Lecture 3 - Maximum Margin Classifiers and Support Vector Machines

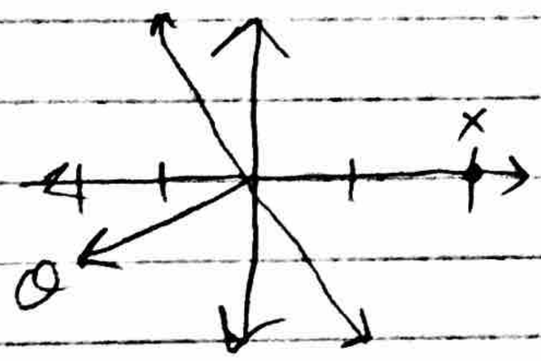
Review:

(1) Let $\theta = \begin{pmatrix} -2 \\ -1 \end{pmatrix}$, $\theta_0 = 0$.

a) Draw θ.

b) Draw the decision boundary.

c) Let $x = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$. What is $h(x;\theta)$? (You can either compute it or determine it by looking at your graph.)



$h(x;\theta) = \text{sign}(\theta \cdot x) = \text{sign}\left(\begin{pmatrix} -2 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 0 \end{pmatrix}\right)$

$= \text{sign}(-2 \cdot 2 + (-1) \cdot 0) = \text{sign}(-4) = -1$

$h(x;\theta) = -1$

(2) Let $\theta = \begin{pmatrix} 1 \\ 3 \\ -2 \end{pmatrix}$, $\theta_0 = -1$, $x = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$, $y = -1$.

a) What is the agreement?

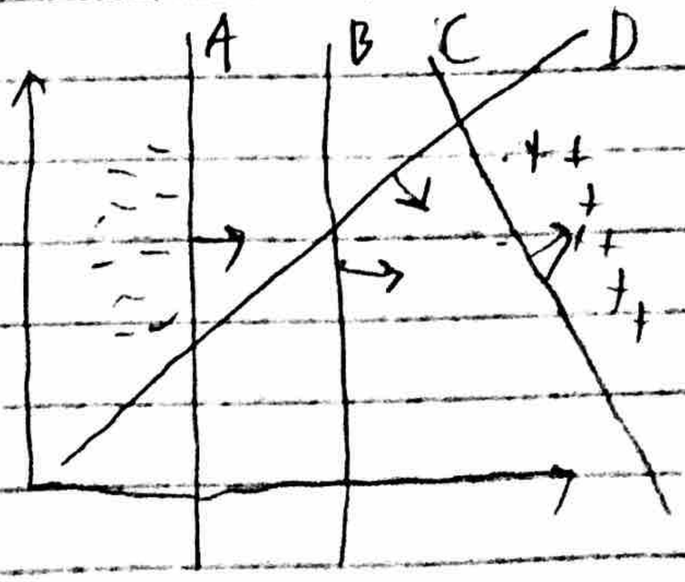
$$\text{agreement} = y(\theta \cdot x + \theta_0) = -1\left[\begin{pmatrix} 1 \\ 3 \\ -2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} - 1\right]$$

$$= -\left[1 + 3 - 4 \quad -1\right] = -1 \cdot (-1) = 1$$

b) Is x classified correctly?

Yes - agreement > 0

(3) Which is the best linear classifier for the following set of data?



Answer depends on "best". All correctly classify all the points so all are perfect wrt training points. But B is best in terms of ability to generalize. What is special about B? Its margin.

3-2

Today
- Margins
  - Motivation
  - Definition
  - Examples
- Support vector machines
  - Objective function
    - Hinge loss
    - Regularization
  - Offline vs. online algorithms
  - Online SVM
    - Gradient descent
    - Pegasos algorithm

Margins

Motivation: Perceptron has many solutions, which is best?

Goal: Generalize to the test set.

