Support Vector Machines: Training with Stochastic Gradient Descent

Machine Learning
Spring 2018

THE UNIVERSITY OF UTAH

The slides are mainly from Vivek Srikumar

Support vector machines

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

SVM objective function

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i} \max\left(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\right)$$

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

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

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C \sum_i \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right)$$

This function is convex in **w,** b

For convenience, use simplified notation:

$\mathbf{w_0} \leftarrow \mathbf{w}$

$\mathbf{w} \leftarrow [\mathbf{w_0}, b]$

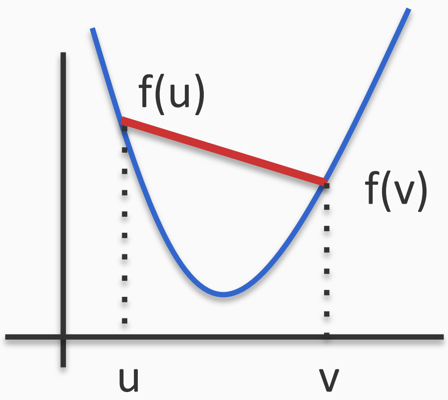
$\boldsymbol{x}_i \leftarrow [\boldsymbol{x}_i, 1]$

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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

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

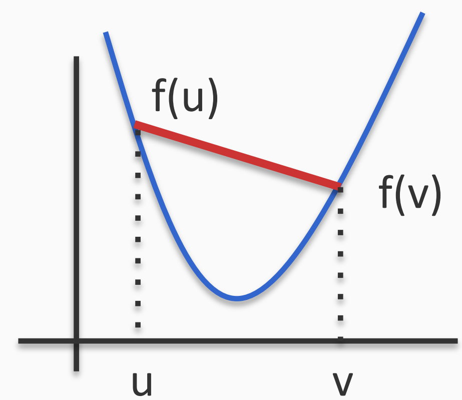


Recall: Convex functions

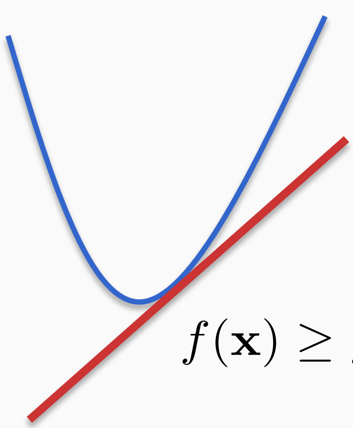
A function f is convex if for every $\boldsymbol{u}, \boldsymbol{v}$ in the domain, and for every $\lambda \in [0,1]$ we have

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

From geometric perspective



Every tangent plane lies below the function

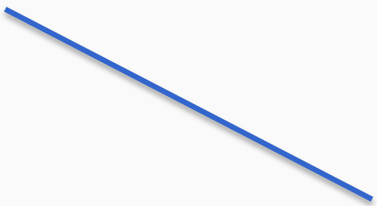


$$f(\mathbf{x}) \geq f(\mathbf{u}) + \nabla f(u)^\top (\mathbf{x} - \mathbf{u})$$

Convex functions

$$f(x) = -x$$

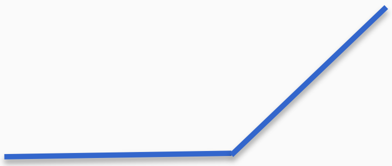
Linear functions



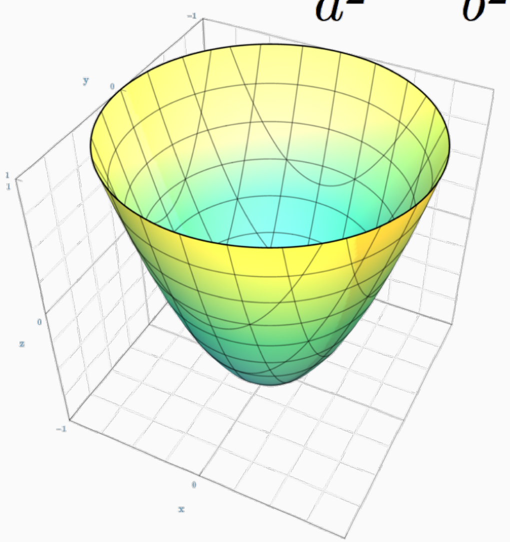
$$f(x) = x^2$$

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

max is convex



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



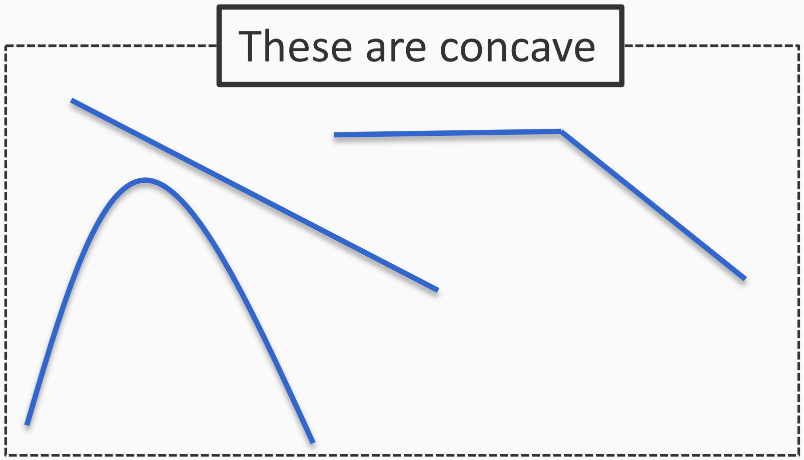
Some ways to show that a function is convex:

1. Using the definition of convexity

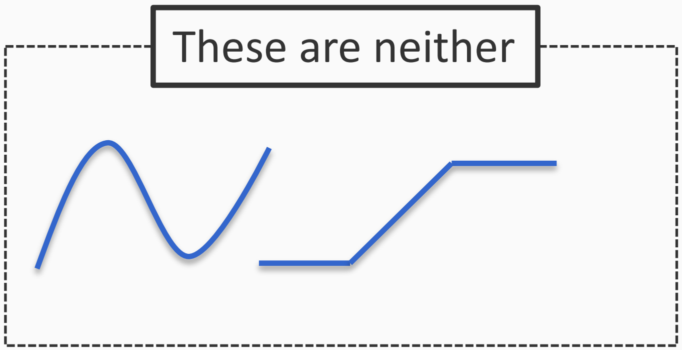
2. Showing that the second derivative is nonnegative (for one dimensional functions)

3. Showing that the second derivative is positive semi-definite (for vector functions)

Not all functions are convex



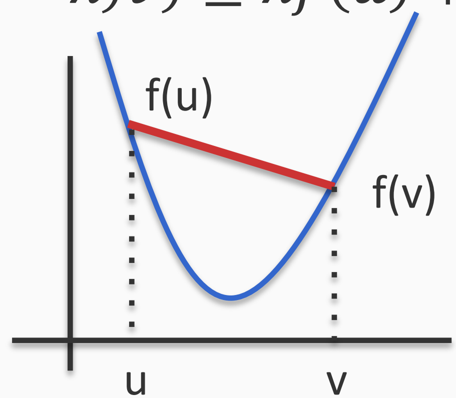
$$f(\lambda \boldsymbol{u} + (1-\lambda)\boldsymbol{v}) \geq \lambda f(\boldsymbol{u}) + (1-\lambda) f(\boldsymbol{v})$$



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

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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!

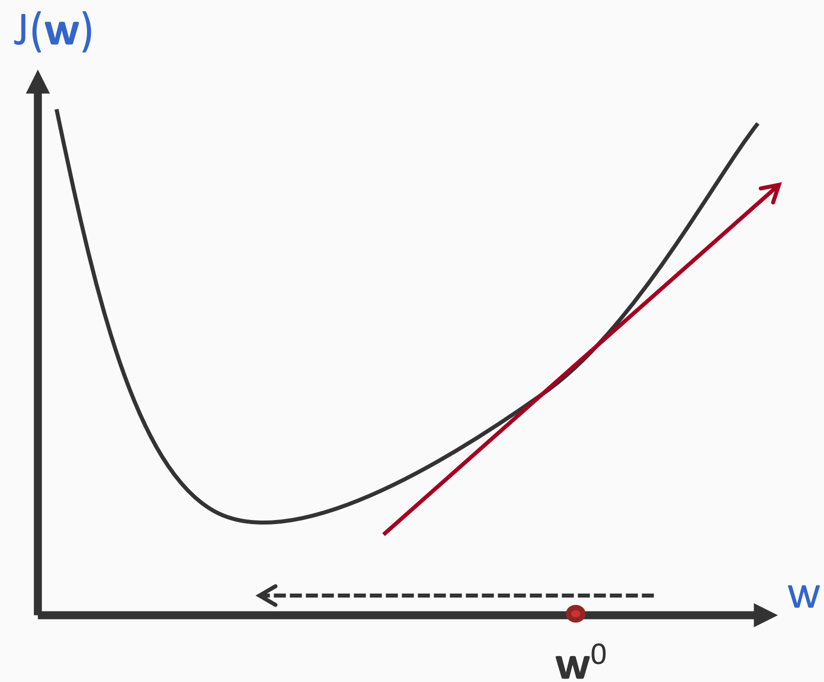
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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 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

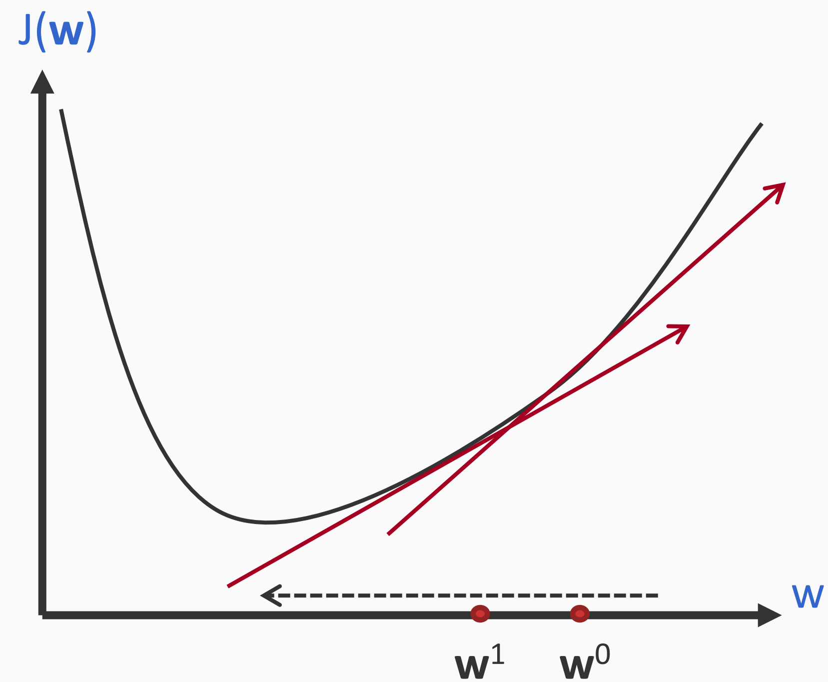
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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 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

