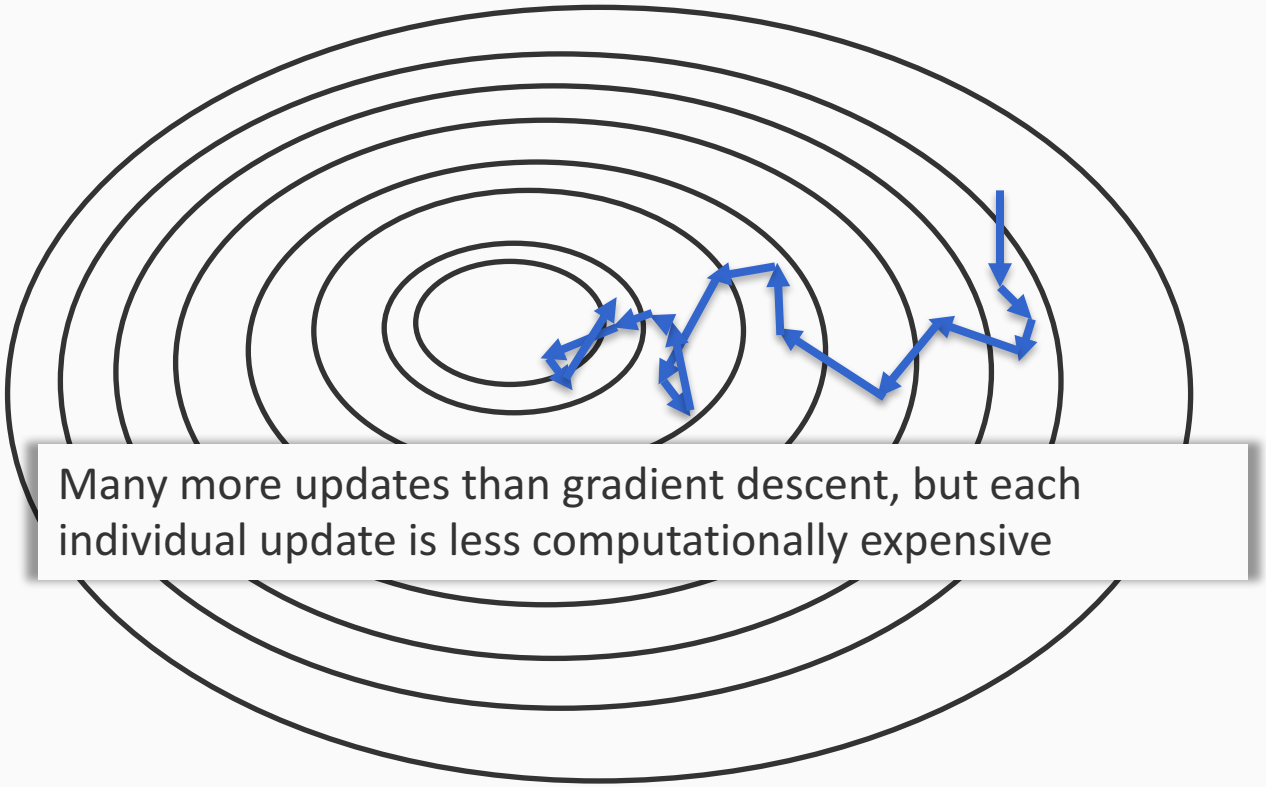
Stochastic Gradient descent

# Gradient Descent vs SGD



Stochastic Gradient descent

# Gradient Descent vs SGD




Stochastic Gradient descent

# Outline: Training SVM by optimization

✓ Review of convex functions and gradient descent

✓ Stochastic gradient descent

✓ Gradient descent vs stochastic gradient descent

4. **Sub-derivatives of the hinge loss**

5. Stochastic sub-gradient descent for SVM

6. Comparison to perceptron

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

# Stochastic gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w}^0 = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Pick a random example $(\mathbf{x}_i, y_i)$ from the training set S
   2. Treat $(\mathbf{x}_i, y_i)$ as a full dataset and take the *derivative of the SVM objective* at the current $\mathbf{w}^{t-1}$ to be $\nabla J^t(\mathbf{w}^{t-1})$
   3. Update: $\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \gamma_t \nabla J^t(\mathbf{w}^{t-1})$
3. Return final $\mathbf{w}$

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

(The hinge loss is not a differentiable function!)

