Tools & Models for Data Science

Support Vector Machines

Chris Jermaine & Risa Myers

Rice University

ML Roadmap

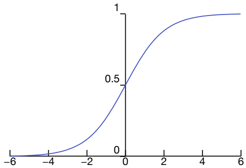
- Linear Regression
- Generalized Linear Models

Alternatives to Logistic Regression

- The “Big Three” for classification
  - Logistic regression
  - Support Vector Machines
  - kNN

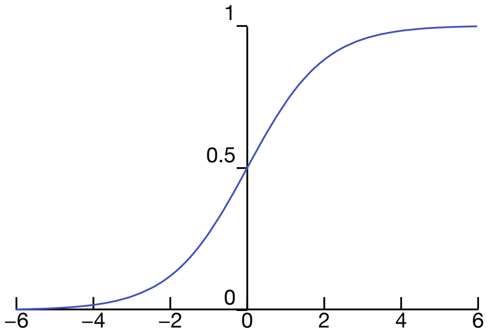
Recall: Logistic Regression

- Specialized case of GLM
- Where the error can come from a family of distributions
- If we use the Bernoulli distribution
  - We get Logistic Regression
  - There is a different, but mathematically equivalent representation of Logistic Regression using the sigmoid function



The Sigmoid Function

- $S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$
- There are others (Hyperbolic tangent, Arctangent, ...)
- Key properties
  - Monotonic
  - “S” shape
  - Horizontal asymptotes



Training and Inference

- During training
  - During training we have a set of n $\{x_i, y_i\}$ pairs
  - We use a method (MLE, gradient ascent, etc.) to learn the vector of regression coefficients, r
  - To maximize the log likelihood function

$$\sum_i y_i(x_i \cdot r) - \log(1 + e^{x_i \cdot r})$$

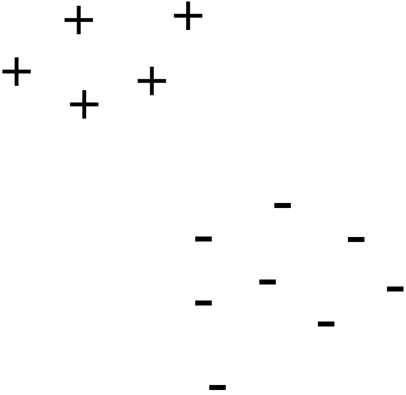
- During inference
  - We use r and the new x_is to choose the label, y_i, to maximize the LLH
  - y_i in $\{0, 1\}$
  - Basically, if $x_i \cdot r < 0$, pick $y_i = 0$

Human Logistic Regression

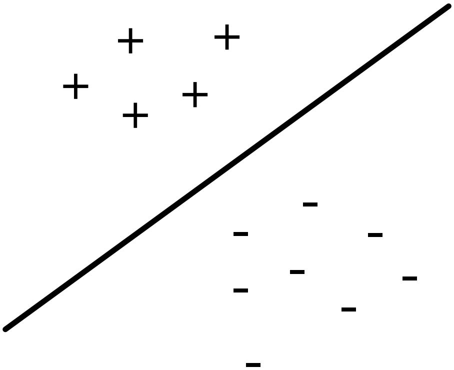
- When the data are "linearly separable"
  - Recall

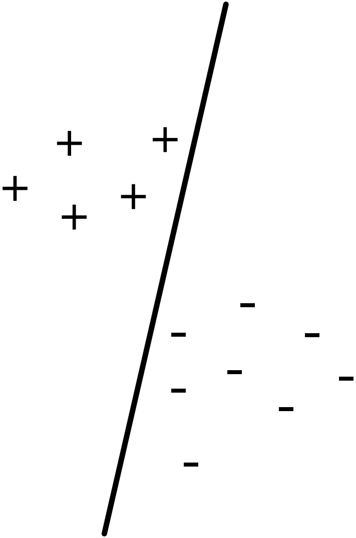
$$LLH = \sum_i y_i(x_i \cdot r) - \log(1 + e^{x_i \cdot r})$$

  - Where x_i are the training data
  - r is the vector of regression coefficients
  - $y_i = sign(x_i \cdot r)$
  - Where $sign(x_i \cdot r) = \begin{cases} 1 & \text{if } x_i \cdot r > 0 \\ 0 & \text{if } x_i \cdot r < 0 \end{cases}$
  - Assigns a class based on which side of the "cutting" line the point is on
  - ? Where does the cutting line go?

