CME 252: Support Vector Machines

AJ Friend
ICME, Stanford University

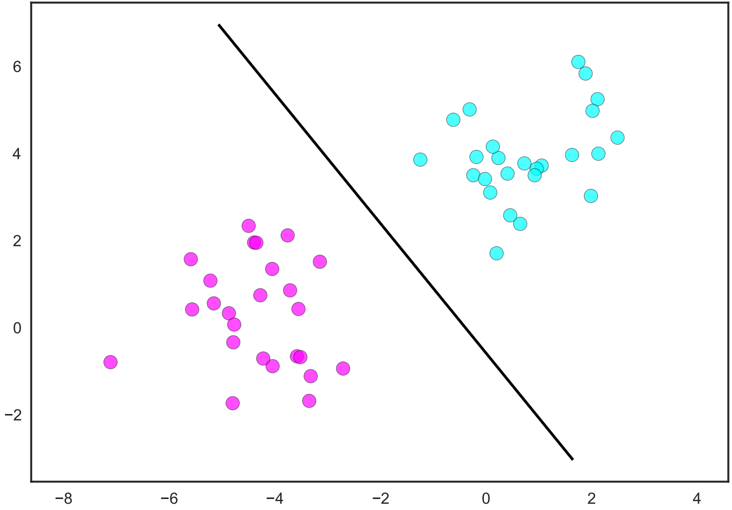
Introduction

Outline

Support Vector Machines

- many related/overlapping names:
  - maximum margin classifier
  - support vector classifier
  - (robust) linear discrimination/classification
  - support vector machine
- I won't always use the right name
- we'll start with:
  - find a hyperplane to separate data points into two classes
  - use hyperplane to classify new (unseen) points

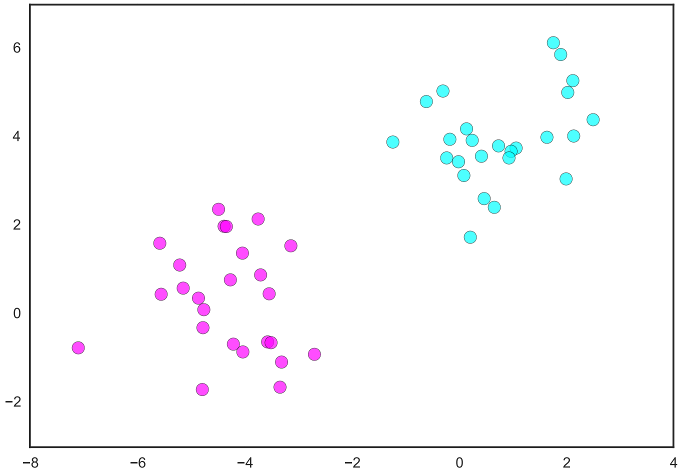
Support Vector Machines



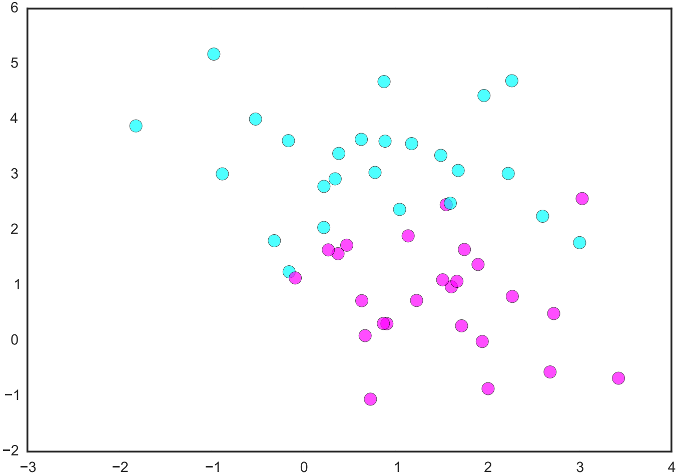
Scenarios

- classify data in increasingly sophisticated scenarios:
  - strictly linearly separable
  - approximately (not strictly) linearly separable
  - approximately non-linearly separable (hyperplanes won't work)