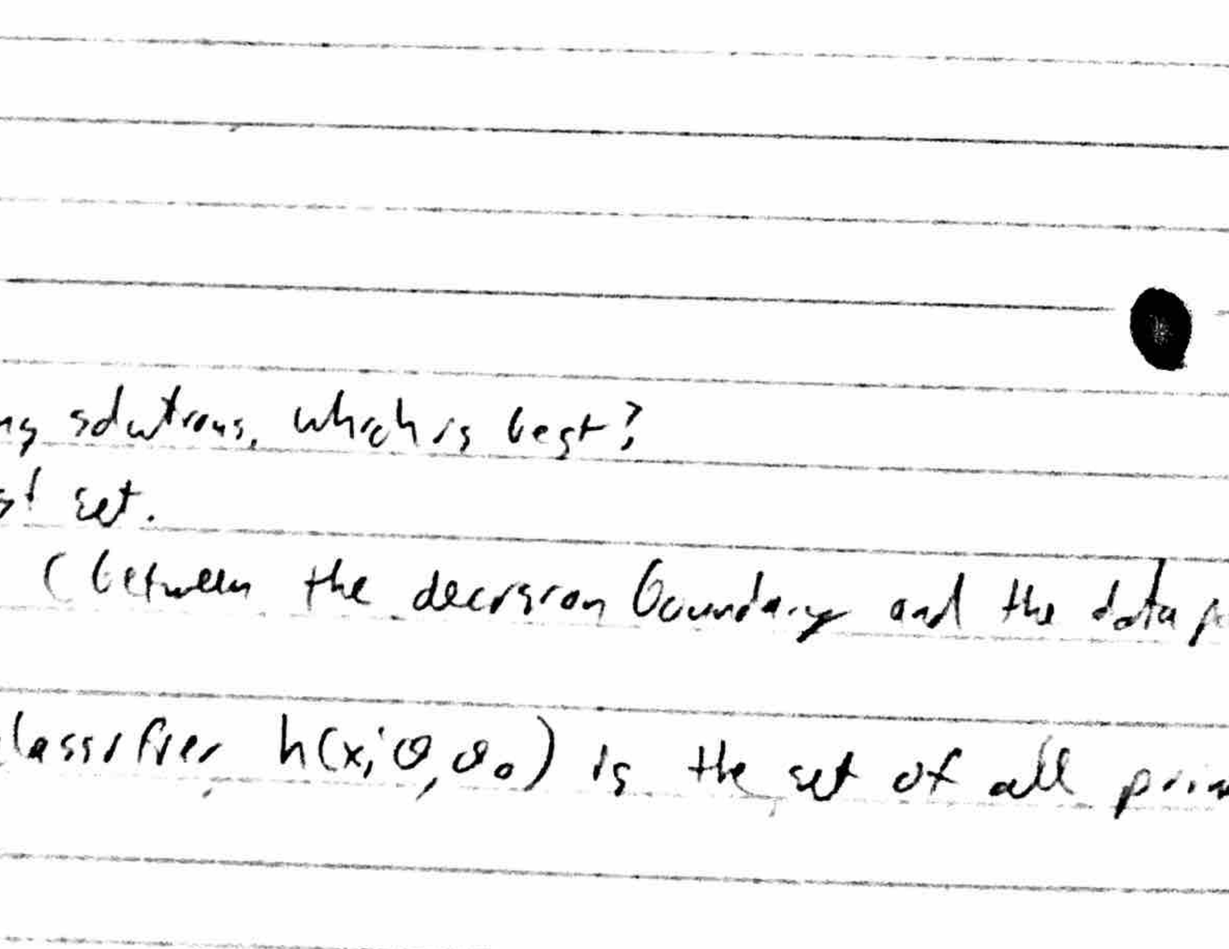
How? Maximize the margin (between the decision boundary and the data points)

Def: The margin of a linear classifier $h(x; \theta, \theta_0)$ is the set of all points where $\theta \cdot x + \theta_0 = \pm 1$.

the margin is [a pair of lines in $\mathbb{R}^2$ / a pair of hyperplanes in $\mathbb{R}^d$] parallel to the decision boundary.

Example

$\theta = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ $\theta_0 = 1$



(1) Draw decision boundary.

(2) Draw margins.