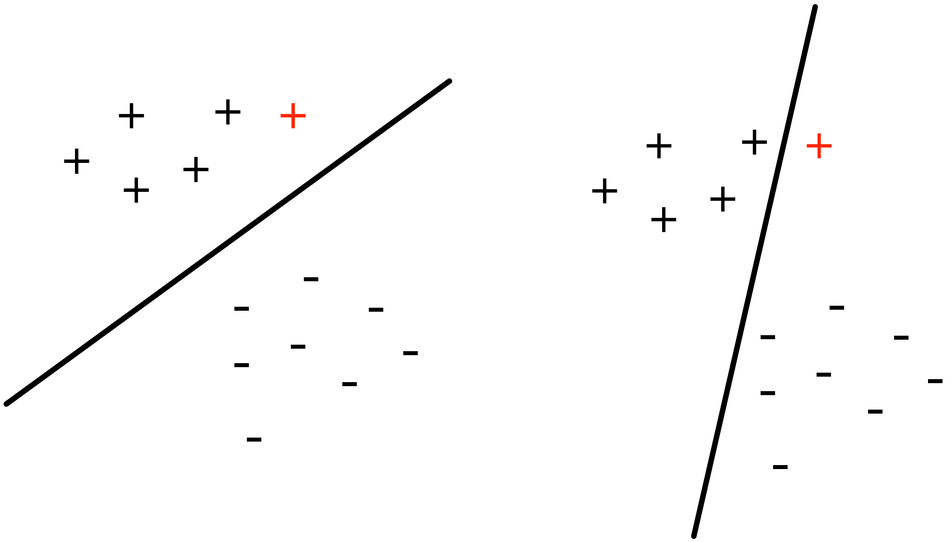


Humans vs. Machine Logistic Regression



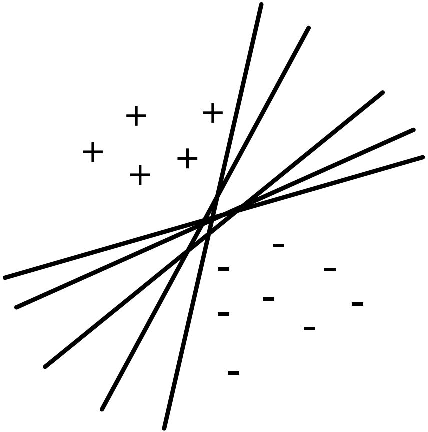


Humans vs. Machine Logistic Regression



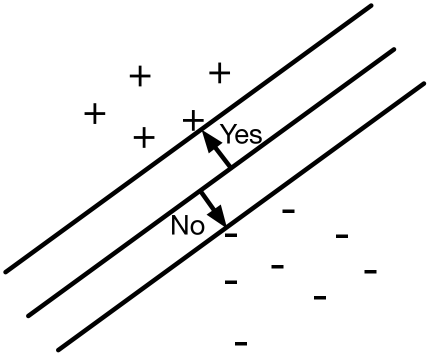
Problem With Un-Regularized Logistic Regression

- Possible to choose infinite models that get infinite LLH
- Just choose ANY cutting plane between classes
- That is, choose any r that perfectly classifies data, so $y_i = sign(x_i \cdot r)$
- Then use $r' = BIG \times r$
- And $\sum_i y_i (x_i \cdot r') - \log(1 + e^{x_i \cdot r'})$ will be really big
- Infinitely many models give infinite LLH
- Bad: Not clear which plane is preferred



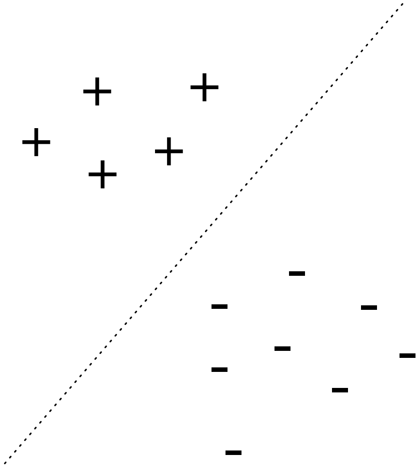
SVMs: Geometric, Not Probabilistic

- What should a classifier do in this super-easy case?
  - Just put the widest strip possible between two classes
  - Future points above center of strip are “yes”
  - Below center of strip are “no”
  - Points that keep strip from expanding are “support vectors”