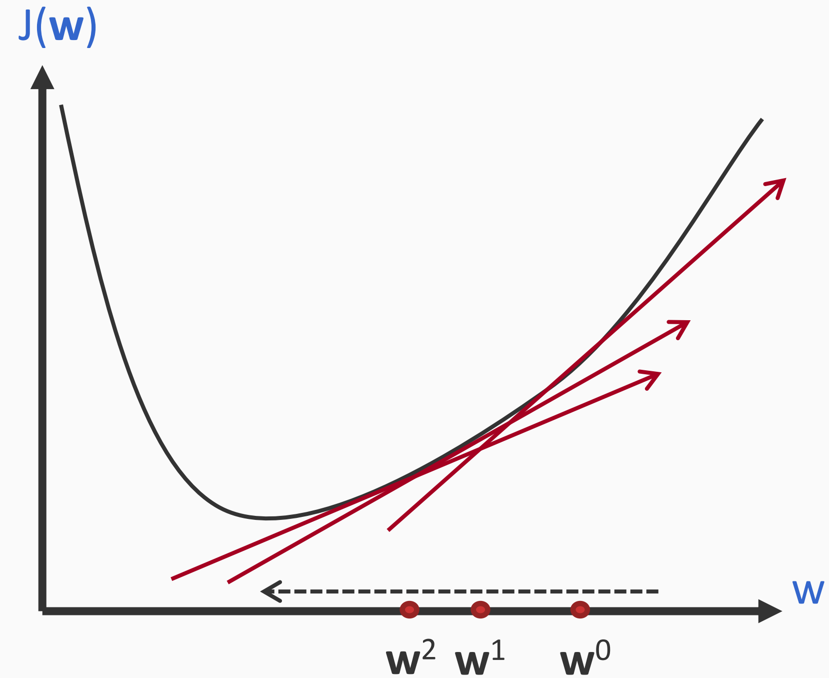
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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 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

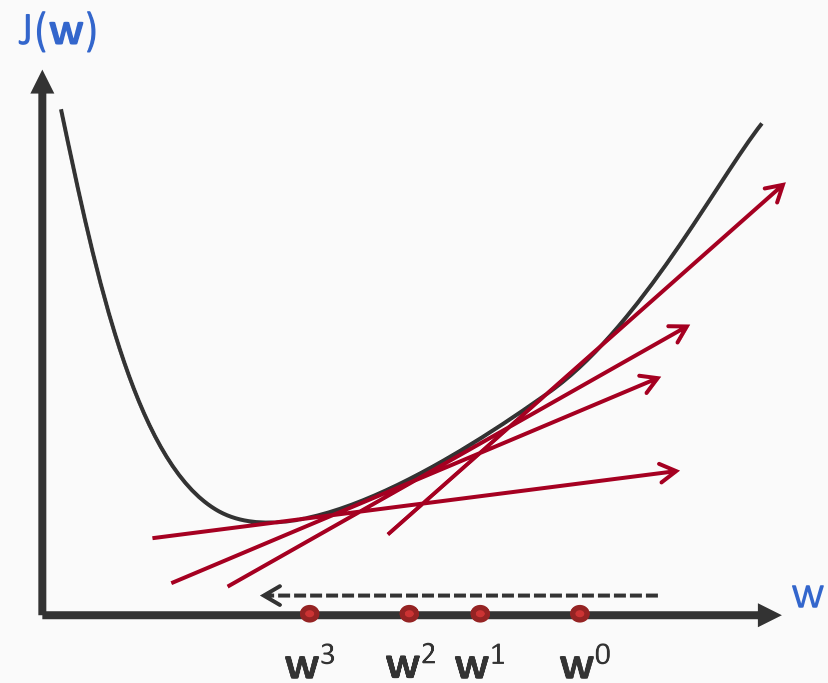
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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 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

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….
   1. Compute gradient of J(**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

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….
   1. Compute gradient of J(**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

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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 **w**

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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

3. Return final **w**

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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. Repeat ($\mathbf{x}_i$, y_i) to make 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 **w**

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

Stochastic gradient descent for SVM

Given a training set S = {($\mathbf{x}_i$, y_i)}, $\mathbf{x} \in \Re^n$, y ∈ {-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. Repeat ($\mathbf{x}_i$, y_i) to make a full dataset and take the derivative of the SVM objective at the current $\mathbf{w}^{t-1}$ to be $\nabla J^t(\mathbf{w}^{t-1})$

$$J^t(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \cdot N \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

3. Return final **w**

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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. Repeat ($\mathbf{x}_i$, y_i) to make a full dataset and take the derivative of the SVM objective at the current $\mathbf{w}^{t-1}$ to be $\nabla J^t(\mathbf{w}^{t-1})$

$$J^t(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \cdot N \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

Number of training examples

3. Return final **w**

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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. Repeat $(\mathbf{x}_i, y_i)$ to make a full dataset and take the derivative of the SVM objective at the current $\mathbf{w}^{t-1}$ to be $\nabla J^t(\mathbf{w}^{t-1})$

$$J^t(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \cdot N \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

    3. Update: $\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \gamma_t \nabla J^t(\mathbf{w}^{t-1})$
3. Return final $\mathbf{w}$

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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. Repeat $(\mathbf{x}_i, y_i)$ to make 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 γ_t is small enough.

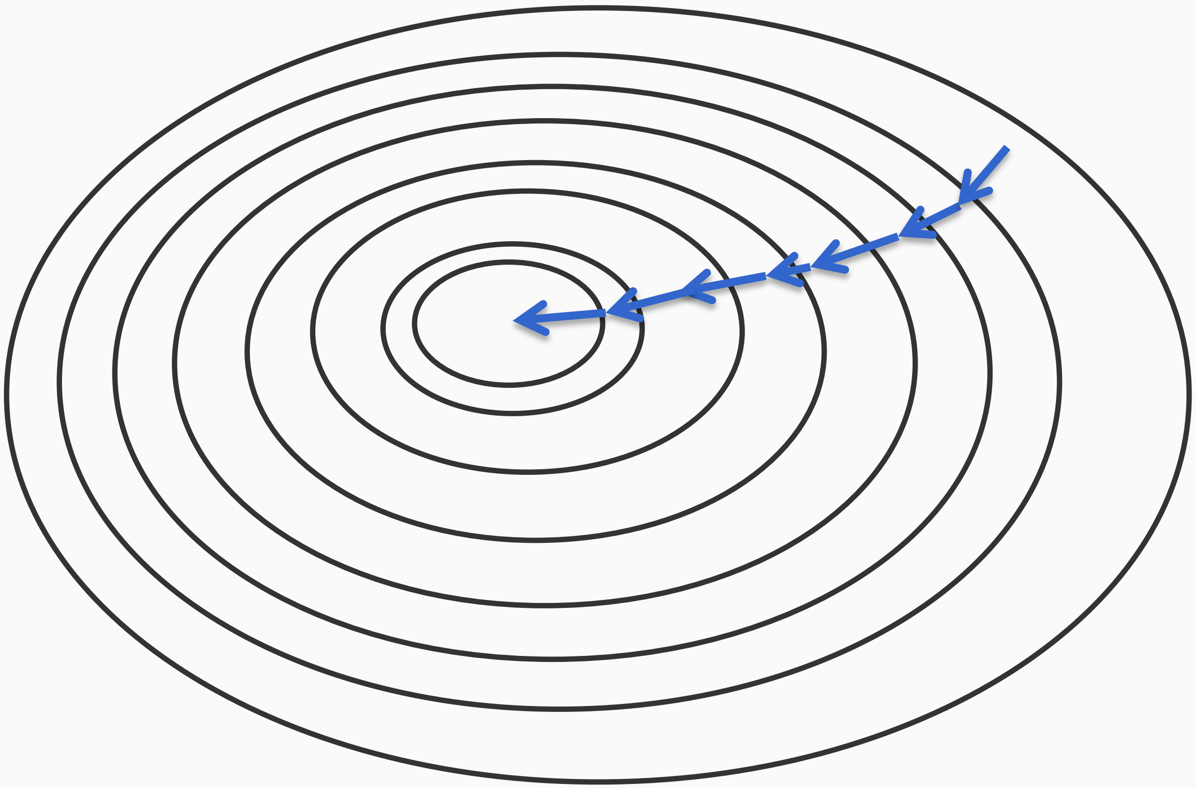
Outline: Training SVM by optimization

✓ Review of convex functions and gradient descent

✓ Stochastic gradient descent

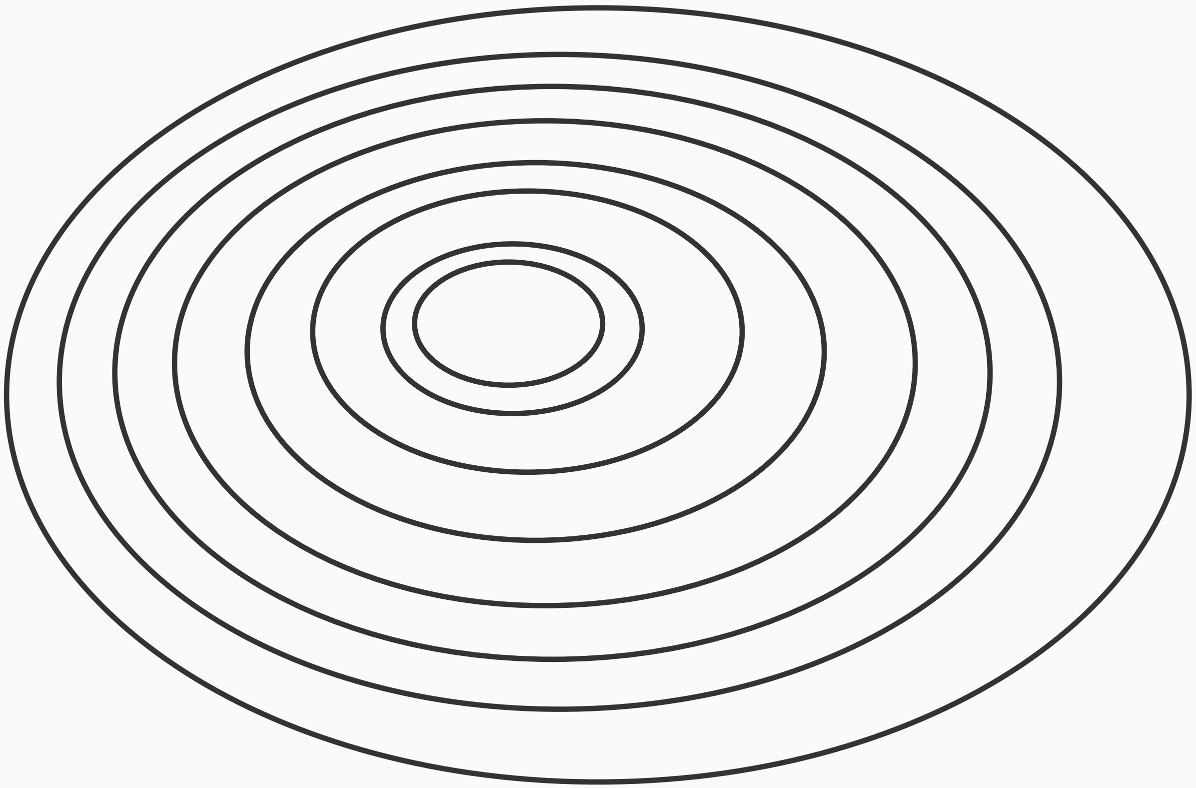
3. **Gradient descent vs stochastic gradient descent**
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron

