A Section 9: Support Vector Machines

Prepared & Presented by Will Claybaugh

CS109A Introduction to Data Science

Pavlos Protopapas and Kevin Rader

What do you get when you cross an elephant and a rhino?

Q: What does logistic regression think of LDA/QDA?

What do you get when you cross an elephant and a rhino?

Q: What does logistic regression think of LDA/QDA?

- LDA/QDA tell the complete story of how the data came to be
- Correspondingly, it makes heavy assumptions, and much can go wrong

A: You’re modelling too much

- Logistic doesn’t care how the X data came to be, it only tells the story of the Y data
- Since there are fewer assumptions, the math is more advanced and the method is slower

Anyone take the old SATs?

SVM:Logistic Regression::Logistic Regression:QDA

Less is More

SVMs

- Only predict the final class, not the probability of each class
- Make no assumptions about the data
- Still work well with large numbers of features

Our Path

- **I: Get comfy with the key expressions and concepts**
  - Bundles, signed distance, class-based distance
- **II: Extract the highlights of SVMs from the loss function**
  - Only certain observations matter; effects of the C parameter
- **III: Derivation of the primal and dual problems, fulfilling the promises from Part II**
  - Lagrangian, Piramal/Dual games, KKT conditions as souped-up “derivative=0”
- **IV: Interpret the dual problem and see SVMs in a new way**
  - SVMs can be seen as an advanced neighbors-style algorithm

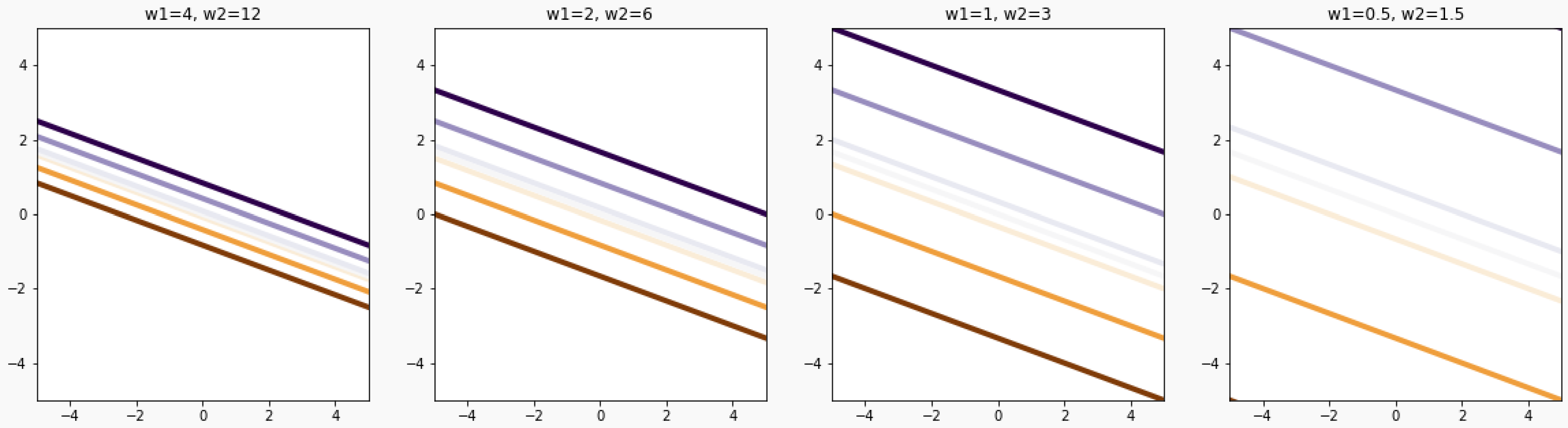
Part I

REVIEW

P

Act I: Setting

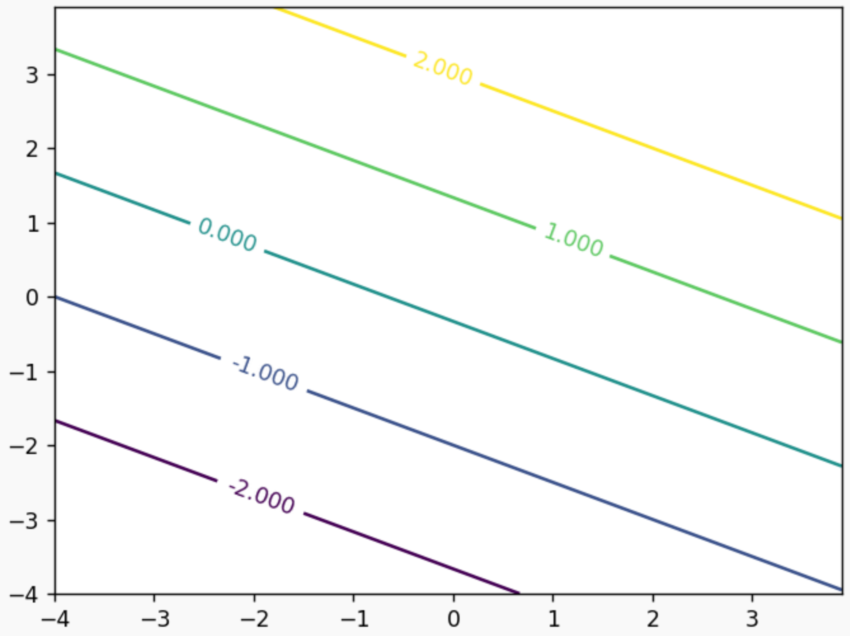
- Like Logistic regression, SVMs set three parameters:
  a weight on each feature (w1 and w2) and an intercept (b)
    - This is MORE than we need to define a line
    - So what are we really defining?



Key Concept #1

- Via $w^T x + b$, w and b define an output at each point of input space
- This is our first key quantity, and will live in our 'reminder corner'
- $w^T x + b$ gives us:
  - The rule to classify test points: if $w^T x + b$ is + classify as +; if - classify as -
  - A new measure of distance [from the decision boundary in units of $1/||w||$]
- We [arbitrarily] define +1 and -1 as the margin for a given w, b (bundle)

$w^T x + b$ =



Live Demo

$w^T x + b$: Signed distance

DEMO:

In the notebook, we manipulate w1, w2, and b to see how they affect the bundle produced

Conclusions:

- $w1$ and $w2$ control the slope of the bundle, and the larger the norm, the more tightly packed the bundle is
- b controls the height of the bundle, but its effect depends on the magnitude of $w1$ and $w2$

Key Concept #2

$w^T x + b$: Signed distance
$y_i(w^T x_i + b)$: Class-based distance

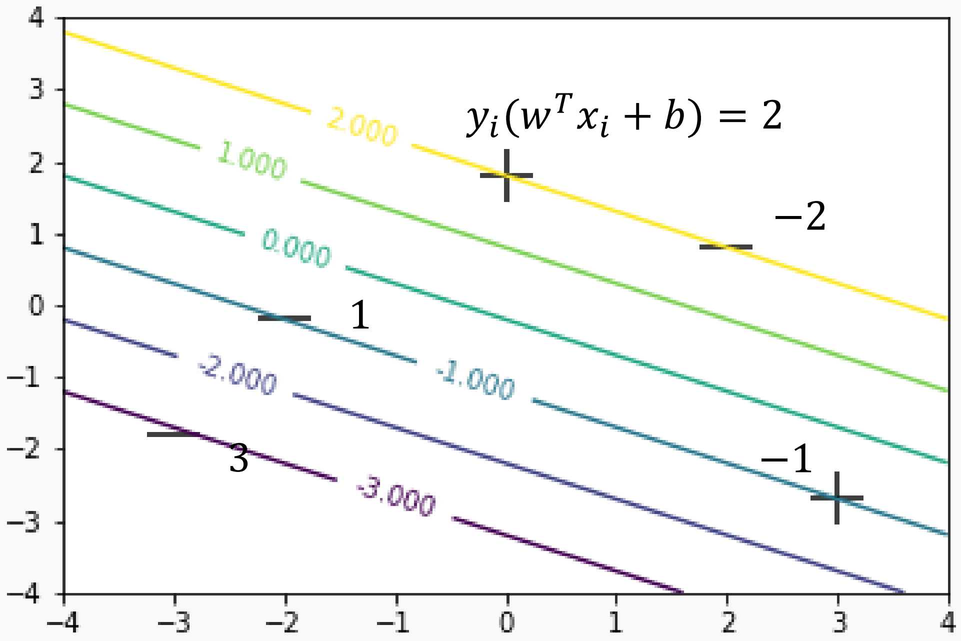
- The expression $y_i(w^T x_i + b)$ occurs a ton with SVMs
  - It takes the signed distance function and multiplies it by an observation's class
  - We're calling it "class-based distance"