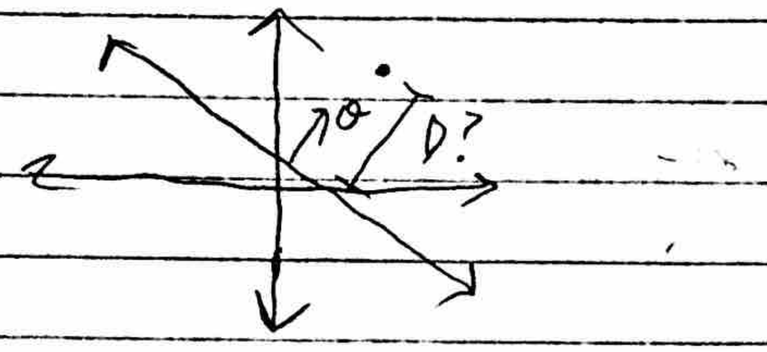
$\theta \cdot x + \theta_0 = 0 \rightarrow x_1 + x_2 + 1 = 0 \rightarrow x_2 = -x_1 - 1$

$\theta \cdot x + \theta_0 = 1 \rightarrow x_1 + x_2 + 1 = 1 \rightarrow x_2 = -x_1$

$\theta \cdot x + \theta_0 = -1 \rightarrow x_1 + x_2 + 1 = -1 \rightarrow x_2 = -x_1 - 2$

Distance to the margin

(?) Distance from a line to a point



$$D = \frac{|\theta \cdot x + \theta_0|}{\|\theta\|}$$

Reminder: $\|\theta\| = \sqrt{\theta_1^2 + \theta_2^2 + \ldots + \theta_d^2}$

= magnitude of the vector θ

= distance from the origin to the point θ

Distance to the margin

$$D = \frac{|\theta \cdot x + \theta_0|}{\|\theta\|} = \frac{|\pm 1|}{\|\theta\|} = \frac{1}{\|\theta\|}$$

$$D = \frac{1}{\|\theta\|}$$

Note: Distance to the margin is inversely proportional to the magnitude of θ.

Large θ $\longleftrightarrow$ small margin

Small θ $\longleftrightarrow$ large margin

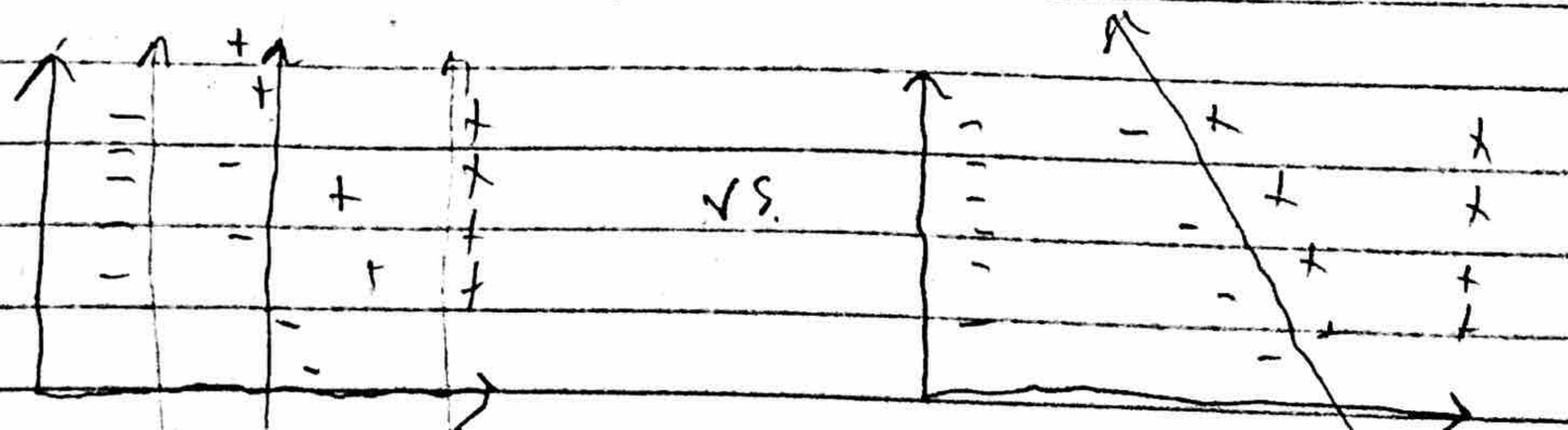
(*) The direction of θ does not affect the distance to the margin, only the magnitude matters.