Gradient Descent vs SGD



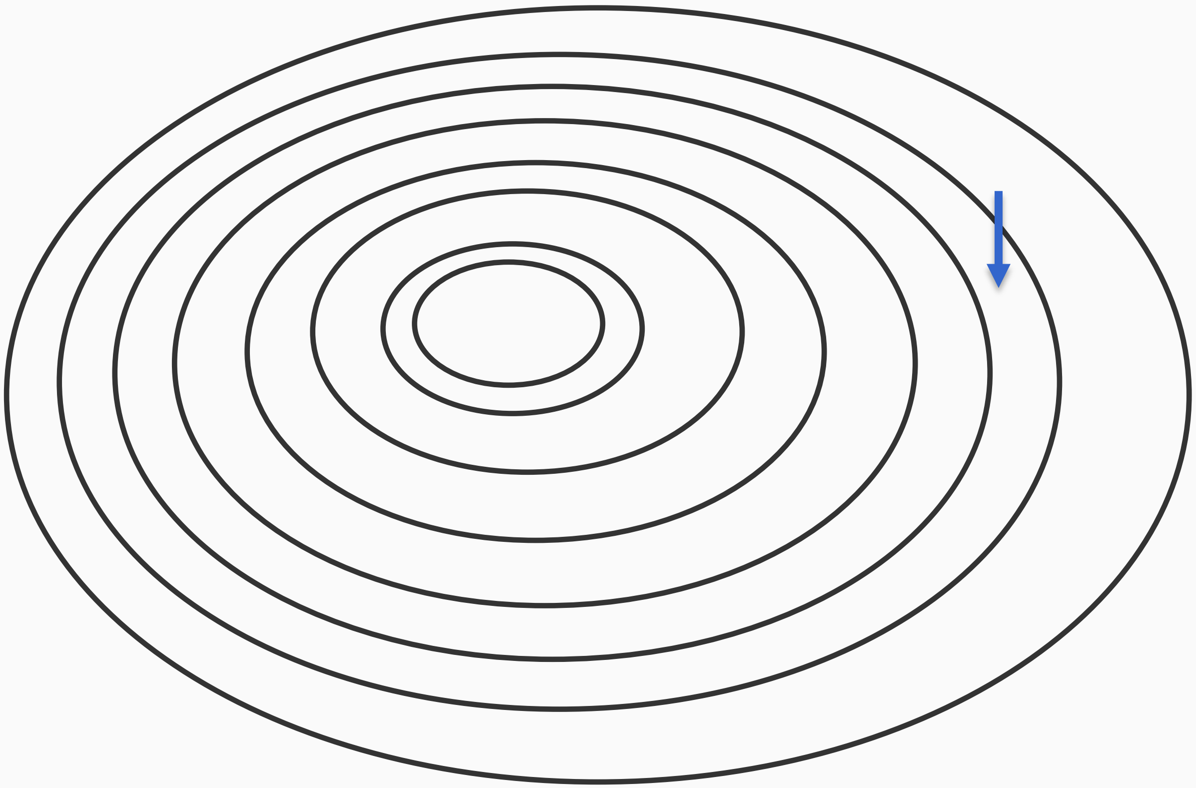
Gradient descent

Gradient Descent vs SGD



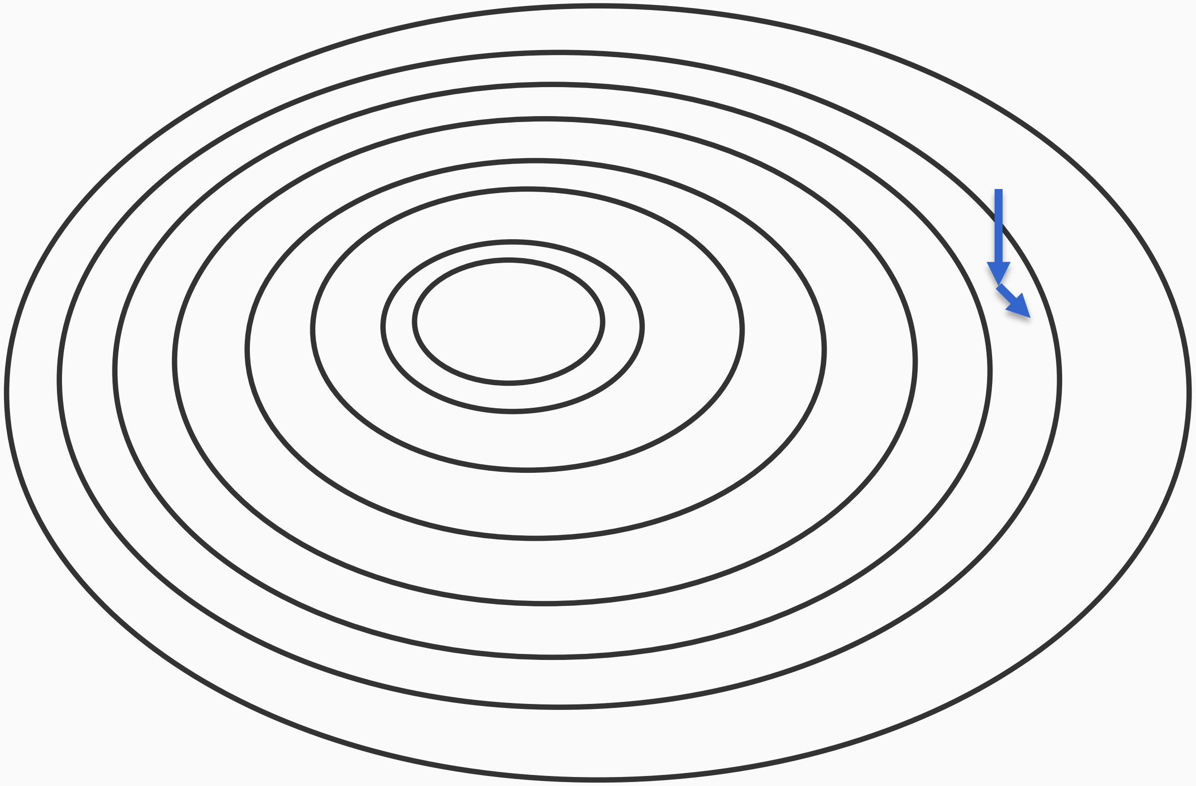
Stochastic Gradient descent

Gradient Descent vs SGD



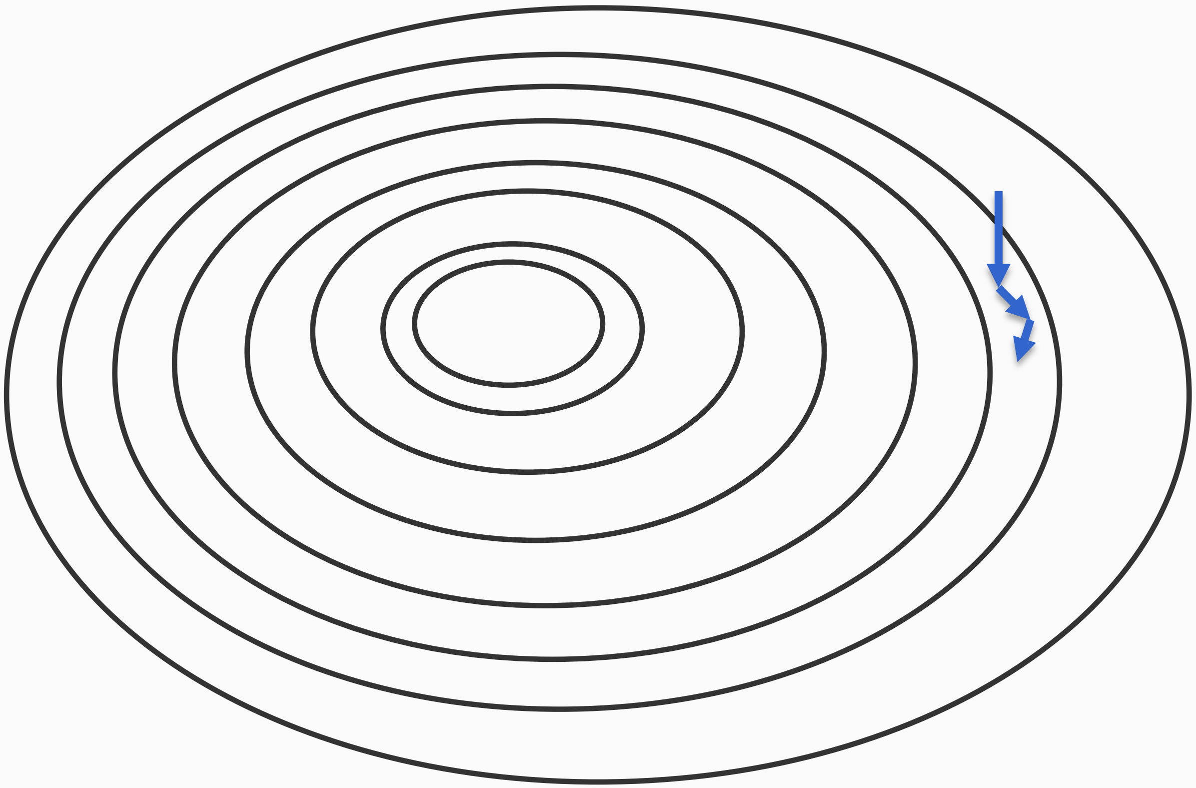
Stochastic Gradient descent

Gradient Descent vs SGD



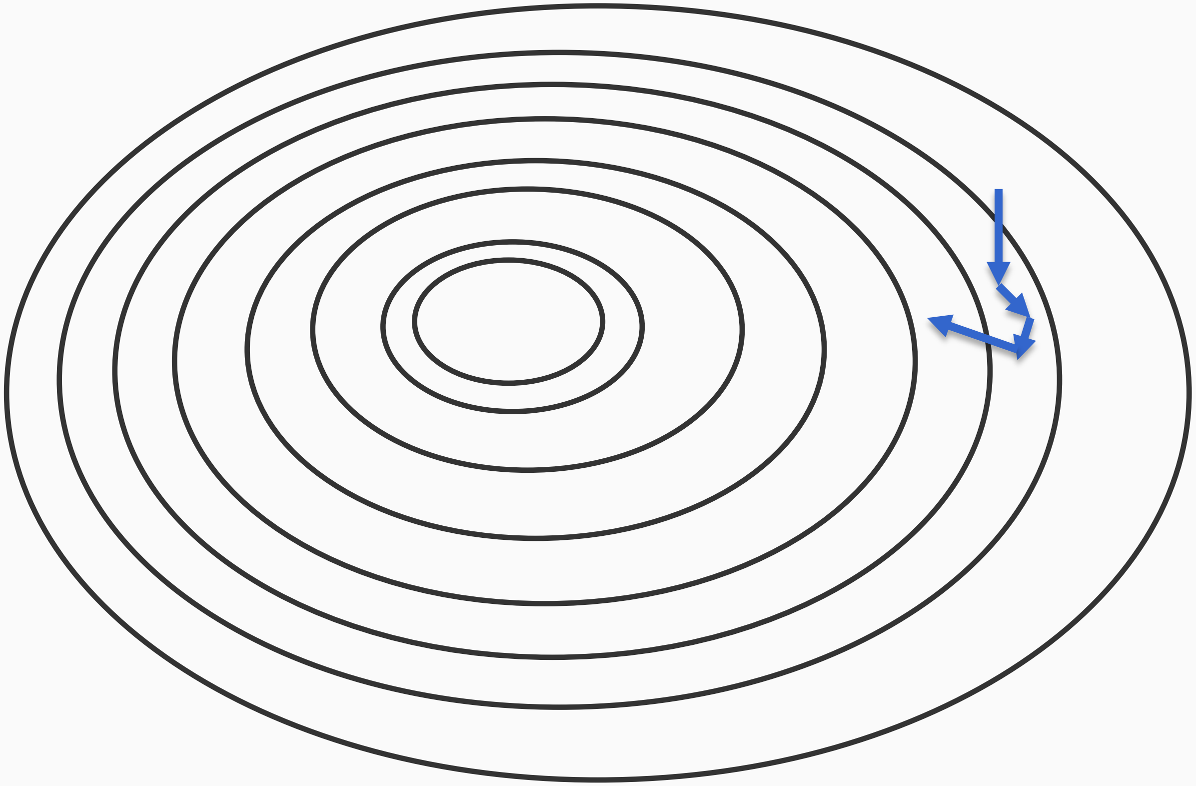
Stochastic Gradient descent

Gradient Descent vs SGD