Example:

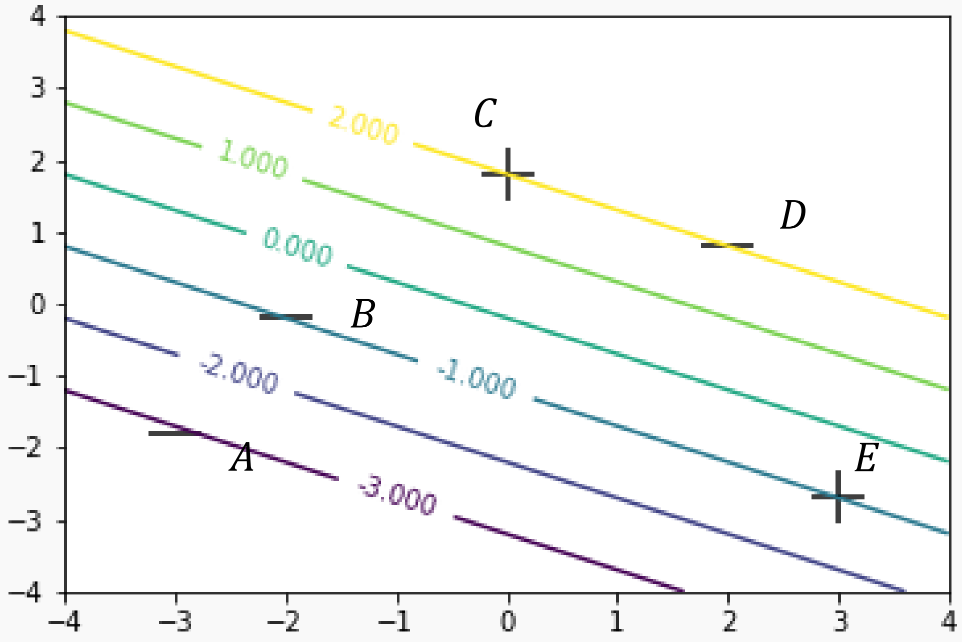
$y_i(w^T x_i + b)$

- is 0 on at the decision boundary
- is above 1 if you are safely beyond your margin
- is 1 (or less) if you are crowding the margin or misclassified
- is negative if you're really messing up

$y_i(w^T x_i + b) =$



A table of the key quantities at each point



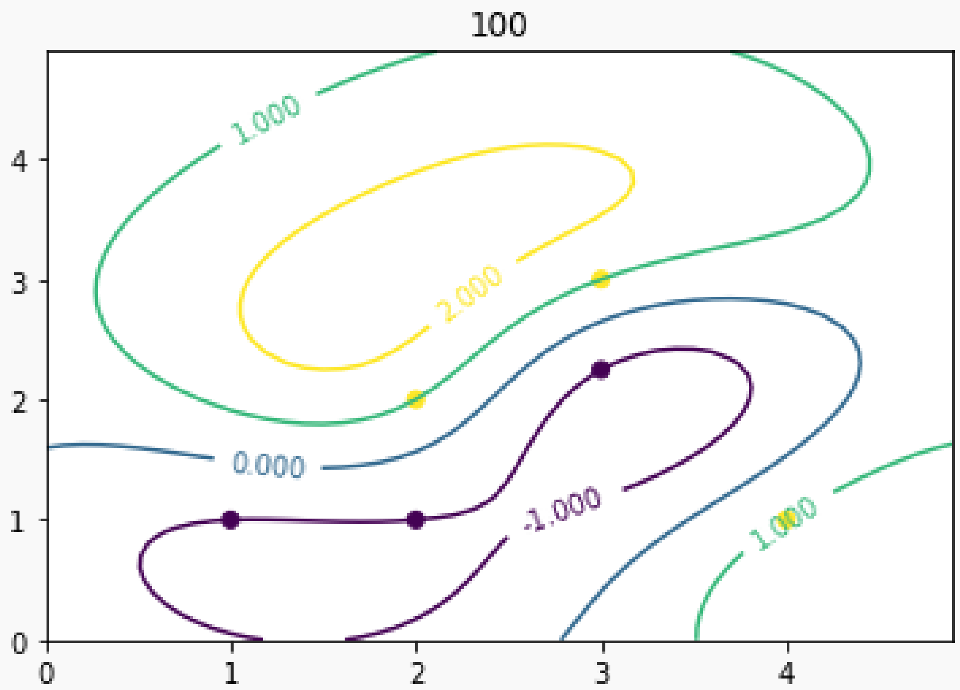
Point	Class	Signed Distance	Class-based distance	Loss
A	-	-3	3	None
B	-	-1	1	Marginal
C	+	2	2	None
D	-	2	-2	Misclass
E	+	-1	-1	Misclass

Kernels

$w^T x + b$: Signed distance
$y_i(w^T x_i + b)$: Class-based distance

The same 'signed distance' concepts apply to kernels, although:

1. The lines get wavy
2. The way we measure distance is less clear



Later on, we'll learn

- What kind distance is used for kernels
- Standard distance isn't what you think

Recap

$w^T x + b$: Signed distance
$y_i(w^T x_i + b)$: Class-based distance

Recap:

- We’re picking a best bundle (set of weights and b)
- The bundle implies a signed ‘distance’ $w^T x + b$ over the space, where 0 is the decision boundary
- Class-based distance $y_i w^T x_i + b$ is directly related to how sad we are about a training point