Takeaway: Better generalization $\longleftrightarrow$ large margin $\longleftrightarrow$ small θ

(?) Question: What happens if we care too much about maximizing the margin?

Answer: Potential loss of accuracy.

Example:



(?) Question: How can we balance the competing interests of large margin and high accuracy?

3-4 Answer: SVM

Support Vector Machines (Maximum margin linear classifier)

Objective: High accuracy and large margin ⇕

Low loss and small θ

Objective function: $\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y^{(i)} \theta \cdot x^{(i)}) + \frac{\lambda}{2} \|\theta\|^2$

Taking it apart:

1) $\min_{\theta}$ → Find the θ which minimizes the expression.

2) $\frac{1}{n} \sum_{i=1}^{n}$ → Averages the loss across all training examples.

3) $\text{Loss}(y^{(i)} \theta \cdot x^{(i)})$ → The "loss" of a single training example. (will explain shortly.)

4) $\|\theta\|^2$ → The magnitude of θ (squared). Since we're doing a min, this will give us small $\|\theta\|$ so large margin.

This is called a regularization term because it regulates (controls) the complexity of the model and helps prevent overfitting and encourages generalization.

5) $\frac{\lambda}{2}$ → λ is a hyperparameter which allows us to control which is more important: accuracy or margin.
($\frac{1}{2}$ is arbitrary, makes math easier.)
Large λ → $\|\theta\|$ dominates and small θ/large margin is most important
small λ → Loss dominates and high accuracy is most important

(?) (As in lab2, we can test different values of λ to see what performs best.)

(Check if all parts make sense.)

(So now that we understand how SVM makes its tradeoffs, what is Loss?)

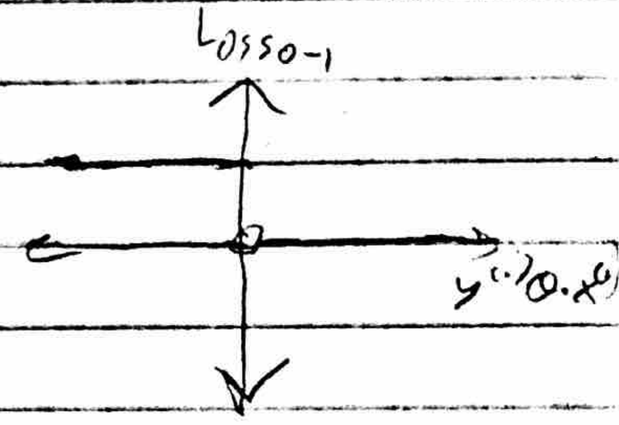
Loss functions

Loss functions are a generalization of accuracy (like agreement).
Loss functions are a measure of how far away we are from our objective.
Smaller loss is better, goal is to minimize loss.
Loss is defined for individual examples. Total loss is sum of loss for each example.

0-1 loss

$$\text{Loss}_{0-1}(y^{(i)}\theta\cdot x^{(i)}) = \begin{cases} 1 & y^{(i)}\theta\cdot x^{(i)} \le 0 \\ 0 & y^{(i)}\theta\cdot x^{(i)} > 0 \end{cases}$$

(?) Graph loss



0-1 loss is essentially the inverse of accuracy.
Correct = 0
Incorrect = 1
Zero loss = 100% accuracy

0-1 loss is limited because it doesn't say how close we are to being correct; it only says if we are correct or not (on the right side of the decision boundary or not).

(?) How could we improve the 0-1 loss? Could we include distance information?