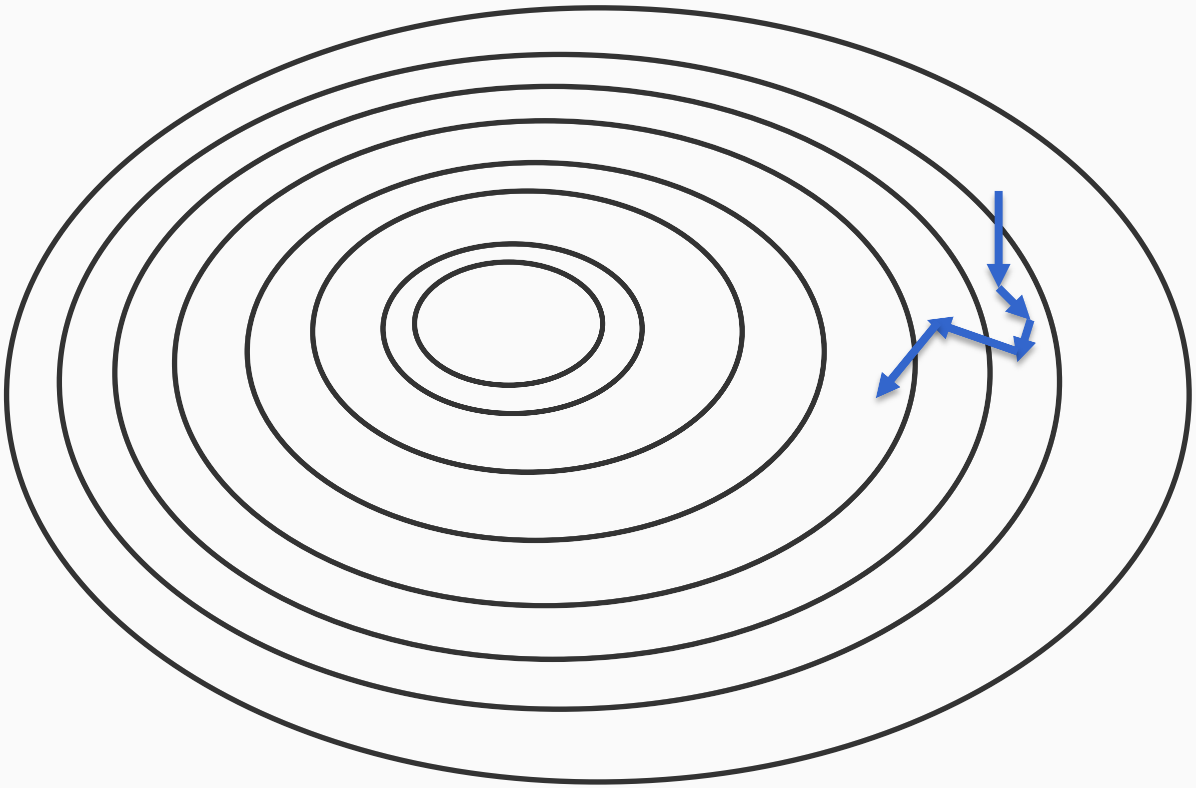
Stochastic Gradient descent

Gradient Descent vs SGD



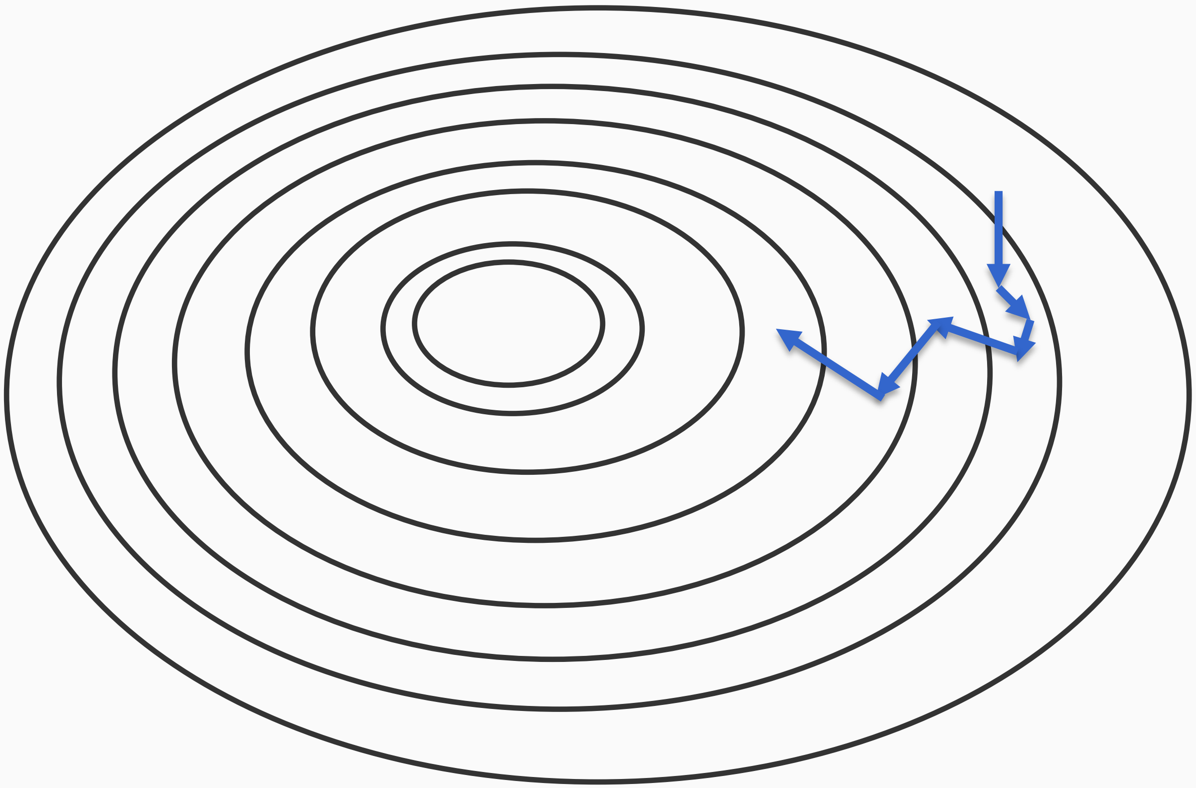
Stochastic Gradient descent

Gradient Descent vs SGD



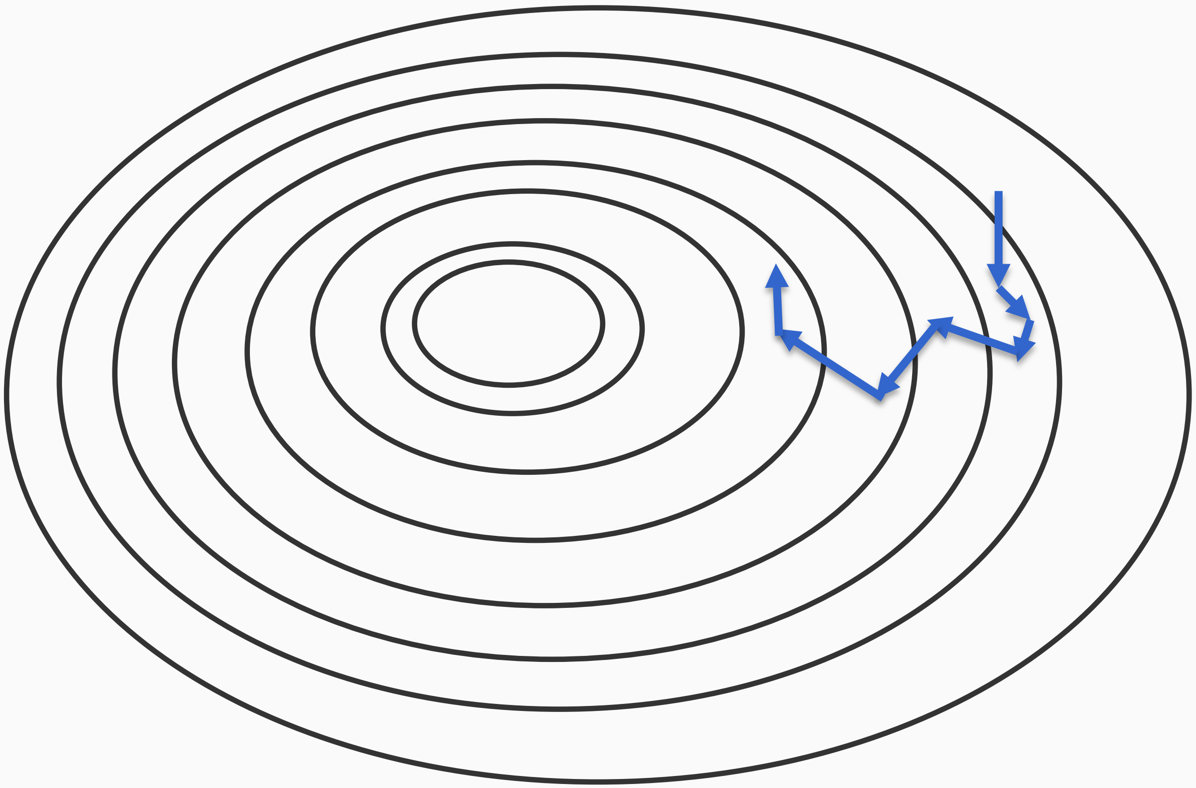
Stochastic Gradient descent

Gradient Descent vs SGD



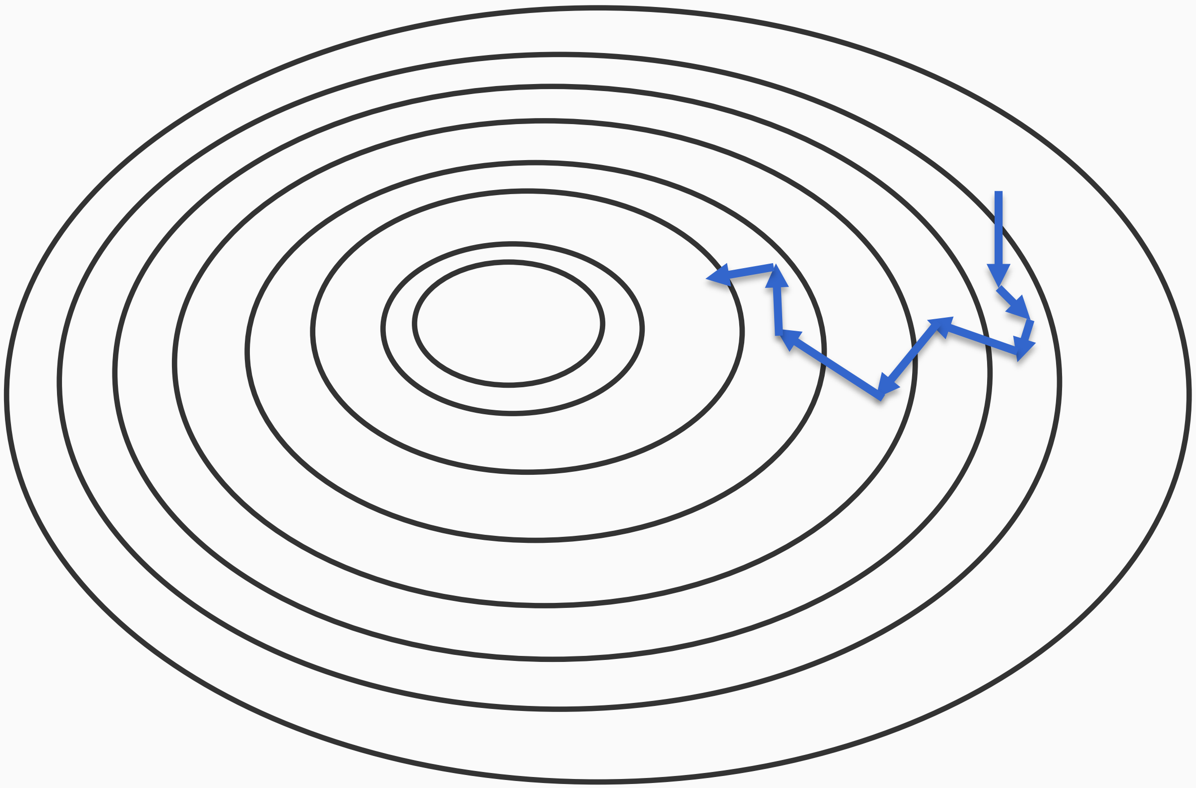
Stochastic Gradient descent

Gradient Descent vs SGD



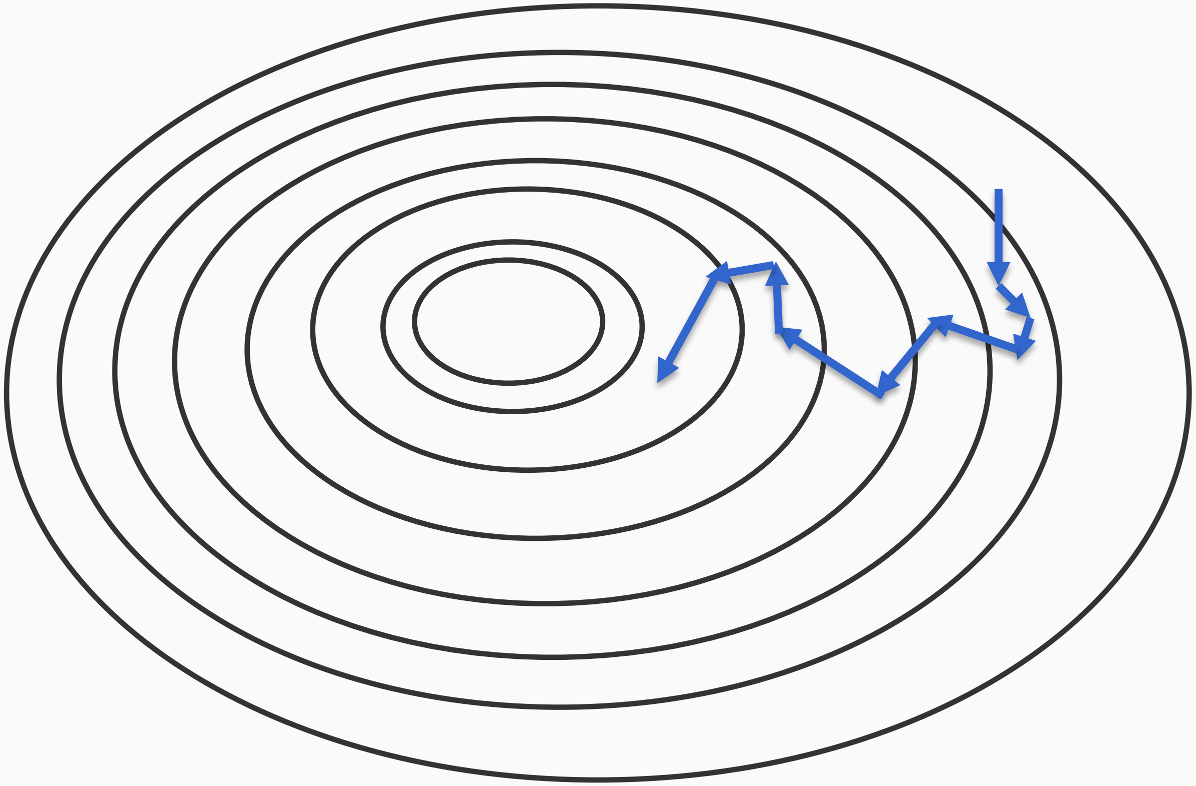