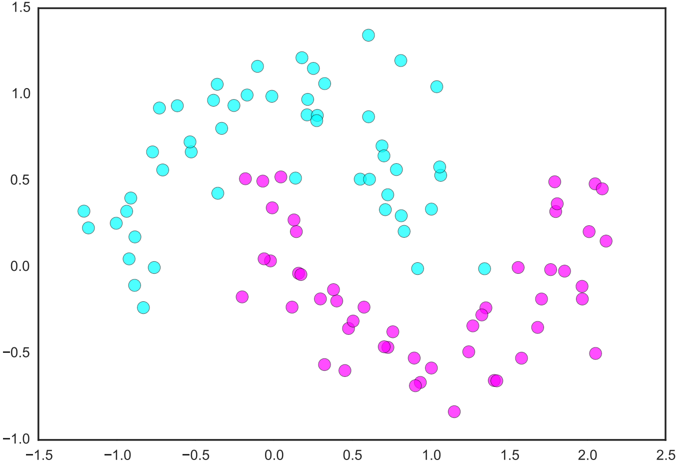
Strictly Linearly Separable Data



Approximately Linearly Separable Data



Approximately Non-linearly Separable



Linearly Separable Problem

Outline

Linearly Separable Problem

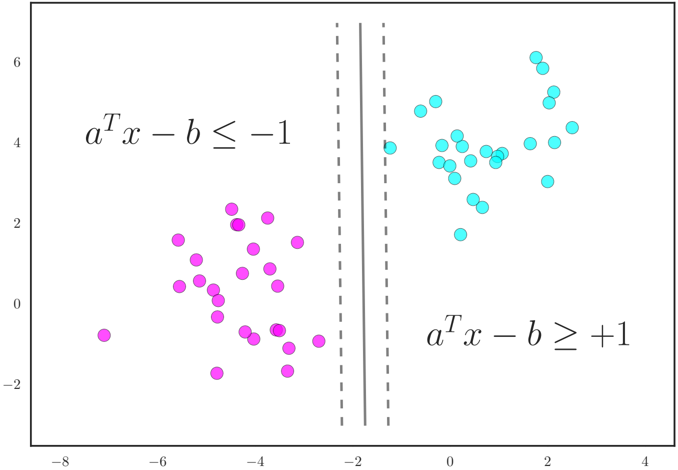
- data: $x_i \in \mathbf{R}^n$ with labels $y_i \in \{+1, -1\}$ for $i = 1, \dots, N$
- assume **strictly** linearly separable
- find hyperplane $\{x \mid a^T x = b\}$ that separates points by label

$$a^T x_i - b > 0 \text{ if } y_i = +1$$
$$a^T x_i - b < 0 \text{ if } y_i = -1$$

- **rescale** a, b so that

$$a^T x_i - b \geq +1 \text{ if } y_i = +1$$
$$a^T x_i - b \leq -1 \text{ if } y_i = -1$$

Linearly Separable Problem



Linearly Separable Problem

- for all i, rewrite constraints as

$$y_i \left(a^T x_i - b\right) \geq 1$$

- get **feasibility** problem

$$\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & y_i \left(a^T x_i - b\right) \geq 1 \text{ for } i = 1, \ldots, N \end{array}$$

with variables $a \in \mathbf{R}^n$, $b \in \mathbf{R}$

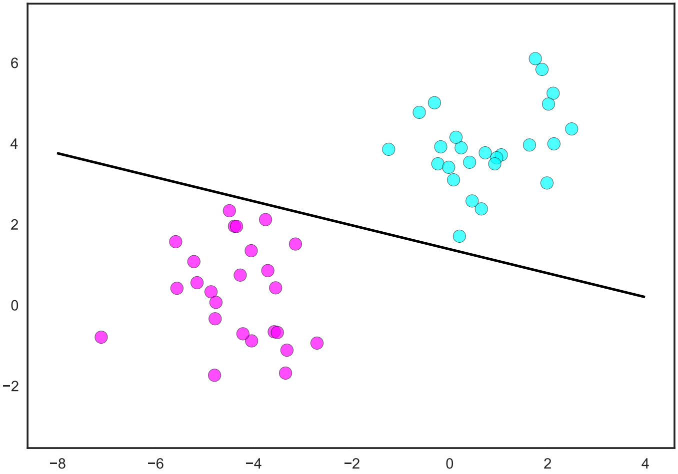
CVXPY for Separable Problem

```
a = Variable(n)
b = Variable()

obj = Minimize(0)
constr = [mul_elemwise(y, X*a - b) >= 1]
Problem(obj, constr).solve()
```

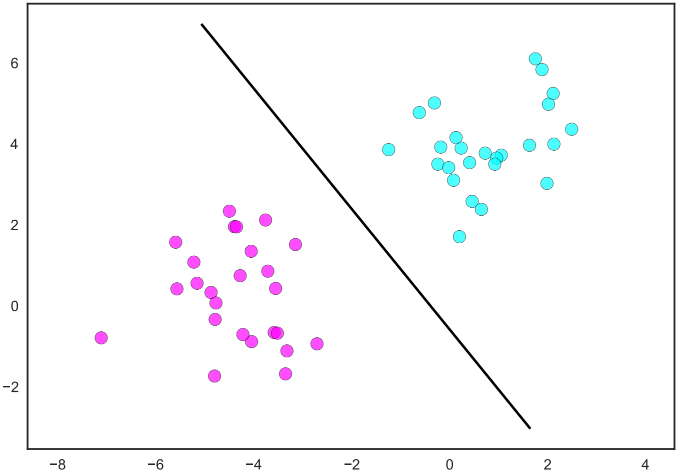
Which Separator?