Hinge Loss

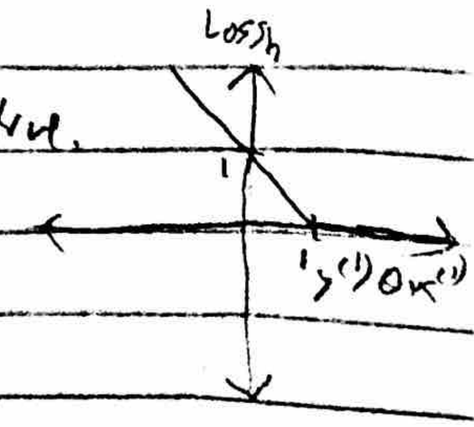
Idea: Make the loss proportional to the distance from a correct classification.

(?) Hinge loss

$$\text{Loss}_h(y^{(i)}\theta\cdot x^{(i)}) = \begin{cases} 1 - y^{(i)}\theta\cdot x^{(i)} & y^{(i)}\theta\cdot x^{(i)} \le 1 \\ 0 & y^{(i)}\theta\cdot x^{(i)} \ge 1 \end{cases}$$

$$\text{Loss}_h(y^{(i)}\theta\cdot x^{(i)}) = \max(0, 1 - y^{(i)}\theta\cdot x^{(i)})$$

(?) Why 1 instead of 0?
We're using margins rather than the decision boundary.
Enforces stronger separation between positive and negative.



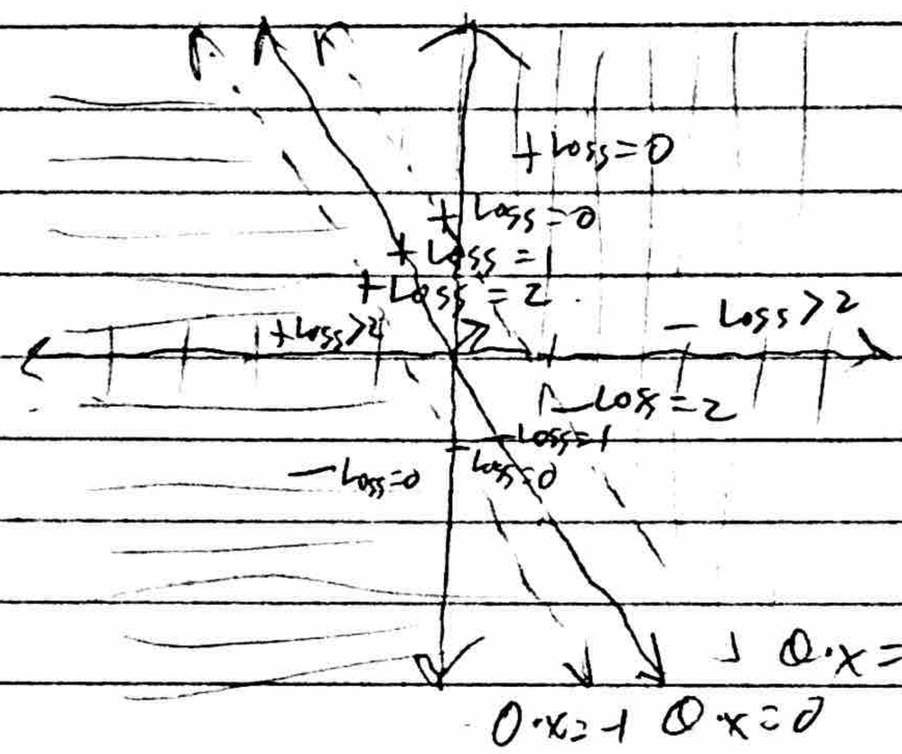
3-6

Example

(?) What is distance to margin?

$$\theta = \begin{pmatrix} 1 \\ 1/2 \end{pmatrix} \qquad D = \frac{1}{\|\theta\|} = \frac{1}{\sqrt{1^2 + (\frac{1}{2})^2}} = \frac{1}{\sqrt{1 + \frac{1}{4}}} = \frac{1}{\sqrt{\frac{5}{4}}} = \frac{2}{\sqrt{5}} \approx 0.9$$

(?) Draw + and − and ask for loss. Show of hands.