Stochastic Gradient descent

Gradient Descent vs SGD



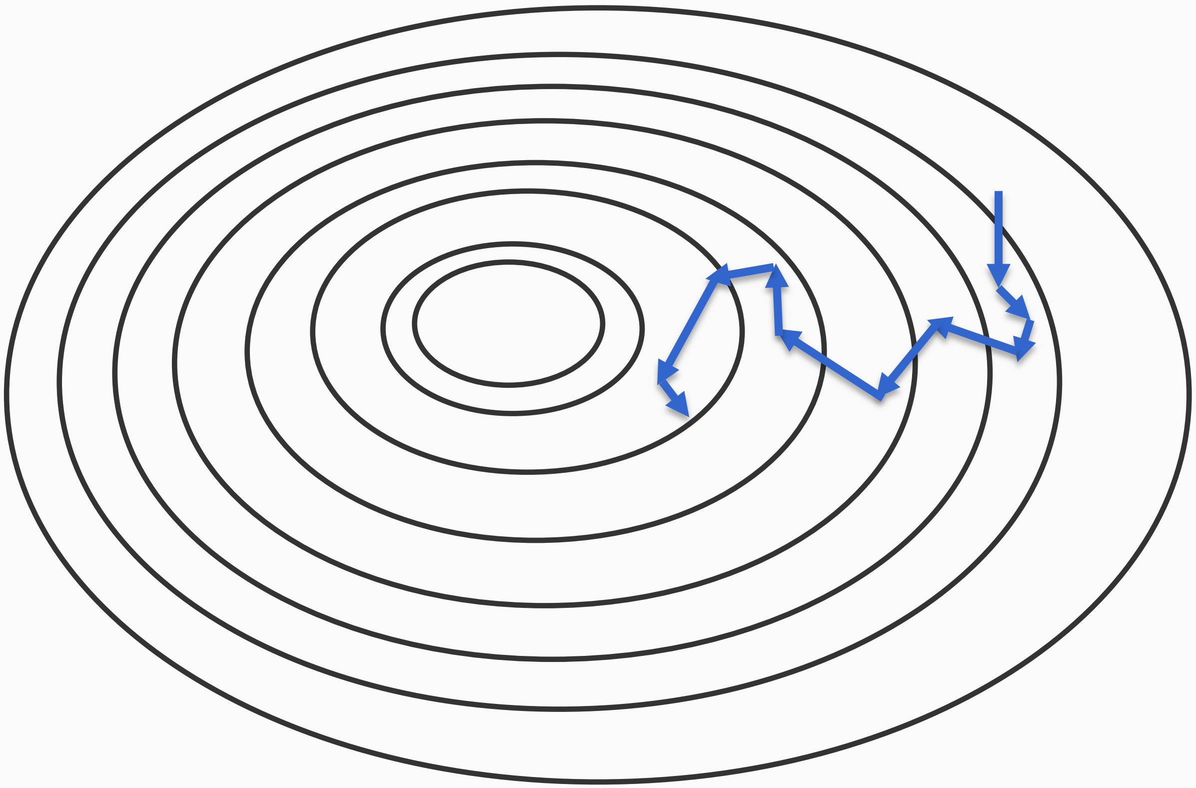
Stochastic Gradient descent

Gradient Descent vs SGD



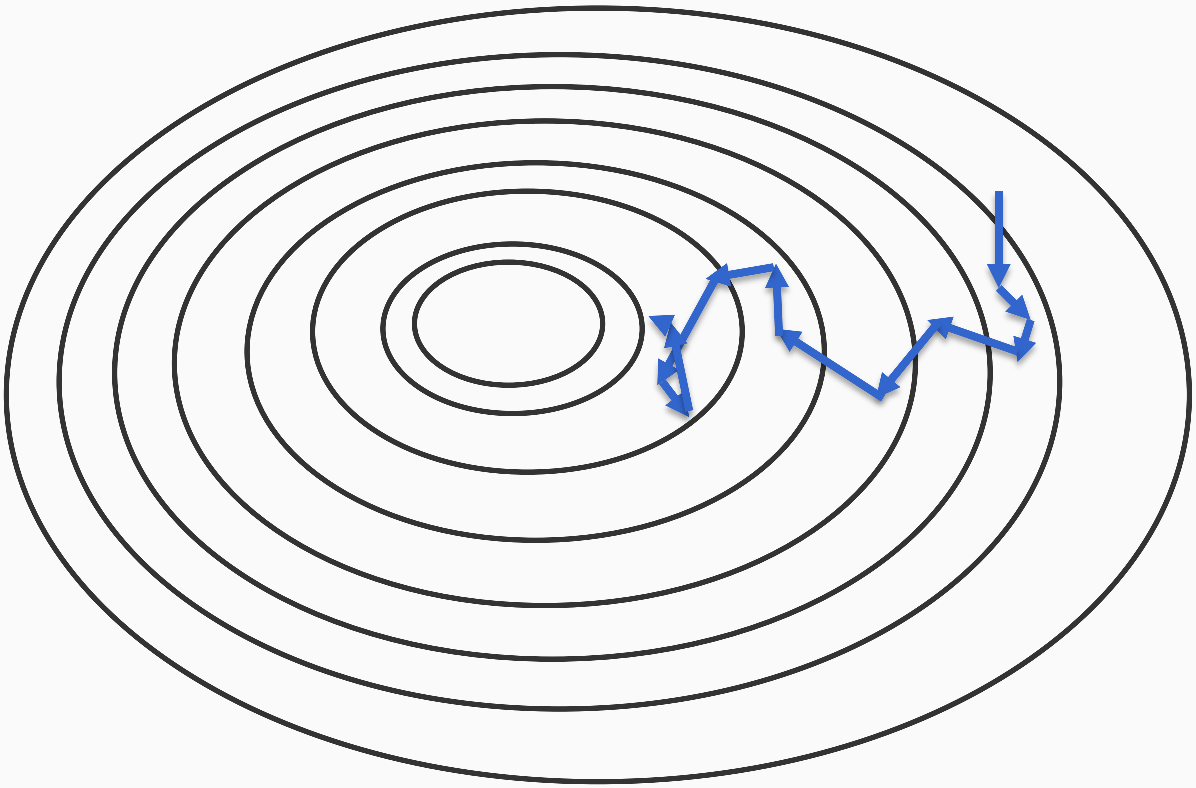
Stochastic Gradient descent

Gradient Descent vs SGD



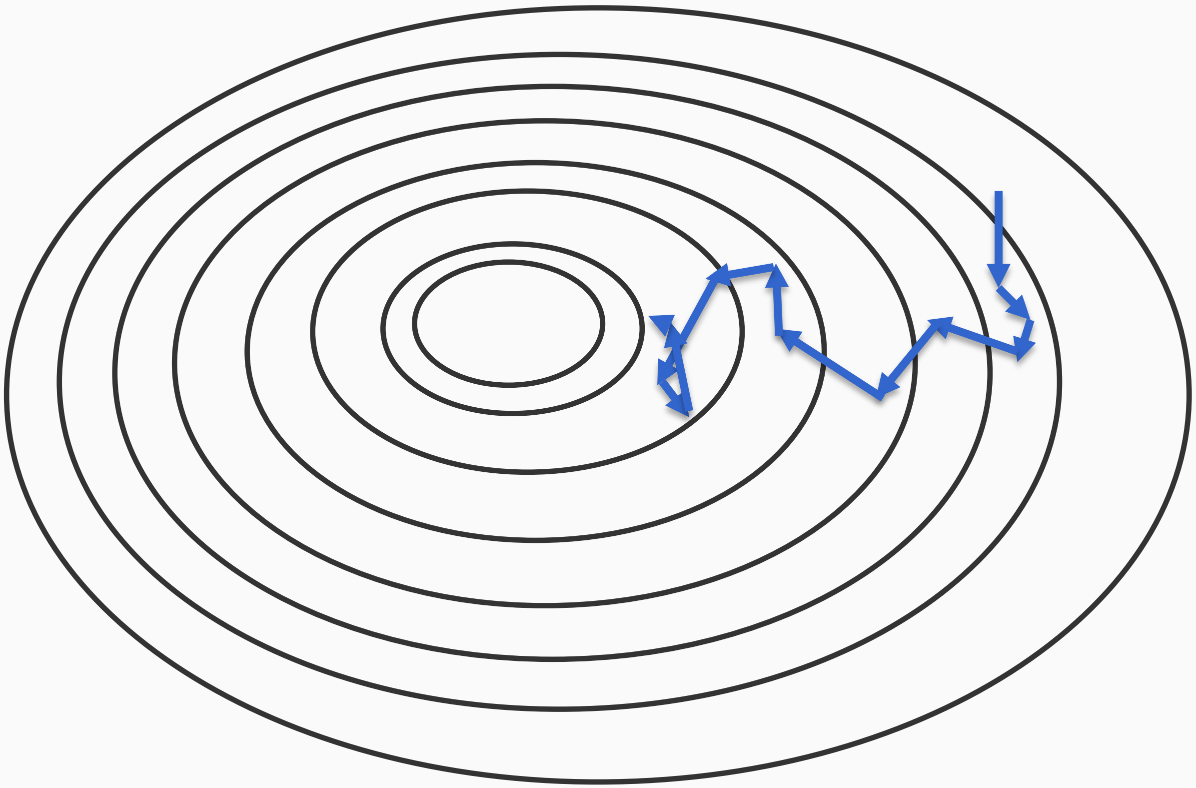
Stochastic Gradient descent

Gradient Descent vs SGD



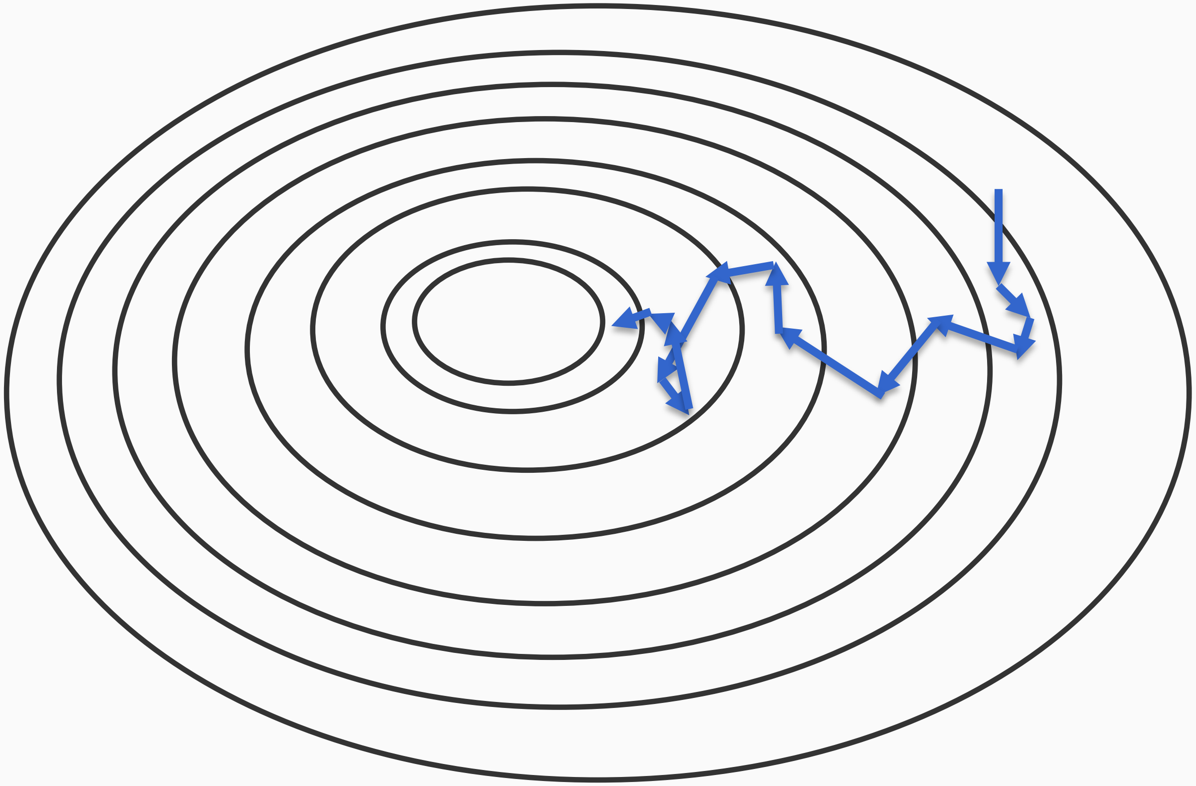
Stochastic Gradient descent