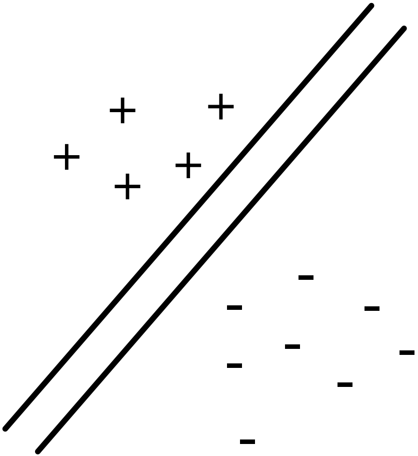
SVMs Steps 1

1. Choose an arbitrary, infinitely thin cutting plane



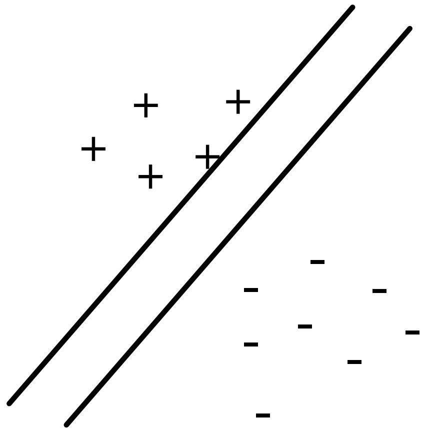
SVMs Steps 2

1. Choose an arbitrary, infinitely thin cutting plane
2. Make it wider



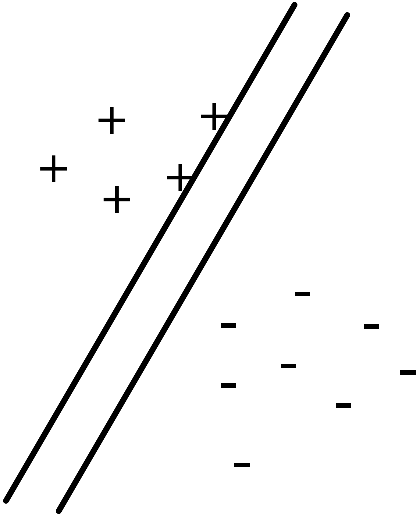
SVMs Steps 3

1. Choose an arbitrary, infinitely thin cutting plane
2. Make it wider
3. Keep widening until you hit a data point



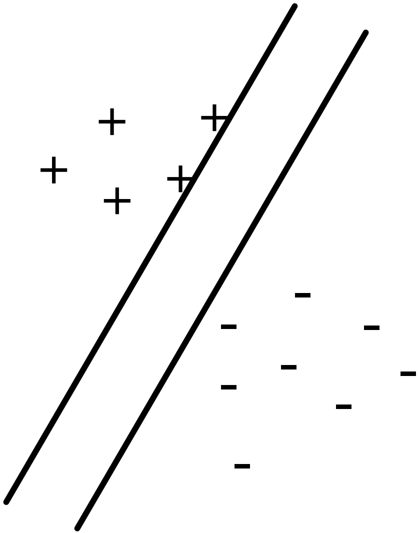
SVMs Steps 3

1. Choose an arbitrary, infinitely thin cutting plane
2. Make it wider
3. Keep widening until you hit a data point, rotating if needed



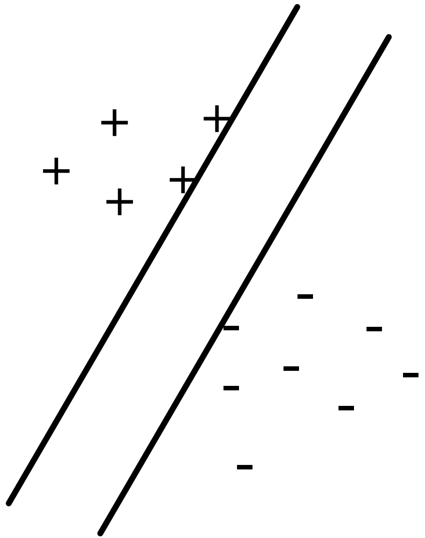
SVMs Steps 4

1. Choose an arbitrary, infinitely thin cutting plane
2. Make it wider
3. Keep widening until you hit a data point, rotating if needed
4. Expand in the other direction