- Kernels put a wavy set of lines over the input space, instead of level ones

Part II

LOSS FUNCTIONS

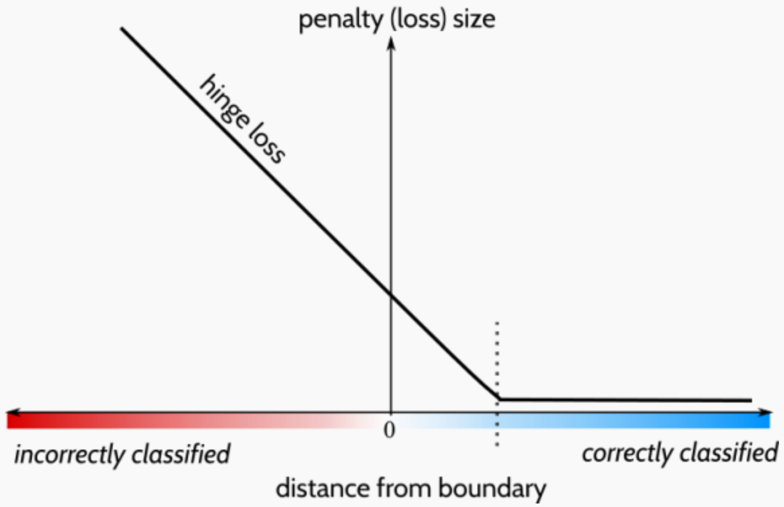
Hinge Loss

$w^T x + b$: Signed distance

$y_i(w^T x_i + b)$: Class-based distance

$1 - y_i(w^T x_i + b)$: Loss

We saw 1 was a critical value for $y_i(w^T x_i +$

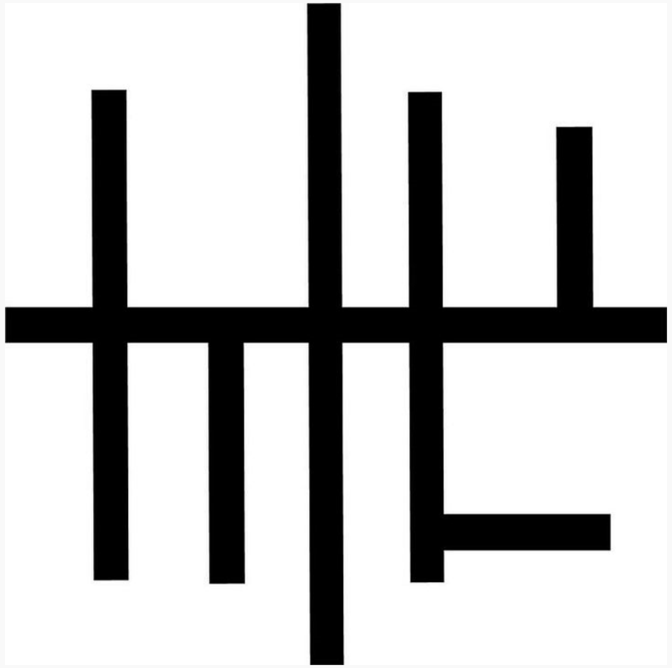
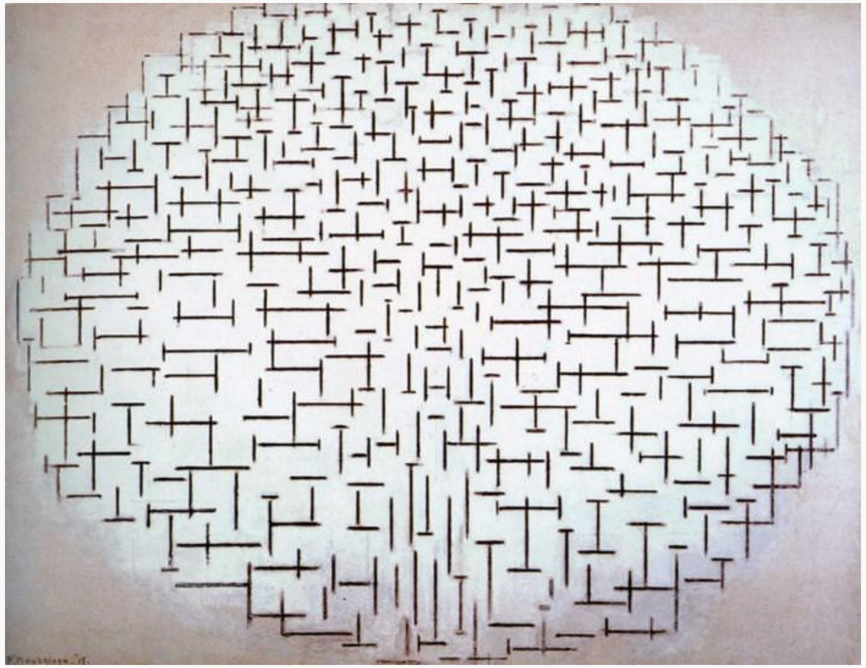
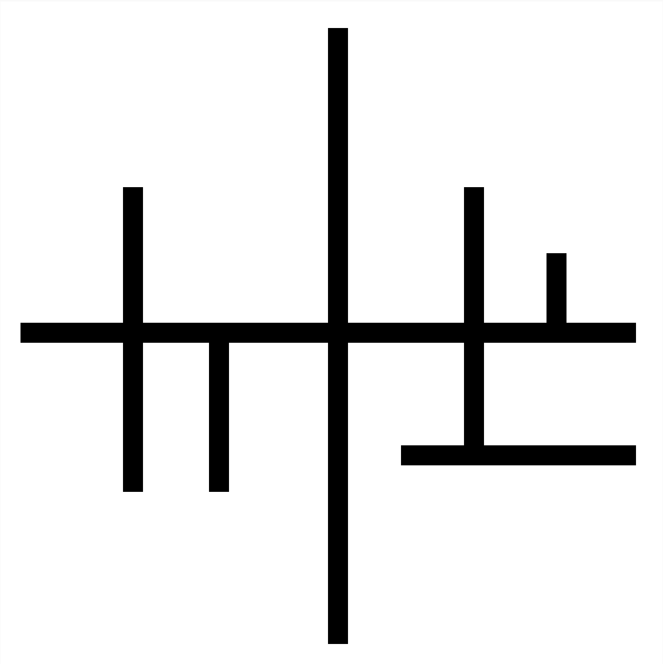


$$Loss = \max(1 - y_i(w^T x_i + b), 0)$$

Loss

$w^T x + b$: Signed distance
$1 - y_i(w^T x_i + b)$: Loss

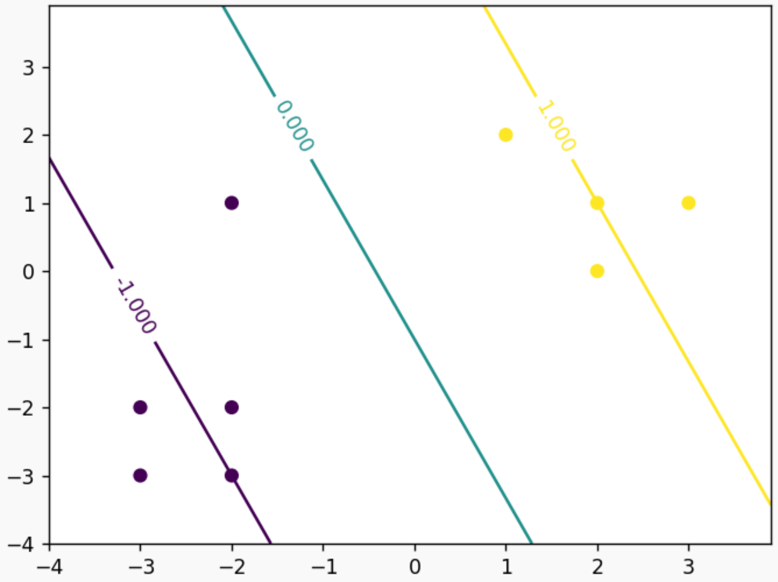
- Which do you like best?

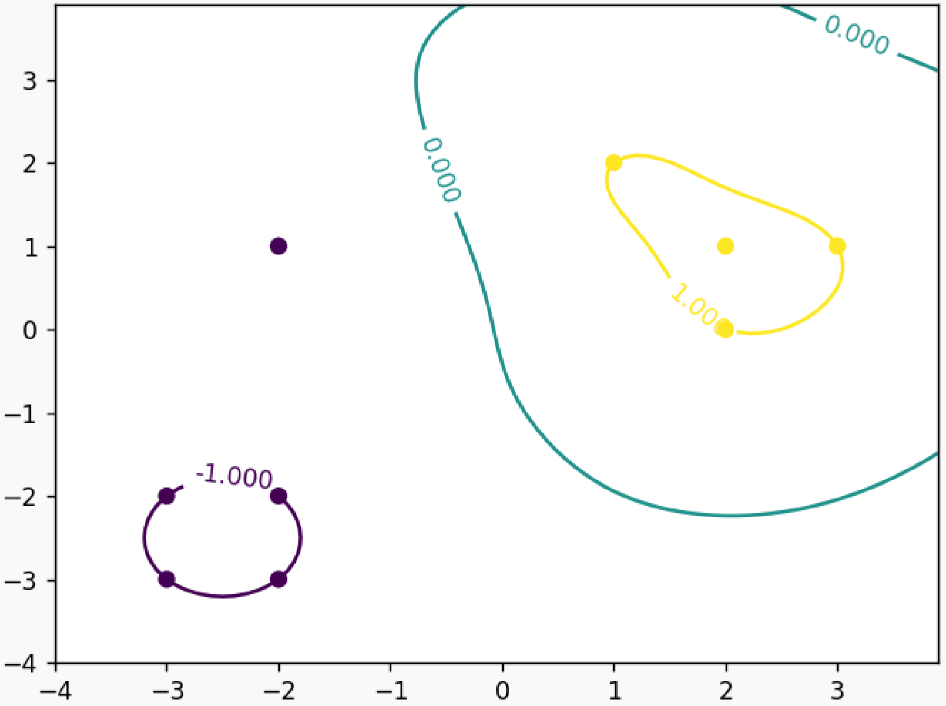




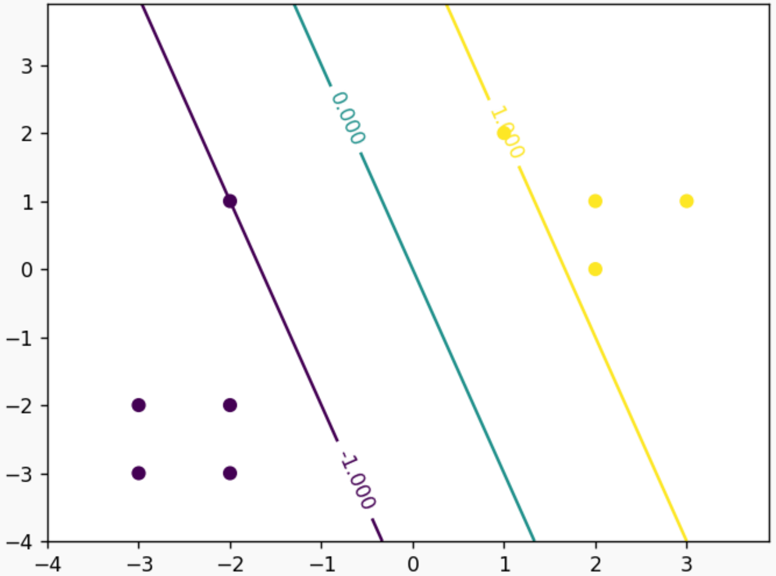
Act II: Loss

$w^T x + b$: Signed distance
$1 - y_i(w^T x_i + b)$: Loss

- Which do you like best?