Gradient Descent vs SGD



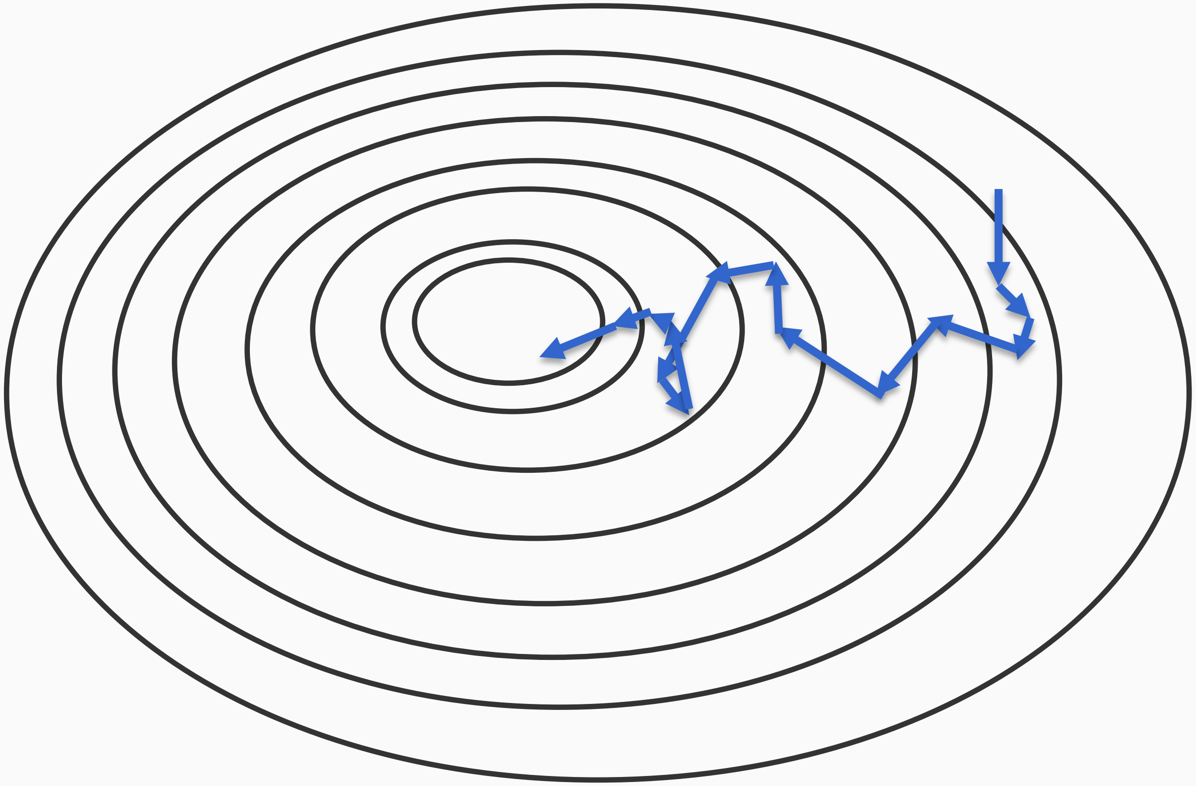
Stochastic Gradient descent

Gradient Descent vs SGD



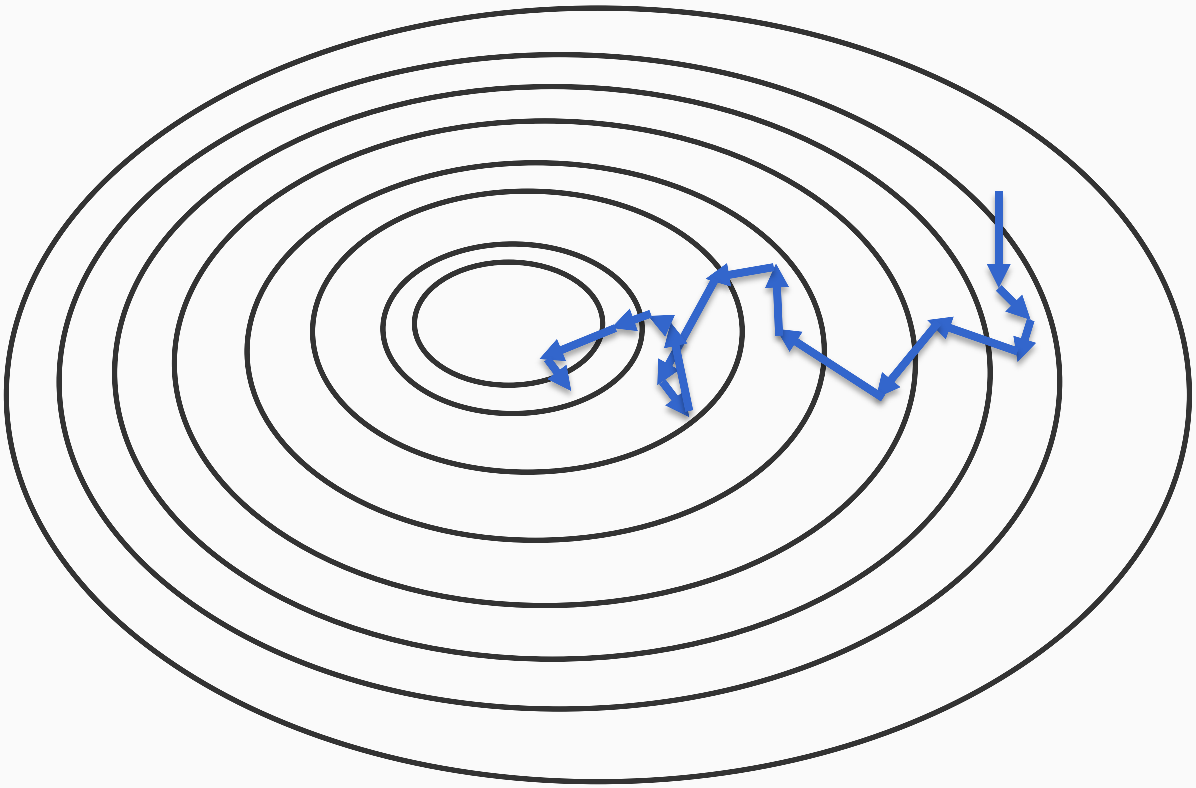
Stochastic Gradient descent

Gradient Descent vs SGD



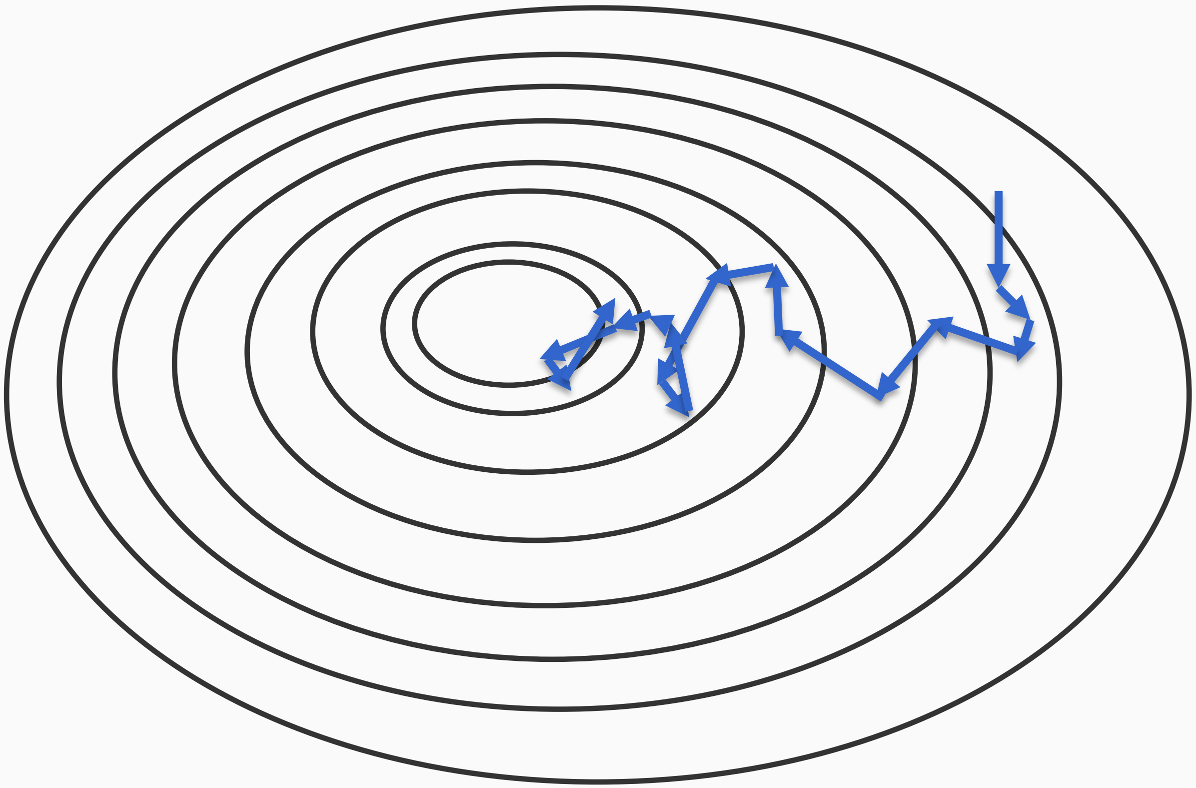
Stochastic Gradient descent

Gradient Descent vs SGD



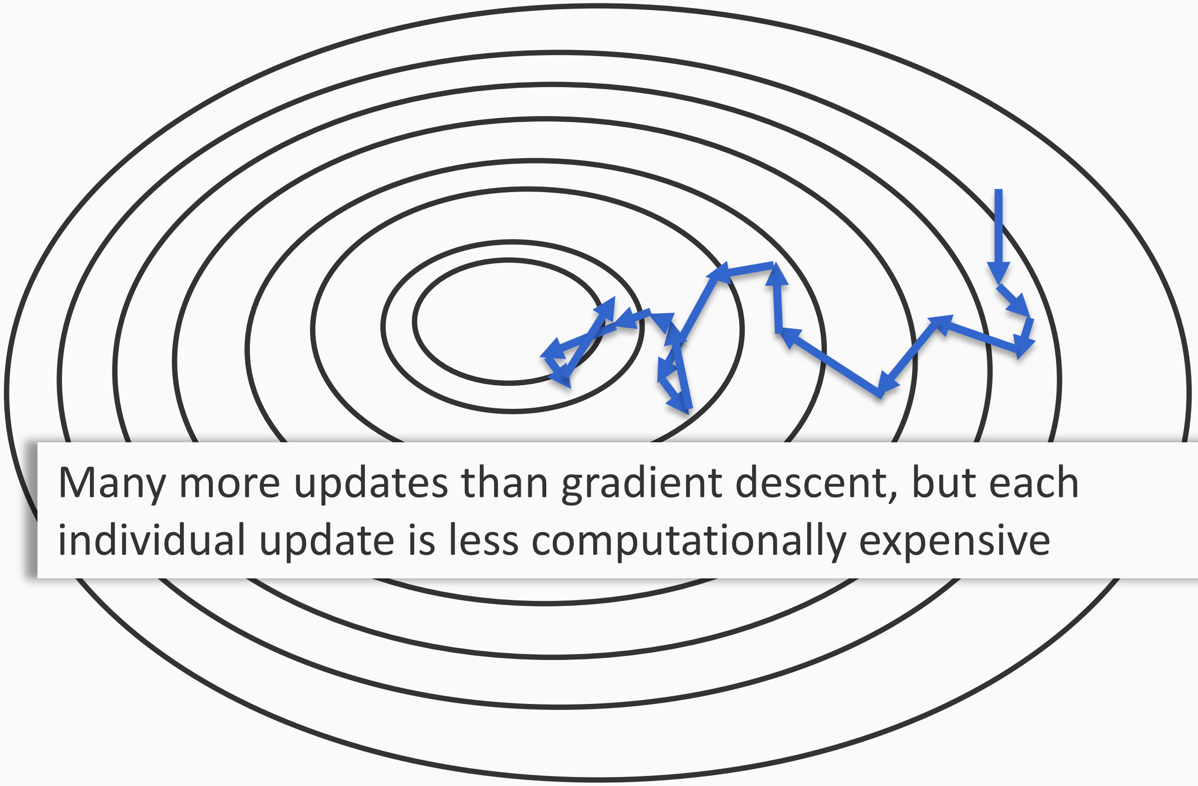
Stochastic Gradient descent

Gradient Descent vs SGD



Stochastic Gradient descent

Gradient Descent vs SGD



Many more updates than gradient descent, but each individual update is less computationally expensive

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

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C\sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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 **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!