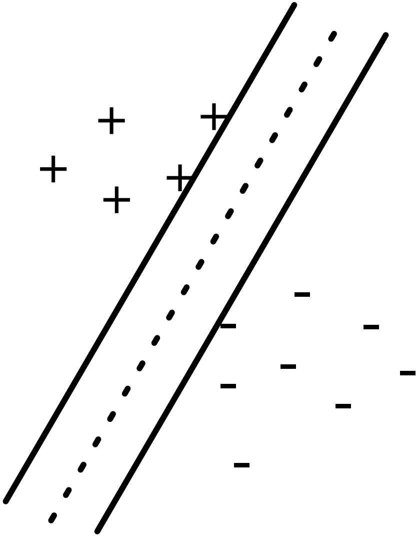
SVMs Steps 4

1. Choose an arbitrary, infinitely thin cutting plane
2. Make it wider
3. Keep widening until you hit a data point, rotating if needed
4. Expand in the other direction



SVMs Steps 5 & 6

1. Choose an arbitrary, infinitely thin cutting plane
2. Make it wider
3. Keep widening until you hit a data point, rotating if needed
4. Expand in the other direction
5. Stop when strip is wedged in by three points
6. Choose the line in the center of the strip



Basic Formulation

- Idea: We want to describe this geometric algorithm mathematically
- Any line/plane/hyperplane can be described by a normal vector w and a distance b
  - The line/plane/hyperplane is all points x where $w \cdot x - b = 0$
  - d is the number of dimensions of our data
  - w is a d dimensional vector
  - b is an intercept term

Basic Formulation

- SVM chooses two parallel planes

$$w \cdot x - b = 1$$

$$w \cdot x - b = -1$$

- NOTE: Notation change from $\{0, 1\}$ to $\{-1, 1\}$ for mathematical convenience
- We need to learn w and b

Basic Formulation

- SVM chooses two parallel planes

$$w \cdot x - b = 1$$

$$w \cdot x - b = -1$$

- We need to learn w and b
- This is a constrained maximization problem
  - Use $||w||$ to denote the l_2 norm of the vector w
  - Can prove distance between them is $\frac{2}{||w||}$
- Where $w \cdot x_i - b \geq 1$ when x_i is "yes"
- Where $w \cdot x_i - b \leq -1$ when x_i is "no"
  - in general, where $y_i(w \cdot x_i - b) \geq 1$

Basic Formulation

- SVM chooses two parallel planes

$$w \cdot x - b = 1$$

$$w \cdot x - b = -1$$

- We need to learn w and b
- This is a constrained maximization problem
- Where $w \cdot x_i - b \geq 1$ when x_i is "yes"
- Where $w \cdot x_i - b \leq -1$ when x_i is "no"
  - in general, where $y_i(w \cdot x_i - b) \geq 1$
- We want to maximize $\frac{2}{||w||}$ subject to the constraint above
  - One class is above the plane
  - The other class is below the plane

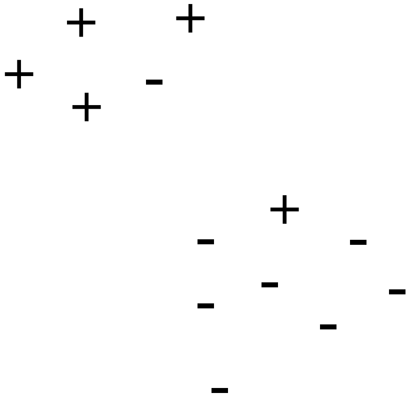
Basic Formulation

- In the end we have...
- Choose w to maximize $\frac{2}{||w||}$ (alt., to minimize $||w||$)
- Subject to $y_i(w \cdot x_i - b) \geq 1$
- That's all a SVM is!
- Easily written as a quadratic program (objective function quadratic)
  - However, quadratic programs are NP-Hard
  - There are solvers out there
  - But, SVMs are a limited version of the quadratic program, so it's not that hard

What does NP-Hard Mean?