The Loss Function

$w^T x + b$: Signed distance

$\|w\|^2 + C \sum_{train} \max(1 - y_i(w^T x_i + b), 0)$: Loss (margin+invasion)

- Tradeoff exists between wanting wider margins and discomfort with points inside the margins

$$Loss(w, b, train\ data) = \sum_{train} \max(1 - y_i(w^T x_i + b), 0) + \lambda \|w\|^2$$

- **View A**: minimize hinge loss, L_2 regularization

$$Loss(w, b, train\ data) = \|w\|^2 + C \sum_{train} \max(1 - y_i(w^T x_i + b), 0)$$

- **View B**: maximize the margin, but pay a price for points inside the margin (or misclassified)

Live Demo

$w^T x + b$: Signed distance

$\|w\|^2 + C\sum_{train} \max(1 - y_i(w^T x_i + b), 0)$: Loss (margin+invasion)

DEMO:

In the notebook, we manipulate C and see how the solution found by SVM changes

Conclusions:

- Big C: we do *anything* to reduce invasion losses
    - If seperable: finds separating plane
    - If not: lumps non-separable points into margin, separates the rest
- Small C: we stop caring about invasion (or even misclassification); just grow the margin

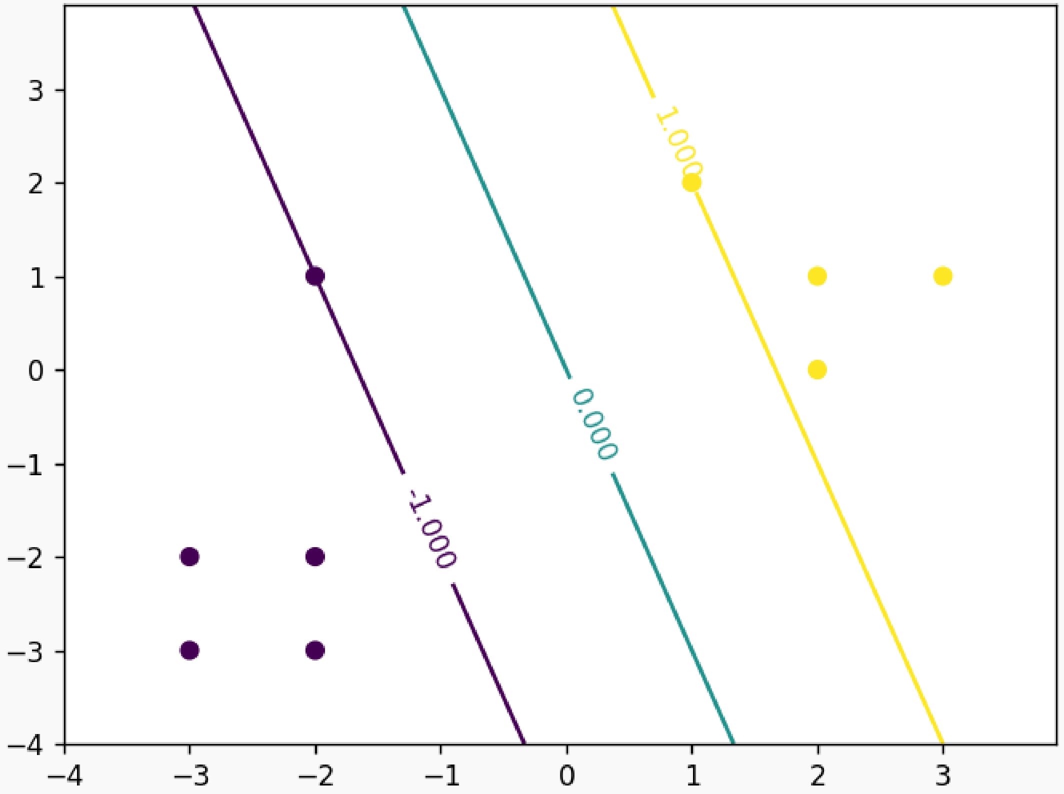
Observations

$w^T x + b$: Signed distance

$\|w\|^2 + C \sum_{train} \max(1 - y_i(w^T x_i + b), 0)$: Loss (margin+invasion)

Observations from SVM loss:

1. Hinge loss zero for most points
   – most points are behind the margin
2. Moving/deleting these points wouldn't change the solution
3. The outcome for a test point only depends on a handful of training points
   - Should be able to write output value as combination of (-2,1) and (1,2)



- Key question: *HOW* can we determine a test point's class using the few important training points?
- Leads to re-casting as a fancified neighbors algorithm

What to watch for

$w^T x + b$: Signed distance

$\|w\|^2 + C \sum_{train} \max(1 - y_i(w^T x_i + b), 0)$: Loss (margin+invasion)

Our reward for sitting through the math:

1. A recipe for the most important training points
2. A way to make decisions while throwing out most of the training data
3. A new and more powerful view of what SVMs do

Like studying linear regression’s loss minimization via calculus, but with a harder target and more advanced math

Part III

MATH

Ideas: http://cs229.stanford.edu/notes/cs229-notes3.pdf
Soft-Margin derivation: http://www.ccs.neu.edu/home/vip/teach/MLcourse/6_SVM_kernels/lecture_notes/svm/svm.pdf

Author's Proof

$w^T x + b$: Signed distance

$\|w\|^2 + C \sum_{train} \max(1 - y_i(w^T x_i + b), 0)$: Loss (margin+invasion)

Outline proof steps

1. Re-cast the loss function as a convex optimization
2. Re-write the one-player game into a two-player game (Primal)
3. Rewrite the two-player game into an equivalent game with opposite turn order (Dual)
4. Observe that assigning (mostly-zero) importance scores to each training point is equivalent to solving the original optimization (KKT)
5. Observe that our original SVM formulation was using a very counter-intuitive definition of distance, and we can do better

Optimization

$w^T x + b$: Signed distance

$\|w\|^2 + C \sum_{train} \max(1 - y_i(w^T x_i + b), 0)$: Loss (margin+invasion)

Objective: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$ Constraints: $\xi_i \geq 1 - y_i(w^T x_i + b)$, $\xi_i \geq 0$

Our goal:

$$\min_{\mathrm{w,b}} \|w\|^2 + C \sum_{train} \max(1 - y_i(w^T x_i + b), 0)$$

First, re-write to an optimization problem with constraints:

$$\min_{\mathrm{w,b},\xi_i} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

Such that

$$\xi_i \geq 1 - y_i(w^T x_i + b),\ \xi_i \geq 0\ for\ all\ i$$

Basically, we delete loss and introduce some ξ_i variables that you get to set

Why is this the same problem?