□ 0 loss for + point

▤ 0 loss for − point

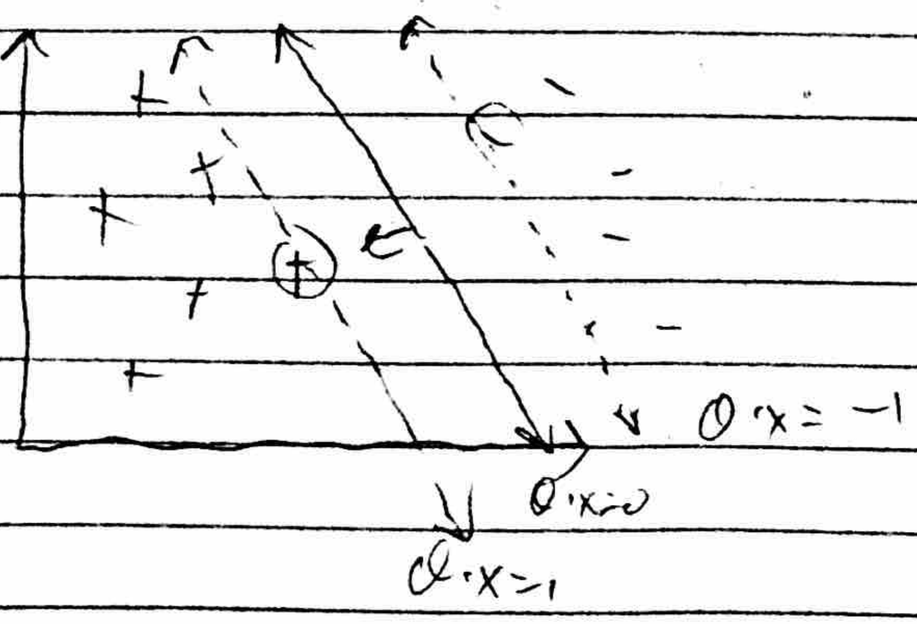
Loss increases linearly from 0 as the point moves away from the margin

(?) Realizable

Realizable Case

When the points are linearly separable (realizable), there is a unique solution where hinge loss is 0 and margin is maximized.

(?) Ask someone to draw solution.

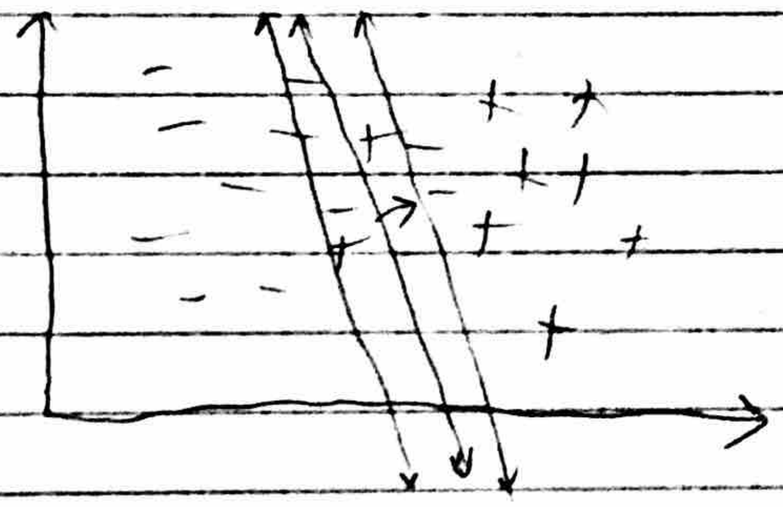


- 0 hinge loss because all points are beyond margin
- Max margin because decision boundary is in the middle between closest positive and negative

(?) What happens if I remove a non-support vector?
What happens if I remove a support vector?

Note: In the realizable case, the margin is determined entirely by the closest + and − which end up on the margin.
These special points are called <u>support vectors</u>. Thus the name SVM.

<u>Unrealizable case</u>
Points can't be perfectly separated so simply balance loss and margin.