Outline

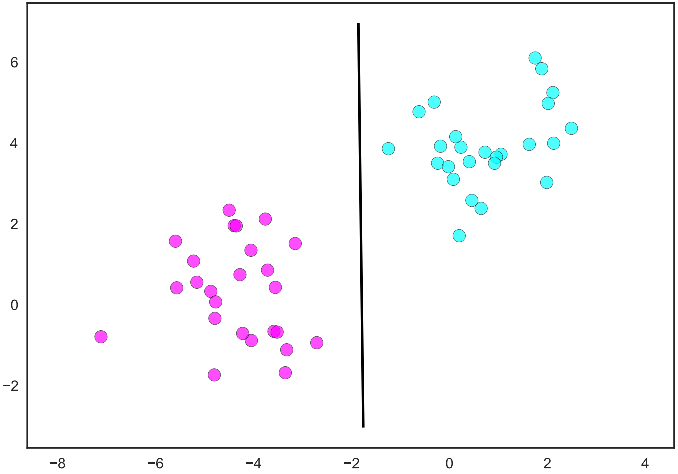
Which Separator?



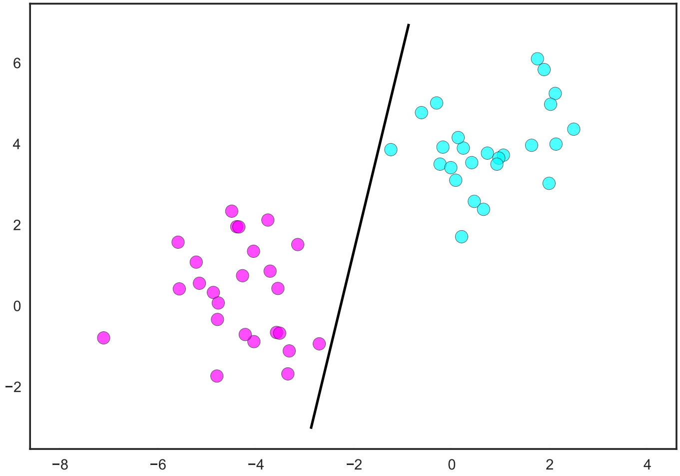
Which Separator?



Which Separator?



Which Separator?



Maximum Margin Classifier

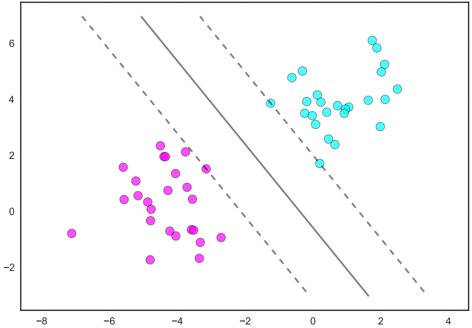
Outline

Maximum Margin Classifier

- infinitely many choices for separating hyperplane
- choose one which maximizes **width** of separating **slab**

$$\{x \mid -1 \leq a^T x - b \leq +1\}$$

- "maximum margin" or "robust linear" classifier



Maximum Margin Classifier

- **margin**, or width of separating slab

$$\{x \mid -1 \leq a^T x - b \leq +1\}$$

  is $2/\|a\|_2$ (via linear algebra)
- suggests optimization problem

$$\begin{array}{ll} \text{maximize} & 2/\|a\|_2 \\ \text{subject to} & y_i \left(a^T x_i - b\right) \geq 1 \text{ for } i = 1, \dots, N \end{array}$$

- but not convex!