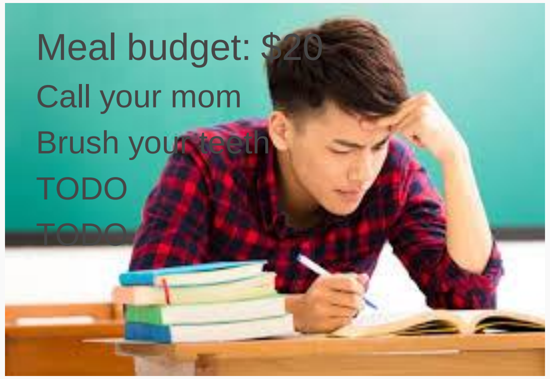
- ξ_i must be at least as big as the loss: you'd be dumb to set them to anything bigger than the loss
- Now you're back to minimizing norm+loss

$w^T x + b$: Signed distance

Objective: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$ Constraints: $\xi_i \geq 1 - y_i(w^T x_i + b),\ \ \xi_i \geq 0$

Yeah, 'hypothetically'…

- You're trying to plan your week
  - You have to choose how much time you allocate to study, work, etc.
- There are hard constraints
  - Max 20 hours of work-study
  - CS109 maximum 1 day late
  - CS207 due by Thursday
  - Minimum 4 hours of sleep
    (Per week)
  - No asec OH on Wednesday
  - Job interview Thursday
  - Project meeting by Monday
  - Meal budget: $20
  - Call your mom
  - Brush your teeth
  - TODO
  - TODO



- What if the constraints were flexible?
  - You'd know what the cost of being each late day is
  - And how about a reward for getting work done *early!*

Lagrange Multipliers

$w^T x + b$: Signed distance

Objective: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$ Constraints: $\xi_i \geq 1 - y_i(w^T x_i + b), \xi_i \geq 0$

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

- This brings us to the Lagrangian

  - It takes all the mandatory requirements and attaches costs to them
- For $\xi_i \geq 1 - y_i(w^T x_i + b)$ we attach a cost α_i for each unit that $1 - y_i(w^T x_i + b)$ exceeds ξ_i
  - Likewise for $\xi_i \geq 0$
- Overall, we get

$$\underbrace{\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i}_{\text{Original objective}} + \underbrace{\sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i]}_{\text{Cost (or benefit) from } \xi_i \geq 1 - y_i(w^T x_i + b) \text{ “objective”}} + \underbrace{\sum_{i=1}^{n} -\beta_i\xi_i}_{\text{Cost (or benefit) from } \xi_i \geq 0 \text{ “objective”}}$$

Doom, Khazad

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \ - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

- But there aren't actually penalties for each day late, or rewards for being early…
- What you need is a demon
- The demon takes any plan you make and manipulates the α_i and β_i costs
  - So you better present a plan that actually meets the constraints

You shall not pass

Primal scream

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \ - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

We have a two-player game (you and demon) equivalent to the original hard-constraint problem:

You choose the parameters

$$\min_{w,b,\xi_i} \max_{\alpha_i \geq 0} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$$

Then the demon chooses the costs

- The demon will try to screw you, so you'll only propose points that meet all constraints
  - And you'll try to minimize the original objective
- Level one complete: we wrote the "Primal Problem"
  - Now, like Gandalf and the Balrog, there's a Duel

Enjoy Capitalism

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i \xi_i$

- Still pondering how to set your schedule, an Econ 101 student walks by

- "The free market solves everything"
    - "What about companies polluting?"
    - "Well, charge them for each ton of carbon they emit
        - If you get prices right, they'll stop"

- Could you set the costs/rewards yourself and let the free market minimize the objective?
    - Can you set costs that guide them to *the same solution as the original*?

The Dual

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \; - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

Reversing the turn order (the min and max), we get

You set the costs/rewards

$$\max_{\alpha\geq 0}\ \min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$$

The market tries to minimize the objective, including any costs/subsidies you offer

- No guarantee there are prices that force the free market to obey constraints and give the same solution as the original
- However, because …
  - 1) the objective is convex and 2) the constraints are convex and 3) there is some solution that fits the constraints (pick ξ large)]:
- …there *are* such prices, even if they're hard to write down

KKT Conditions

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

IF the dual can be rigged to give the same solution as the original

THEN we get helpful facts about the solution called the KKT conditions

- KKT can be used to check a candidate solution, or derive facts of the eventual solution
  1. Derivative of Lagrangian (wrt any parameter) is 0
  2. Derivative of Lagrangian (wrt cost of violating an equality) is 0
  3. Constraint function ≤ 0 (i.e. constraints are satisfied)
  4. Cost*constraint = 0 (only binding constraints get non-zero costs)
  5. Cost ≥ 0 (costs are positive)

That was… mostly Econ

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^n \xi_i + \sum_{i=1}^n \alpha_i[1 - y_i(w^T x_i + b) \; - \xi_i] + \sum_{i=1}^n -\beta_i \xi_i$

Let's recap

- Wrote our optimization problem (minimize loss)
- Massaged into a *convex* optimization problem
    - Horary for hinge loss and convexity
- Applied Lagrangian to make progress on the constrained optimization
    - Costs instead of mandates, but a demon controls costs
- Convexity let us study the dual problem instead
    - We control costs, but it's not always possible to set them well
- KKT gave us a bunch of useful properties we're about to apply

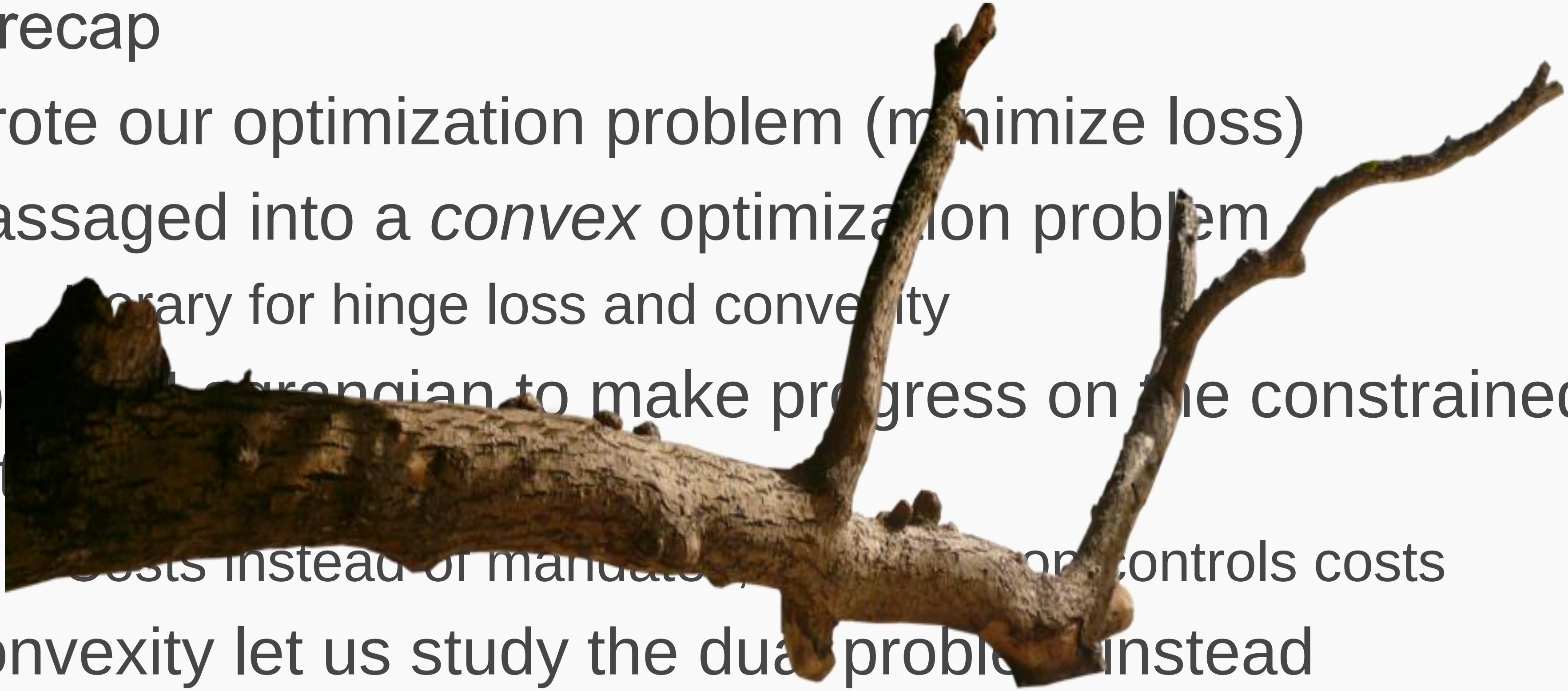
That was… mostly Econ

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \; - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