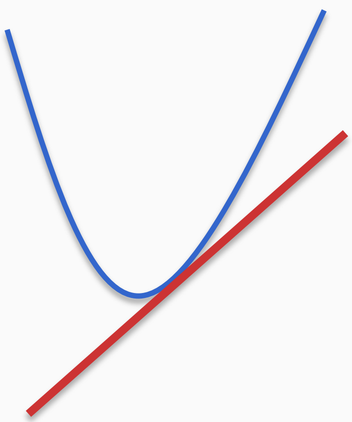
What is the derivative of the hinge loss with respect to w?

$$J^t(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \cdot N \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

Detour: Sub-gradients

Generalization of gradients to non-differentiable functions

(Remember that every tangent lies below the function for convex functions)



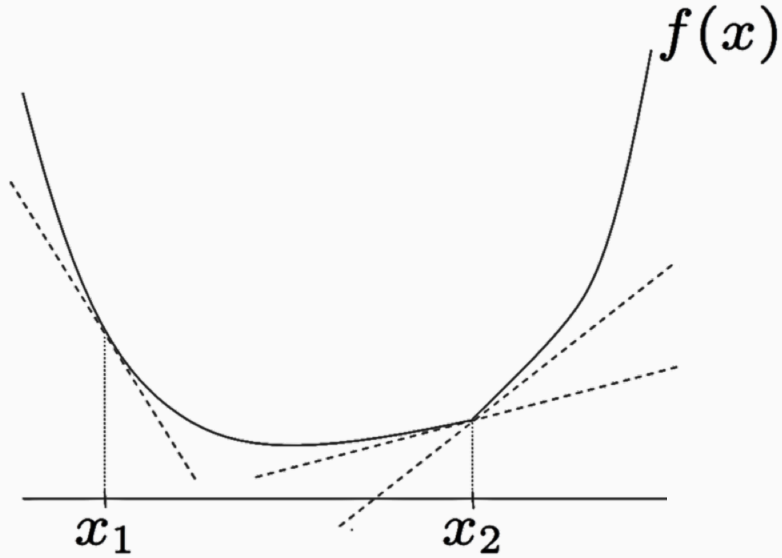
Informally, a sub-tangent at a point is any line lies below the function at the point.

A sub-gradient is the slope of that line

Sub-gradients

Formally, g is a subgradient to f at x if

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



[Example from Boyd]

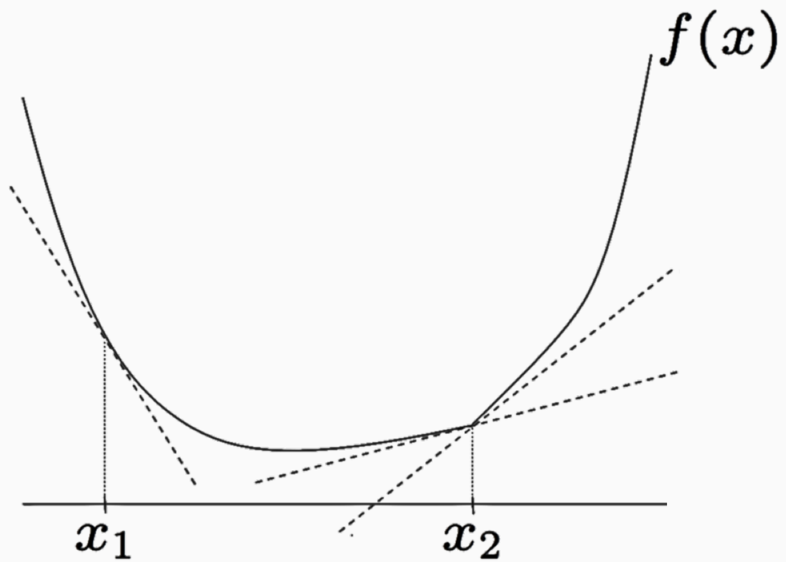
Sub-gradients

Formally, g is a subgradient to f at 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]

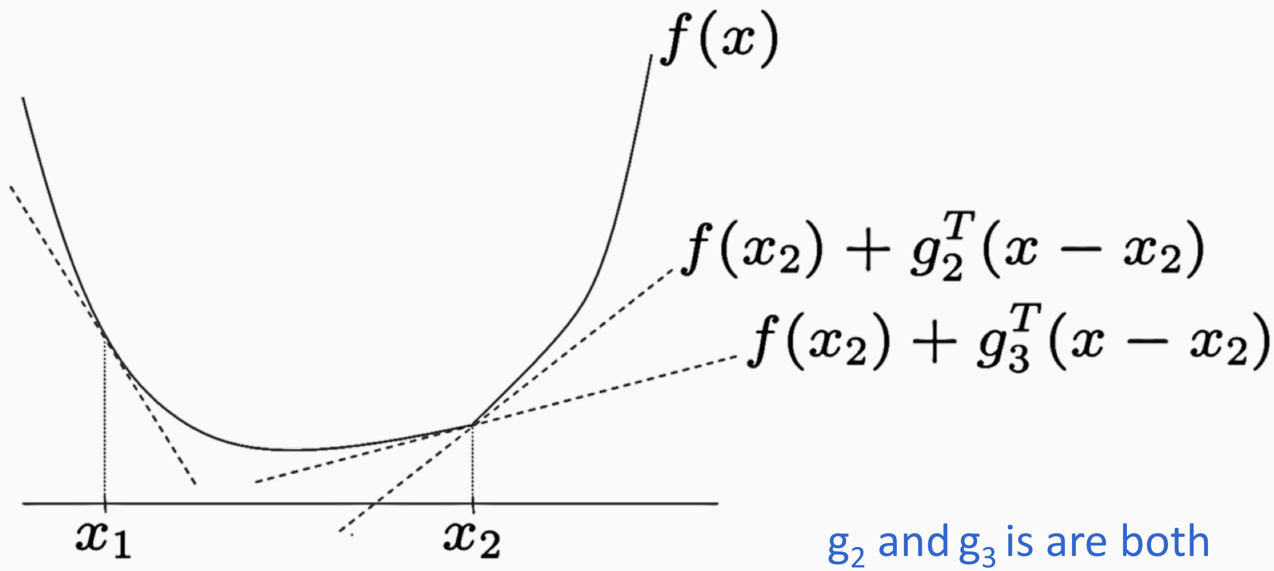
Sub-gradients

Formally, g is a subgradient to f at 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

$$J^t(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \cdot N \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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

Sub-gradient of the SVM objective

$$J^t(\mathbf{w}) = \frac{1}{2}\mathbf{w}_0^\top \mathbf{w}_0 + C \cdot N \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

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

$$\nabla J^t = \begin{cases} [\mathbf{w}_0; 0] & \text{if } \max(0, 1 - y_i \mathbf{w}_i^{\mathbf{x}}) = 0 \\ [\mathbf{w}_0; 0] - C \cdot N y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

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

$$\nabla J^t = \begin{cases} [\mathbf{w}_0; 0] & \text{if } \max(0, 1 - y_i \mathbf{w}_i^{\mathbf{x}}) = 0 \\ [\mathbf{w}_0; 0] - C \cdot N y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

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

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

3. Return **w**

Stochastic sub-gradient descent for SVM

$$\nabla J^t = \begin{cases} [\mathbf{w}_0; 0] & \text{if } \max(0, 1 - y_i \mathbf{w}_i^{\mathbf{x}}) = 0 \\ [\mathbf{w}_0; 0] - C \cdot N y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

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

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

3. Return **w**

Stochastic sub-gradient descent for SVM

$$\nabla J^t = \begin{cases} [\mathbf{w}_0; 0] & \text{if } \max(0, 1 - y_i \mathbf{w}_i^{\mathbf{x}}) = 0 \\ [\mathbf{w}_0; 0] - C \cdot N y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

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

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

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

3. Return **w**

Stochastic sub-gradient descent for SVM

$$\nabla J^t = \begin{cases} [\mathbf{w}_0; 0] & \text{if } \max(0, 1 - y_i \mathbf{w}_i^{\mathbf{x}}) = 0 \\ [\mathbf{w}_0; 0] - C \cdot N y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

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

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

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

      $\mathbf{w} \leftarrow (1-\gamma_t)\, [\mathbf{w}_0; \mathbf{0}] + \gamma_t\, C\, N\, y_i\, \mathbf{x}_i$

      else

      $\mathbf{w}_0 \leftarrow (1-\gamma_t)\, \mathbf{w}_0$
3. Return **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 $\mathbf{w}^0 = 0 \in \Re^n$
2. For epoch = 1 … T:
    1. For each training example $(\mathbf{x}_i, y_i) \in S$:

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

       $\mathbf{w} \leftarrow (1-\gamma_t)\,[\mathbf{w}_0; \mathbf{0}] + \gamma_t\, C\, N\, y_i\, \mathbf{x}_i$

       else

       $\mathbf{w}_0 \leftarrow (1-\gamma_t)\,\mathbf{w}_0$
3. Return $\mathbf{w}$

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

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

       $\mathbf{w} \leftarrow (1-\gamma_t)\, [\mathbf{w}_0; \mathbf{0}] + \gamma_t\, C\, N\, y_i\, \mathbf{x}_i$

       else

       $\mathbf{w}_0 \leftarrow (1-\gamma_t)\, \mathbf{w}_0$
3. Return $\mathbf{w}$

γ_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 γ_t are positive
- Sum of squares of step sizes over t = 1 to ∞ is not infinite
- Sum of step sizes over t = 1 to ∞ 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 ϵ

- 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

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

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

       $\mathbf{w} \leftarrow (1-\gamma_t)\, [\mathbf{w}_0; \mathbf{0}] + \gamma_t\, C\, N\, y_i\, \mathbf{x}_i$

       else

       $\mathbf{w}_0 \leftarrow (1-\gamma_t)\, \mathbf{w}_0$
3. Return $\mathbf{w}$

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

Perceptron vs. SVM

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

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

- SVM optimizes the hinge loss
  - With regularization

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

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: similar framework with different objectives!

  - Compare with Perceptron algorithm: Perceptron does not maximize margin width
    - Perceptron variants can force a margin
- Other successful optimization algorithms exist
  - Eg: Dual coordinate descent, implemented in `liblinear`

Questions?