# Hinge loss is not differentiable!

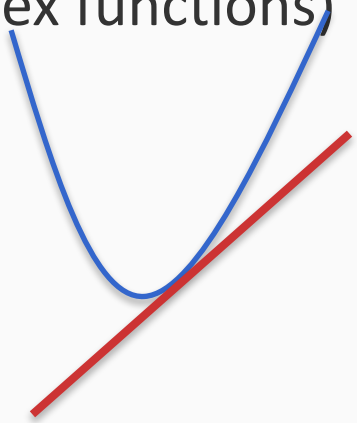
**What is the derivative of the hinge loss with respect to w?**

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

# 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)



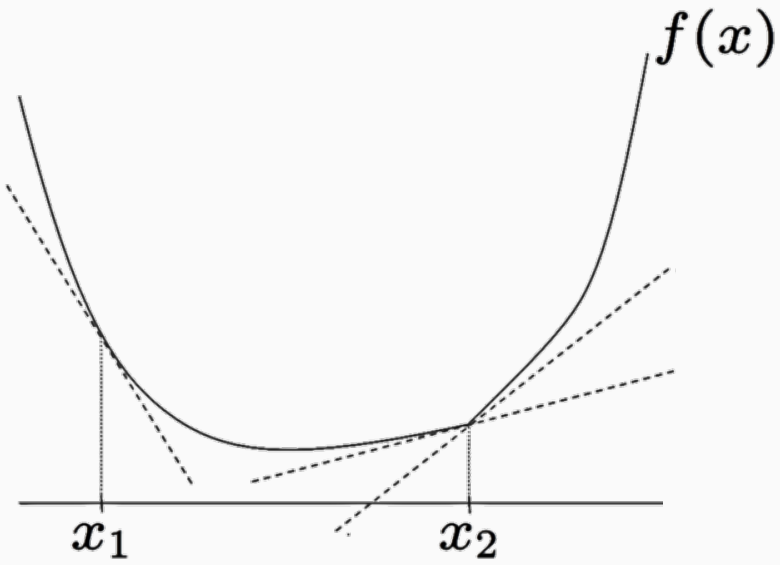
Informally, a sub-tangent at a point is any hyperplane that lies below the function at the point.

A sub-gradient is the slope of that line

# Sub-gradients

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

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




[Example from Boyd]

# Sub-gradients

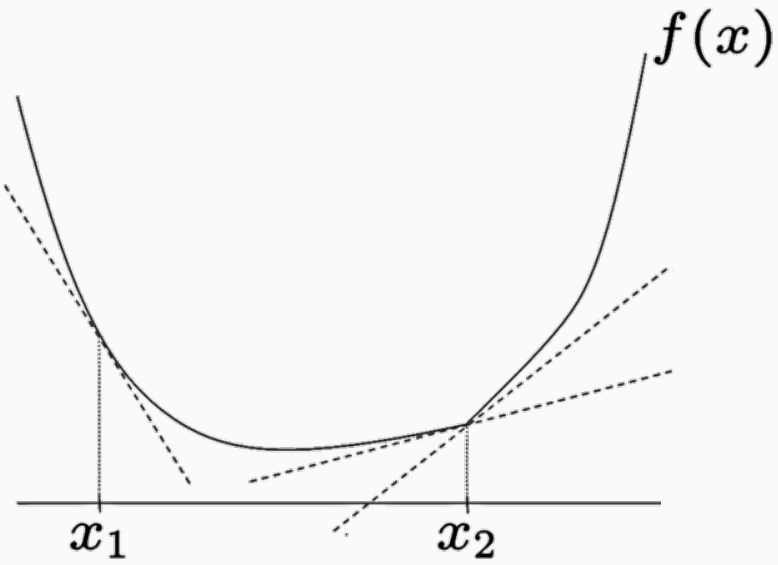
Formally, a vector g is a subgradient to f at point x if

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

f is differentiable at $x_1$
Tangent at this point

$$f(x_1) + g_1^T(x - x_1)$$

$g_1$ is a gradient at $x_1$




[Example from Boyd]