Let’s recap

- Wrote our optimization problem (minimize loss)
- Massaged into a *convex* optimization problem
  - …orary for hinge loss and convexity
- Ap… Lagrangian to make progress on the constrained opt…
  - Costs instead of mandate… controls costs
- Convexity let us study the dual problem instead
  - We control costs, but it’s not always possible to set them well
- KKT gave us a bunch of useful properties we’re about to apply



Part III: Part II

THE LAST MATH

The first rule of KKT is

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \; - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

1) $w = \sum_{i=1}^{n}\alpha_i[y_i x_i]$

Rule 1 (for w) says that

$$\nabla_w \; \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b)] - \xi_i + \sum_{i=1}^{n} -\beta_i\xi_i = 0$$

The derivative is practically trivial:

$$w + \sum_{i=1}^{n}\alpha_i[-y_i x_i] = 0$$

$$w = \sum_{i=1}^{n}\alpha_i[y_i x_i]$$

Conclusions

- The w are just a weighted sum of the training points
- If we know α_i, we can make classification decisions using only the x and y data

The second rule of KKT is

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \ - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

1) $w = \sum_{i=1}^{n}\alpha_i[y_i x_i]$

2) $\sum_{i=1}^{n}\alpha_i y_i = 0$

Rule 1 (for b) says that

$$\nabla_b \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i = 0$$

The derivative is again trivial:

$$\sum_{i=1}^{n}\alpha_i[-y_i] = 0$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

Conclusions

- The α_i assigned to the positive class cancel with the α_i assigned to the negative class
- Not very insightful, but allows a simplification later

The third rule of KKT is

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \; - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

1) $w = \sum_{i=1}^{n}\alpha_i[y_i x_i]$

2) $\sum_{i=1}^{n}\alpha_i y_i = 0$

3) $C = \alpha_j + \beta_j$

Rule 1 (for ξ_j) says that

$$\nabla_{\xi_i} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i = 0$$

The derivative is again trivial:

$$C - \alpha_j - \beta_j = 0$$

$$C = \alpha_j + \beta_j, \qquad \text{for all j}$$

Conclusions

- The cost of setting ξ_i above the loss and the cost of setting ξ_i below 0 add up to the total cost associated with ξ_i
- Again, not terribly informative, but useful on the next slide

1) $w = \sum_{i=1}^{n}\alpha_i[y_i x_i]$

2) $\sum_{i=1}^{n}\alpha_i y_i = 0$

3) $C = \alpha_j + \beta_j$

That's all the facts we need. Let's simplify the dual.

The Hardest Part… Is Cleaning Up

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \ - \xi_i] + \sum_{i=1}^{n} -\beta_i \xi_i$

1) $w = \sum_{i=1}^{n}\alpha_i[y_i x_i]$

2) $\sum_{i=1}^{n}\alpha_i y_i = 0$

3) $C = \alpha_j + \beta_j$

Starting from the Lagrangian

$$\frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i \xi_i$$

Fact 3 ($C - \alpha_j - \beta_j = 0$ for all j) to kill C

Rearrange

$$\frac{1}{2}w^T w + \sum_{i=1}^{n} C\xi_i + \sum_{i=1}^{n} -\beta_i \xi_i + \sum_{i=1}^{n} -\alpha_i \xi_i \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b)]$$

Apply the fact

$$\frac{1}{2}w^T w + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b)]$$

http://www.ccs.neu.edu/home/vip/teach/MLcourse/6_SVM_kernels/lecture_notes/svm/svm.pdf

The Hardest Part… Is Cleaning Up

$w^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) \;- \xi_i] + \sum_{i=1}^{n} -\beta_i\xi_i$

1) $w = \sum_{i=1}^{n}\alpha_i[y_i x_i]$

2) $\sum_{i=1}^{n}\alpha_i y_i = 0$

Copy over from slide above

$$\frac{1}{2}w^T w + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b)]$$

Fact 2 ($\sum_{i=1}^{n}\alpha_i y_i = 0$) to kill b

Rearrange

$$\frac{1}{2}w^T w + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i)] + \sum_{i=1}^{n}\alpha_i y_i b$$

Apply the fact

$$\frac{1}{2}w^T w + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i)]$$

http://www.ccs.neu.edu/home/vip/teach/MLcourse/6_SVM_kernels/lecture_notes/svm/svm.pdf

$\sum_{j=1}^{n} \alpha_j y_j x_j^T x + b$: Signed distance

$w^T x + b$: Signed distance

The Hardest Part… Is Cleaning Up

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} -\beta_i \xi_i$

1) $w = \sum_{i=1}^{n}\alpha_i[y_i x_i]$

Copy over from slide above

$$\frac{1}{2}w^T w + \sum_{i=1}^{n}\alpha_i[1 - y_i(w^T x_i)]$$

Fact 1 ($w = \sum_{j=1}^{n}\alpha_j[y_j x_j]$) to kill w

Rearrange

$$\frac{1}{2}w^T w - \sum_{i=1}^{n}\alpha_i y_i(w^T x_i) + \sum_{i=1}^{n}\alpha_i$$

Apply the fact

$$\frac{1}{2}\sum_{i=1}^{n}\alpha_i[y_i x_i^T]\sum_{j=1}^{n}\alpha_j[y_j x_j] - \sum_{i=1}^{n}\alpha_i y_i\left(\sum_{j=1}^{n}\alpha_j[y_j x_j^T]\,x_i\right) + \sum_{i=1}^{n}\alpha_i$$

http://www.ccs.neu.edu/home/vip/teach/MLcourse/6_SVM_kernels/lecture_notes/svm/svm.pdf

The Hardest Part… Is Cleaning Up

$\sum_{j=1}^{n} \alpha_j y_j x_j^T x + b$: Signed distance

Lagrangian: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i [1 - y_i(w^T x_i + b) - \xi_i] + \sum_{i=1}^{n} \beta_i \xi_i$

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j$

Copy

$$\frac{1}{2}\sum_{i=1}^{n} \alpha_i [y_i x_i^T] \sum_{j=1}^{n} \alpha_j [y_j x_j] - \sum_{i=1}^{n} \alpha_i y_i \left(\sum_{j=1}^{n} \alpha_j [y_j x_j^T]\, x_i \right) + \sum_{i=1}^{n} \alpha_i$$

Simplify

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j + \sum_{i=1}^{n} \alpha_i$$

Final form:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j$$

Such that $0 \le \alpha_i \le C$ for all i and $\sum_{i=1}^{n} \alpha_i y_i = 0$

Part IV

WHAT IT MEANS

Tune Back In Now

$\sum_{j=1}^{n} \alpha_j y_j x_j^T x + \mathrm{b}$: Signed distance

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j$

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j$$

Such that $0 \leq \alpha_i \leq C$ for all i and $\sum_{i=1}^{n} \alpha_i y_i = 0$

Interpretation time: what the *heck* are the $\boldsymbol{\alpha_i}$?

1. Lagrangian view: the cost associated with each point; how much the objective would improve if we got to move that point
2. New view: the raw importance of each point

Explanation:

- The first goal is to maximize the alphas, but there's a second term punishing big alphas
- When is that term big?

What Support Looks Like

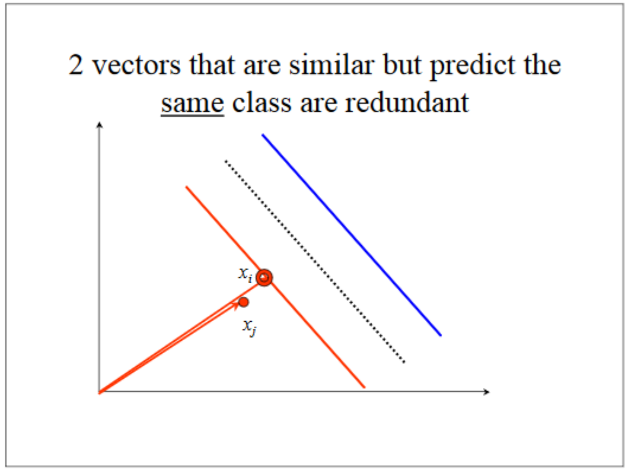
$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j$$