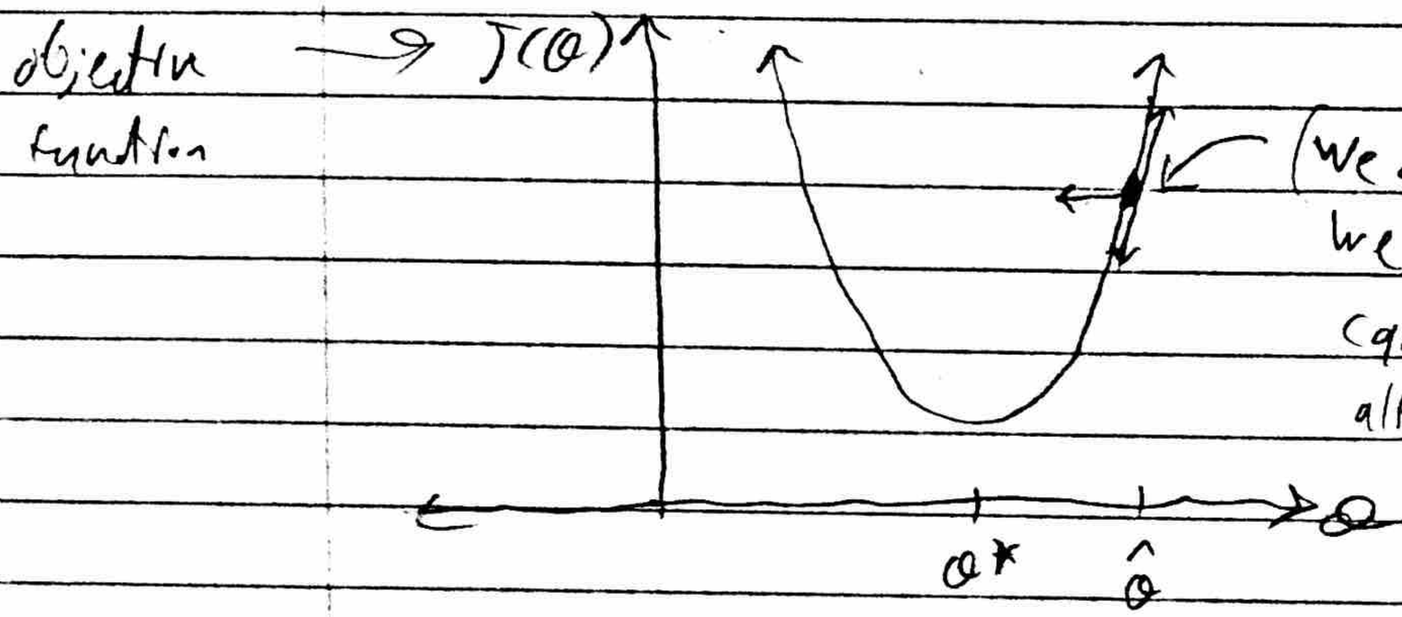
<u>Solving SVM</u>
Two methods:
1) <u>offline</u> - Directly optimize the objective function on all the points at once.
<u>Pro</u>: Achieves optimal solution
<u>Con</u>: Hard (involves Lagrangians)
2) <u>online</u> - Optimize the objective function point by point.
<u>Pro</u>: Easier
<u>Con</u>: No guarantee of optimal solution

(Today we'll do online SVM because it's easier to understand, it still performs very well, and it introduces concepts which will be important later in the course.)

<u>Online SVM</u>
Idea: Optimize objective function for each point one at a time.

(?) How? If we choose α just for one point, it's probably going to work very poorly for other points. How can we slightly adjust α w/o totally overwriting it?

(?) We're trying to find the minimum of the objective function, which is a function of θ, but we don't know where the minimum is (i.e. what θ produces minimum cost). Do we know anything about what direction the minimum is in?



(We are at $\hat{\theta}$ and want to get to θ^*. We don't know where θ^* is, but we can find the direction of $J(\hat{\theta})$ and alter θ in the direction in which $J(\theta)$ decreases.)