- Computational complexity
- COMP 182/382/482 material
- Has to do with how much time it takes to solve the problem
- Non-deterministic polynomial time hardness

One Issue: What If the Data Are Not Linearly Separable?

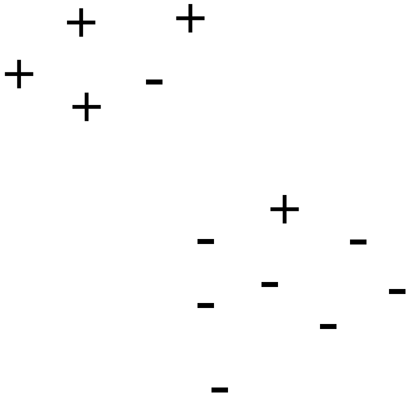


- Then no solution to above problem!
- In other words, our “Subject to” clause may result in ZERO solutions
- Subject to $y_i(w \cdot x_i - b) \geq 1$

? How do we handle this?

One Issue: What If the Data Are Not Linearly Separable?

- Then no solution to above problem!
- In other words, our "Subject to" clause may result in ZERO solutions
- Subject to $y_i(w \cdot x_i - b) \geq 1$
  - Solution: don't require a "hard margin"
  - Allow some error
  - Training points on wrong side of cutting plane get a penalty



“Soft Margin” Formulation

- Go back to the optimization function and add in a “slack” variable
  - Choose w to minimize $||w||^2 + c\sum_i \varepsilon_i$
  - Subject to $y_i(w \cdot x_i - b) \geq 1 - \varepsilon_i$
  - Add a <u>cost</u>, ε, so the learner doesn’t put everything on the wrong side
  - Ideally, ε is 0
  - c is a user supplied variable - it tells the learner how significant a misclassification is
- The “slack” needs to appear in both the objective function and in the constraint
- That’s all a soft-margin SVM is!
- This is a classic approach to handling non-linearly separable data

How To Solve?

- How do we determine w and b?

$$w \cdot x - b = 1$$

$$w \cdot x - b = -1$$

- Choose w to minimize $||w||^2 + c\sum_i \varepsilon_i$
- Subject to $y_i(w \cdot x_i - b) \geq 1 - \varepsilon_i$
- This is a constrained optimization problem

Rewrite as an Unconstrained Problem

- How do we determine w and b?

$$w \cdot x - b = 1$$

$$w \cdot x - b = -1$$

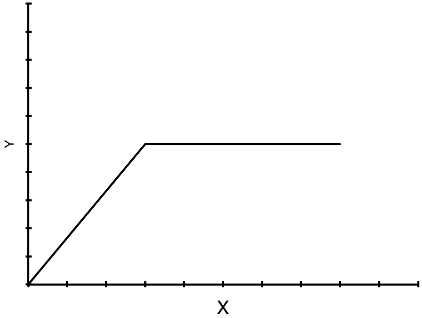
- Choose w to minimize $||w||^2 + c\sum_i \varepsilon_i$
- Subject to $y_i(w \cdot x_i - b) \geq 1 - \varepsilon_i$
- Minimize

$$\frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_i \max(0, 1 - y_i(w \cdot x_i))$$

- where $\lambda = \frac{1}{n \times c}$
- n is the number of points
- c is the cost from the previous slide
- Amenable to gradient descent (one issue: non-smooth max function)

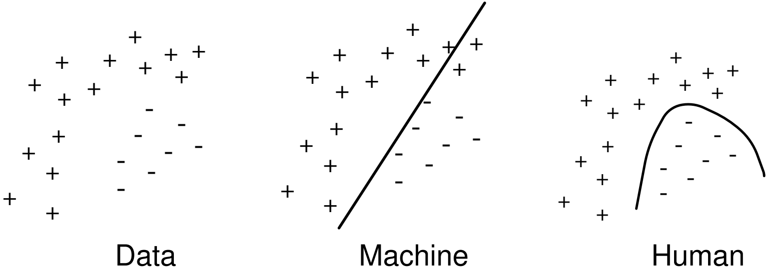
Non-smooth Max Function

- Can’t take the derivative at the max
- In practice it doesn’t matter
- Ignore those cases