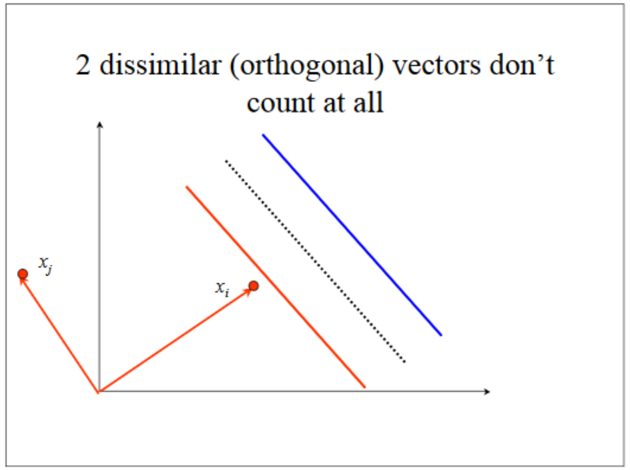
Large α_i **hurt** us when they're associated with observations that are

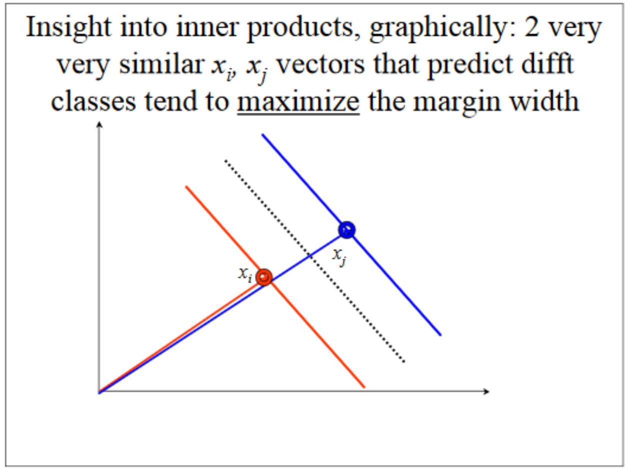
1) From the same class
2) Pointing in the same direction

Large α_i **help** us when they're associated with observations that are

1) From different classes
2) Pointing in the same direction







http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

Further, our predictions depend on the α_i

$$\text{Decision} = w^T x + b = \sum_{j=1}^{n} \alpha_j y_j x_j^T x + \mathrm{b}$$

- We make our decision by
  - Measuring the test point x's similarity to each training point x_j
  - Weighting by the training point's overall importance (α_j)
  - Summing over all training points, comparing the + score against the – score (set by y_j)
- SVMs are an intelligent form of nearest neighbors!!!
  - We consider how similar our new point is to each training point
  - In addition, each training point has a raw importance score
  - (What does KNN think about SVMs?)

$\sum_{j=1}^{n} \alpha_j y_j x_j^T x + \mathrm{b}$: Signed distance

Simplified Dual: $\max\limits_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j$

Example: classify $O = (1{,}0)$

$$\sum_{j=1}^{n} \alpha_j y_j x_j^T x + \mathrm{b}$$

Contributions:

– (.03)(-1)(-2) = .06

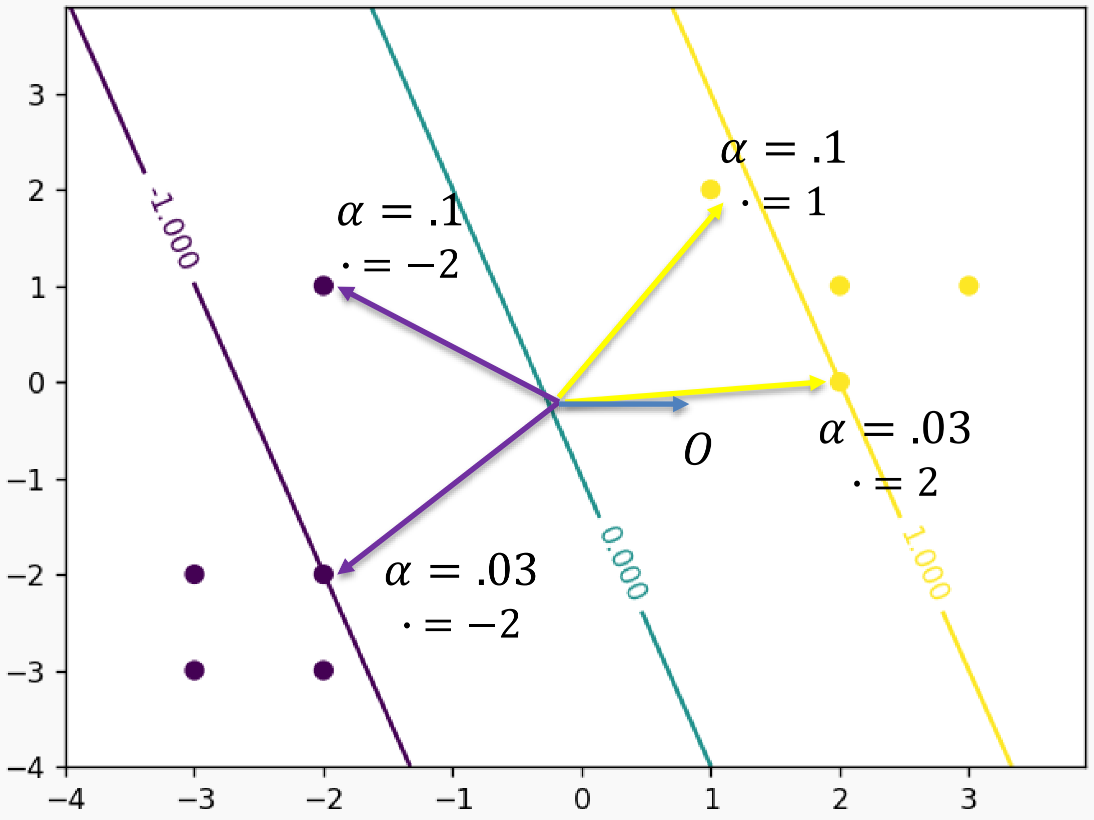
– (.1)(-1)(-2) = .2

– (.1)(1)(1) = .1

– (.03)(1)(2) = .06

– b =.16

Total: .58 -> classify as +



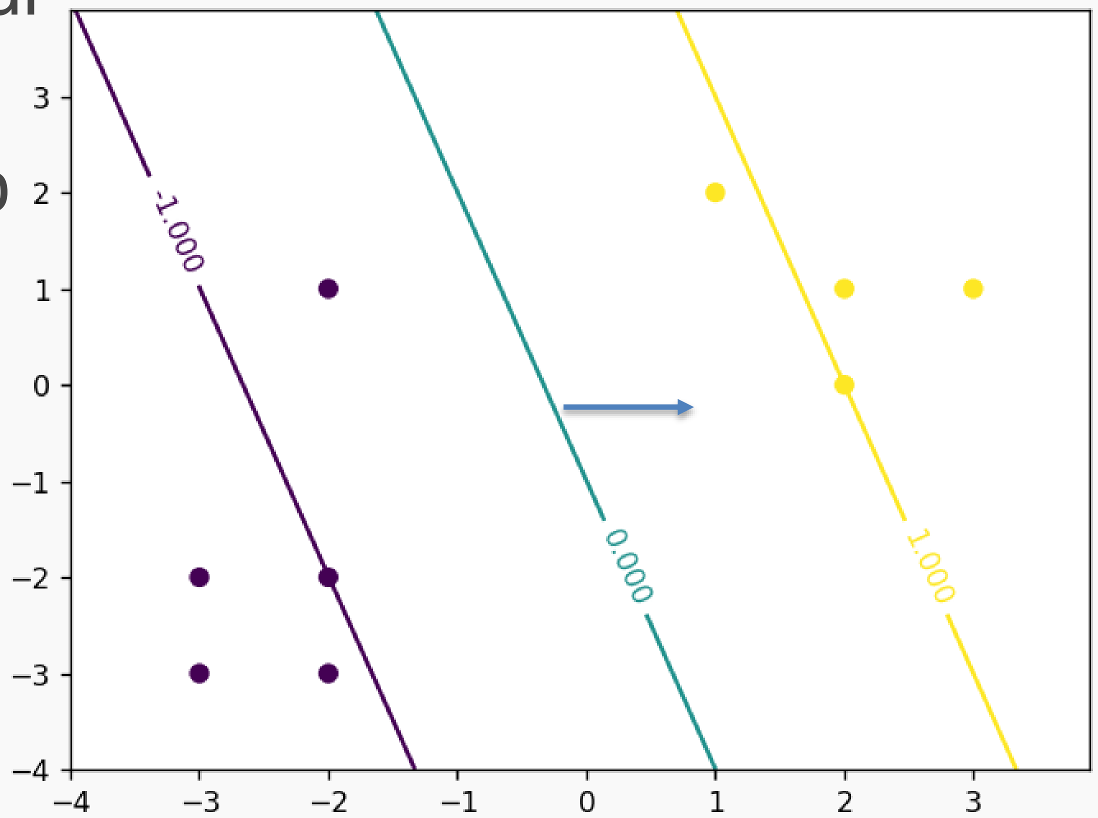
Kernels

$\sum_{j=1}^{n} \alpha_j y_j x_j^T x + \text{b}$: Signed distance

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i x_i^T x_j$

There's something weird about our calculation

- Our vector (1,0) is as similar to (2,0) as it is to (2,20)
- Is there a more meaningful measure of similarity?