Idea: The gradient (generalization of derivative) tells us the direction in which changing θ most increases $J(\theta)$. We will go in the opposite direction.

Gradient Descent

$$\theta_{new} = \theta_{old} - \eta \nabla_\theta J(\theta)$$

$\nabla_\theta J(\theta)$ → gradient of $J(\theta)$ with respect to θ

η → learning rate - controls how much we change θ in the direction specified by the gradient

$-$ → travel in the opposite direction of the gradient

Gradient Review

The gradient is a vector of derivatives.

If $\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{pmatrix}$ then $\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} J(\theta) \\ \frac{\partial}{\partial \theta_2} J(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_d} J(\theta) \end{pmatrix}$

Example

$J(\theta) = \theta \cdot x$ where $\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$

$J(\theta) = \theta_1 x_1 + \theta_2 x_2$

(!) $\frac{\partial}{\partial \theta_1} J(\theta) = \frac{\partial}{\partial \theta_1}(\theta_1 x_1 + \theta_2 x_2) = 1 \cdot x_1 + 0 \cdot x_2 = x_1$

(!) $\frac{\partial}{\partial \theta_2} J(\theta) = \frac{\partial}{\partial \theta_2}(\theta_1 x_1 + \theta_2 x_2) = 0 \cdot x_1 + 1 \cdot x_2 = x_2$

$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} J(\theta) \\ \frac{\partial}{\partial \theta_2} J(\theta) \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

Gradient of SVM Objective Function

(ignore $\frac{1}{n}$) For a single point $x^{(i)}$, $J(\theta) = \text{Loss}_h(y^{(i)} \theta \cdot x^{(i)}) + \frac{\lambda}{2}\|\theta\|^2$

$\nabla_\theta J(\theta) = \nabla_\theta(\text{Loss}_h(y^{(i)} \theta \cdot x^{(i)}) + \frac{\lambda}{2}\|\theta\|^2) = \nabla_\theta \text{Loss}_h(y^{(i)} \theta \cdot x^{(i)}) + \nabla_\theta \frac{\lambda}{2}\|\theta\|^2$

(!) $\nabla_\theta \frac{\lambda}{2}\|\theta\|^2 = 2 \cdot \frac{\lambda}{2} \theta \cdot \frac{\partial \theta}{\partial \theta} \cdot \theta = \lambda\theta$ (chain rule)

(?) $\nabla_\theta \text{Loss}_h(y^{(i)} \theta \cdot x^{(i)}) = \nabla_\theta \begin{cases} 1 - y^{(i)} \theta \cdot x^{(i)}, & y^{(i)} \theta \cdot x^{(i)} \le 1 \\ 0, & y^{(i)} \theta \cdot x^{(i)} > 1 \end{cases}$

$= \begin{cases} -y^{(i)} \cdot x^{(i)} \\ 0 \end{cases}$

Combining the pieces:

$\nabla_\theta J(\theta) = \begin{cases} -y^{(i)} x^{(i)} + \lambda\theta, & y^{(i)} \theta \cdot x^{(i)} \le 1 \\ \lambda\theta, & y^{(i)} \theta \cdot x^{(i)} > 1 \end{cases}$

Online SVM update step

$\theta_{new} = \theta_{old} - \eta \nabla_\theta J(\theta) = \theta_{old} - \eta \cdot \begin{cases} -y^{(i)} x^{(i)} + \lambda\theta, & y^{(i)} \theta \cdot x^{(i)} \le 1 \\ \lambda\theta & y^{(i)} \theta \cdot x^{(i)} > 1 \end{cases}$

Now after every point we will move θ in the direction that minimizes the objective function.

We can formalize this process as an algorithm

3-10

Pegasos Algorithm (Online SVM)

$\theta = \vec{0}$

repeat T times:

  for $i = 1, \ldots, n$:

    if $y^{(i)} \theta \cdot x^{(i)} \leq 1$:

      $\theta = \theta - \eta \cdot (-y^{(i)} x^{(i)} + \lambda \theta)$

    else:

      $\theta = \theta - \eta \cdot (\lambda \theta)$

return θ