The Kernel Trick

- Motivation:
  - SVMs (and logistic regression) are linear
  - But many classification problems are not...
  - Kernel trick idea: map into higher-D, use linear classifier there
  - Not just for SVMs, but closely linked with them

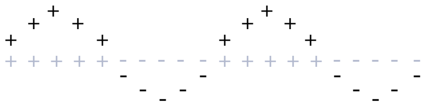


The Kernel Trick

- Biggest motivation for SVMs
- If the number of dimensions $\gg$ number of data points, SVM can build a good classifier
- Consider the case of periodic data, perhaps a photovoltaic sensor

+ + + + + - - - - - + + + + + - - - - -

- Clearly, the data are not linearly separable
- So, map to 2D space by adding a sine value
- Now, the data are linearly separable



Kernels

- Projections that maps the data into a higher dimension space where they are linearly separable
- This approach can be used with any sort of classifier
- SVMs are synonymous with kernel methods

What is a Kernel?

- Math tells us:
  - ANY mapping from vector pairs to matrix entries...
  - ...Is equivalent to embedding vectors in SOME high-D space
  - Where $x'_i \cdot x'_j$ is the matrix entry at i, j
  - As long as we get a positive semi-definite matrix
- This mapping is called a “kernel”

- Given n points, the pairwise dot products $\rightarrow$ a Positive Semi-definite Matrix
- Given n points, the pairwise dot products $\leftarrow$ a Positive Semi-definite Matrix in some space

The Actual "Trick"

- **It's possible to learn an SVM without explicitly performing the mapping**
- Have

$$\frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_i \max(0, 1 - y_i(w \cdot x_i))$$

- where $\lambda = \frac{1}{n \times c}$
- To do this, start with the "dual" formulation of the problem:
  - Maximize

$$\sum_i \alpha_i \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j)$$

  - Subject to

$$0 \leq \alpha_i \leq \frac{1}{\lambda}$$

  - where $\lambda = \frac{1}{n \times c}$
- We want to choose the αs

The Actual "Trick"

- **Learn an SVM without explicitly performing the mapping**
  - Maximize

$$\sum_i \alpha_i \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j)$$

  - Subject to

$$0 \leq \alpha_i \leq \frac{1}{\lambda}$$

  - where $\lambda = \frac{1}{n \times c}$
- We want to choose the αs
- Key observation: each x_i is **only** used as input to dot product
- So no need to explicitly map to high-D
  - As long as have the n by n matrix
  - Where each entry is pairwise dot product in high-D, we're good!
  - We know that the mapping to the high-D space exists

Standard Kernels

- “Polynomial kernel”
  - Replace $(x_i \cdot x_j)$ with $(1 + (x_i \cdot x_j))^d$ for $d > 0$
- “Gaussian kernel”
  - Replace $(x_i \cdot x_j)$ with $\exp(-||x_i - x_j||^2 / (2\sigma^2))$

Kernel Trick Plus/Minus

- Good: can give better classification accuracy
  - Often used in practice for this reason!!
- Bad: often sensitive to kernel parameter(s) (σ in Gaussian kernel, for example)
- Bad: computationally more complex...
  - Dual formulation not easily amenable to gradient descent
  - Means kernels useful mostly for smaller problems

Which Kernel?

- Try different ones and compare

Finally: Log Regression, SVM, or Deep Learning?

- Have small data, well featurized
  - SVM is one of the top options
  - $O(n^2)$ calculations to compute the matrix for the kernel trick
  - Gets unwieldy after about 50K points
- Have "big data", well featurized
  - Regularized Logistic Regression
  - Works well
  - Inexpensive to train
- Regularized Logistic Regression is comparable to SVM **without** the kernel trick
- Have "big data" and/or no features
  - e.g. text, images
  - Deep Learning

Questions?

- What do we know now that we didn't know before?

- How can we use what we learned today?

Questions?

- What do we know now that we didn't know before?
  - SVM is an ML model for classification
  - It can handle non-linearly separable data
  - There's a kernel trick that helps us do this
- How can we use what we learned today?
  - We can use SVM instead of another classification method