Part IV: Part II

KERNELS

Kernels

$\sum_{j=1}^{n} \alpha_j y_j K(x_j, x) + \mathrm{b}$: Signed distance

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i k(x_j, x_i)$

Maximum margin view:

- Kernels map to a larger space where the classes *can* be separated by a plane
- Want to pick the plane with most margin

Neighbors view:

- Kernels define a measure of similarity between observations
- Classify based on test point's similarity to training points, and importance of training points

Example kernel: RBF

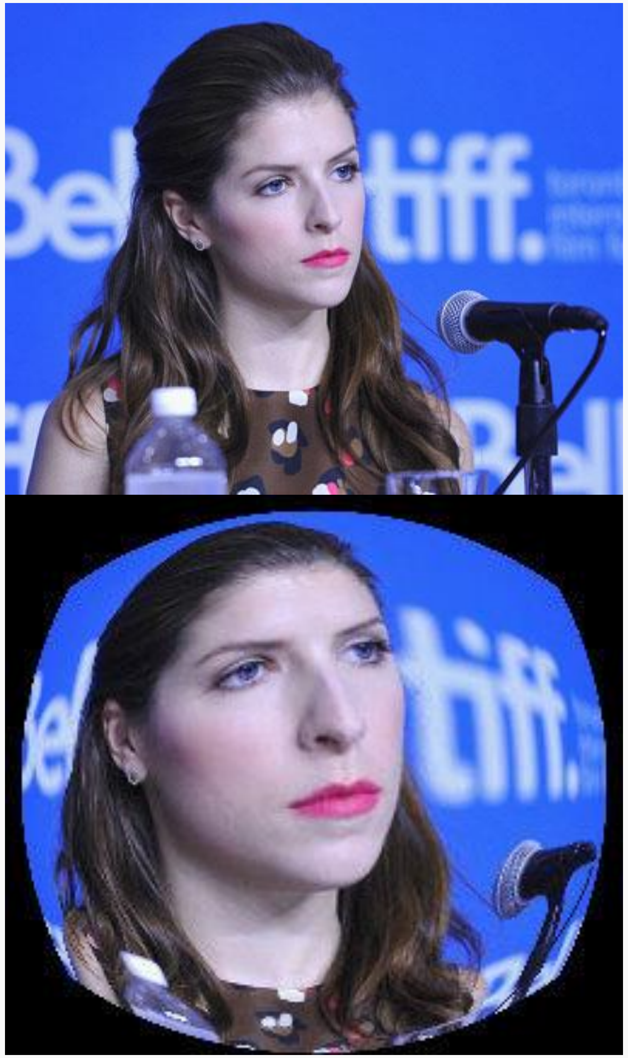
$\sum_{j=1}^{n} \alpha_j y_j K(x_j, x) + \mathrm{b}$: Signed distance

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i \; - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i k(x_j, x_i)$

RBF kernel:

$$rbf(x, y) = e^{-\left(\frac{\|x-y\|}{\gamma}\right)^2}$$

- Based on actual distance between points
- Similarity decreases rapidly because of e^{-dist}
- γ determines a ‘cliff’ because of the $(\;)^2$
  - if x and y are within γ, fraction <1
  - $\rightarrow$ they are more similar than you think
- It’s like a fishbowl lens



Kurtz, Sanders, Mustard, and Mustang

$\sum_{j=1}^{n} \alpha_j y_j K(x_j, x) + \text{b}$: Signed distance

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i k(x_j, x_i)$

RBF kernel has a geographic character to it:
it uses literal Euclidean distance

Other kernels (similarity measures) exist for:

- Documents
- Points in graphs
- Randomly adding polynomial terms
- Geostatistics
- Images
- Sound
- Many more

http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/

What makes a valid kernel?

$\sum_{j=1}^{n} \alpha_j y_j K(x_j, x) + \text{b}$: Signed distance

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i k(x_j, x_i)$

What makes a valid kernel?

a) Think of a set of features and compute the inner product post-transformation
b) Find a function so that no matter what points x you feed in, the matrix you build is Positive Semi-Definite (all eigenvalues ≥ 0)
    a) This is a Reproducing Kernel Hilbert Space
    b) Don’t ask.

    …Or take ES 201 :)

What if I need to *use* these?

$\sum_{j=1}^{n} \alpha_j y_j K(x_j, x) + b$: Signed distance

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i k(x_j, x_i)$

Practical kernel advice:

- Consider domain-specific kernels
- If more features than observations, you probably want linear
- If more observations than features, try RBF, but it may be slow

Other practical advice:

- SKlearn points out that its kernel implementation is too slow for >5-10K observations / features
    - LinearSVC scales to millions, though no kernels allowed

REVIEW
P

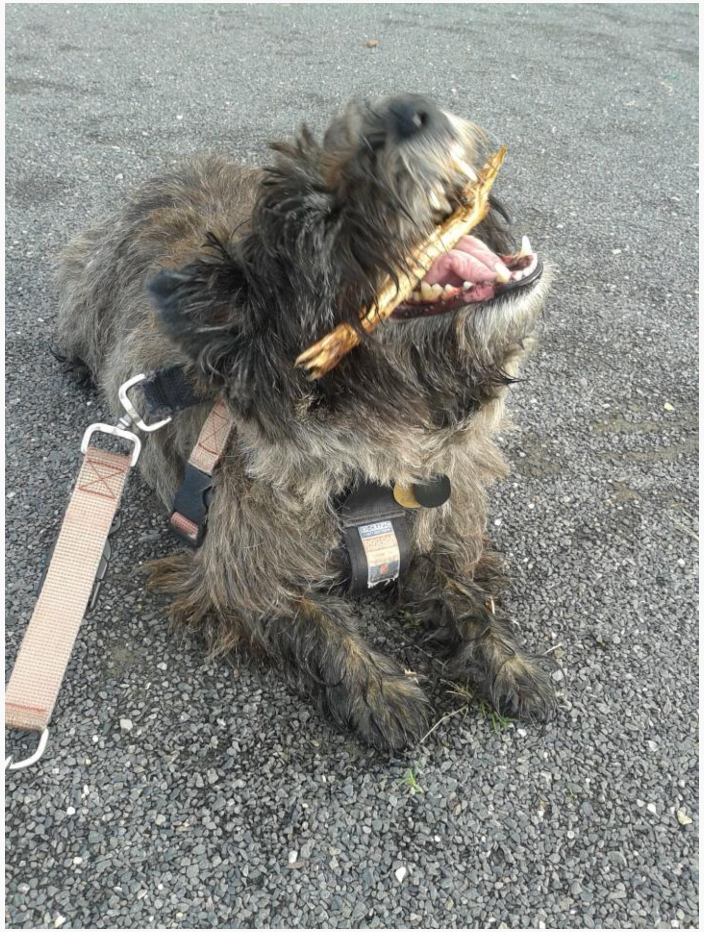
Summary

$\sum_{j=1}^{n} \alpha_j y_j K(x_j, x) + \text{b}$: Signed distance

Simplified Dual: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_j y_i k(x_j, x_i)$

In toto, here's what should stick:

- SVMs define bundles, not boundaries
- Convex optimization (here) is a better version of derivative = 0
  - Lagrangian, Primal, Dual;
  - Costs, Demons, Capitalism
- SVMs are BOTH
  - Drawing maximum margin plane
  - Measuring similarity to and importance of neighbors
- Kernels are how we define custom similarity



Q: What does logistic regression think of LDA/QDA?

A: What does KNN think of SVMs?