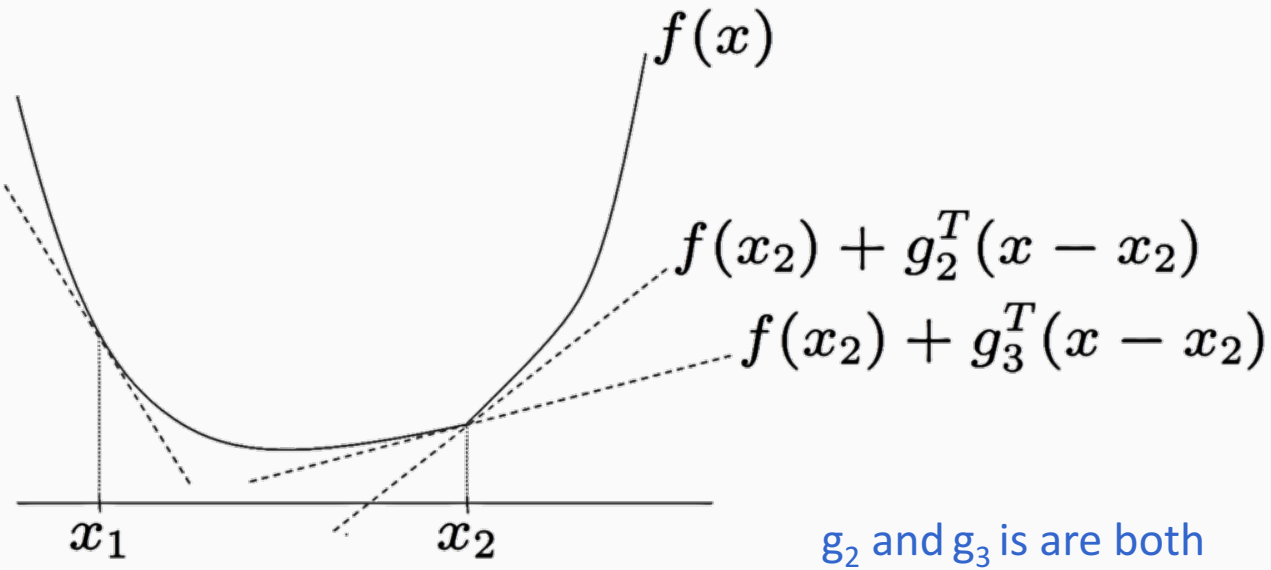
# Sub-gradients

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

$f(x_1) + g_1^T(x - x_1)$




$g_1$ is a gradient at $x_1$

$g_2$ and $g_3$ is are both subgradients at $x_2$

[Example from Boyd]

# 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

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

# Outline: Training SVM by optimization

- ✓ Review of convex functions and gradient descent
- ✓ Stochastic gradient descent
- ✓ Gradient descent vs stochastic gradient descent
- ✓ 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 = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$

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 = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
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 = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. For each training example $(\mathbf{x}_i, y_i) \in$ S:

      Update $\mathbf{w} \leftarrow \mathbf{w} - \gamma_t \nabla J^t$

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 = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. For each training example $(\mathbf{x}_i, y_i) \in$ S:

      If $y_i \mathbf{w}^T \mathbf{x}_i \leq 1$,

      $\mathbf{w} \leftarrow (1 - \gamma_t)\ \mathbf{w} + \gamma_t\ C\ y_i\ \mathbf{x}_i$

      else

      $\mathbf{w} \leftarrow (1 - \gamma_t)\ \mathbf{w}$
3. Return $\mathbf{w}$

# Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. For each training example $(\mathbf{x}_i, y_i) \in$ S:

      If $y_i \mathbf{w}^T \mathbf{x}_i \leq 1$,

      $\mathbf{w} \leftarrow (1 - \gamma_t)\ \mathbf{w} + \gamma_t\, C\, y_i\, \mathbf{x}_i$

      else

      $\mathbf{w} \leftarrow (1 - \gamma_t)\ \mathbf{w}$
3. Return $\mathbf{w}$

$\gamma_t$: learning rate, many tweaks possible

Important to shuffle examples at the start of each epoch

# Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Shuffle the training set
   2. For each training example $(\mathbf{x}_i, y_i) \in$ S:

      If $y_i \mathbf{w}^T \mathbf{x}_i \leq 1$,

      $\mathbf{w} \leftarrow (1 - \gamma_t)\ \mathbf{w} + \gamma_t\, C\, y_i\, \mathbf{x}_i$

      else

      $\mathbf{w} \leftarrow (1 - \gamma_t)\ \mathbf{w}$
3. Return $\mathbf{w}$

$\gamma_t$: learning rate, many tweaks possible

# 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 $\infty$ is not infinite
- Sum of step sizes over t = 1 to $\infty$ 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(Nd ln(1/$\epsilon$))
  - Stochastic gradient descent: O(d/$\epsilon$)
- 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(Nd ln(1/$\epsilon$))
  - Stochastic gradient descent: O(d/$\epsilon$)
- More subtleties involved, but SGD is generally preferable when the data size is huge
- Recently, many variants that are based on this general strategy, targeting multilayer neural networks
  - Examples: Adagrad, momentum, Nesterov's accelerated gradient, Adam, RMSProp, etc…

# Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Shuffle the training set
   2. For each training example $(\mathbf{x}_i, y_i) \in$ S:

      If $y_i \mathbf{w}^T \mathbf{x}_i \leq 1$,

      $\mathbf{w} \leftarrow (1-\gamma_t)\ \mathbf{w} + \gamma_t\, C\, y_i\, \mathbf{x}_i$

      else

      $\mathbf{w} \leftarrow (1-\gamma_t)\ \mathbf{w}$
3. Return $\mathbf{w}$

# Outline: Training SVM by optimization

- ✓ Review of convex functions and gradient descent
- ✓ Stochastic gradient descent
- ✓ Gradient descent vs stochastic gradient descent
- ✓ Sub-derivatives of the hinge loss
- ✓ Stochastic sub-gradient descent for SVM

**6. Comparison to perceptron**

# Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Shuffle the training set
   2. For each training example $(\mathbf{x}_i, y_i) \in$ S:

      If $y_i \, \mathbf{w}^T \mathbf{x}_i \leq 1$,

      $\mathbf{w} \leftarrow (1\text{-}\gamma_t) \, \mathbf{w} + \gamma_t \, C \, y_i \, \mathbf{x}_i$

      else

      $\mathbf{w} \leftarrow (1\text{-}\gamma_t) \, \mathbf{w}$
3. Return $\mathbf{w}$

Compare with the Perceptron update:
If $y \, \mathbf{w}^T \mathbf{x} \leq 0$, update $\mathbf{w} \leftarrow \mathbf{w} + r \, y \, \mathbf{x}$