Maximum Margin Classifier

- reformulate:

$$\text{maximize } 2/\|a\|_2 \iff \text{minimize } \|a\|_2$$

gives

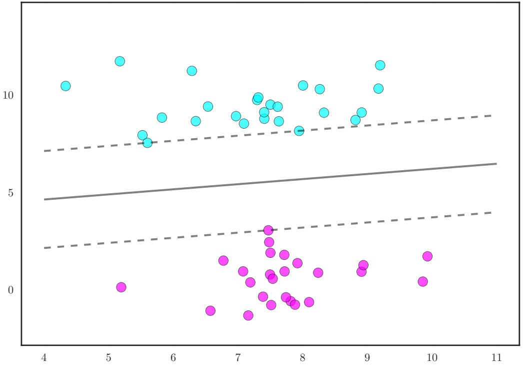
$$\begin{array}{ll} \text{minimize} & \|a\|_2 \\ \text{subject to} & y_i\left(a^T x_i - b\right) \geq 1 \text{ for } i = 1, \ldots, N, \end{array}$$

the **maximum margin classifier** (MMC) problem

CVXPY

```
a = Variable(n)
b = Variable()

obj = Minimize(norm(a))
constr = [mul_elemwise(y, X*a - b) >= 1]
Problem(obj, constr).solve()
```

Example

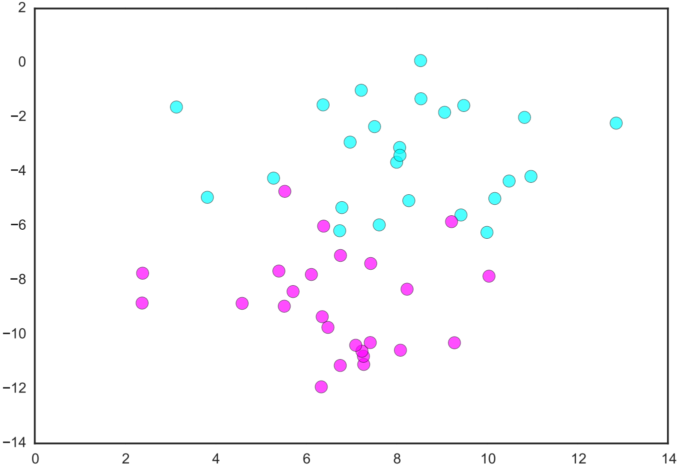


- note that max margin depends on only 3 tangent data points, called **support vectors**
- could throw away remaining data and get same solution

Non-separable Linear Classification

Outline

Non-separable Linear Classification



Non-separable Linear Classification

- no separating hyperplane exists
- try finding linear separator

```
obj = Minimize(0)
constr = [mul_elemwise(y, X*a - b) >= 1]
prob = Problem(obj, constr)
prob.solve()
```

- results in `prob.status == 'infeasible'`

Sparse Violation Classifier

Outline

Sparse Violation Classifier

- idea: "relax" constraints to make problem feasible
- add **slack** variables $u \in \mathbf{R}_+^N$ to allow data points to be on "wrong side" of hyperplane

$$y_i \left(a^T x_i - b\right) \geq 1 - u_i, \quad u_i \geq 0$$

  - $u_i = 0$: x_i on **right** side of hyperplane
  - $0 < u_i < 1$: x_i on **right** side, but **inside slab** $\{x \mid -1 \leq a^T x - b \leq +1\}$
  - $u_i > 1$: x_i on **wrong** side of hyperplane

Sparse Violation Classifier

- u gives measure of how much constraints are violated
- for large u can make **any** data feasible
- want u "small"; minimize its sum

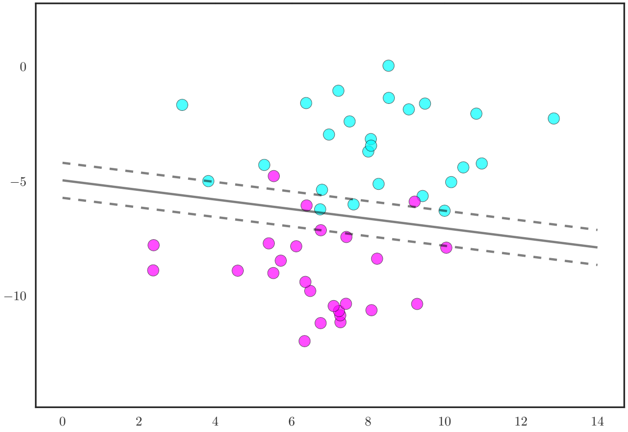
$$
\begin{array}{ll}
\text{minimize} & \mathbf{1}^T u \\
\text{subject to} & y_i \left(a^T x_i - b \right) \geq 1 - u_i \text{ for } i = 1, \ldots, N \\
& u \geq 0
\end{array}
$$

- I'll call it **sparse violation classifier** (SpVC)
- $\mathbf{1}^T u = \|u\|_1$, since $u \geq 0$; good **heuristic** for separator with few (sparse) violations

CVXPY

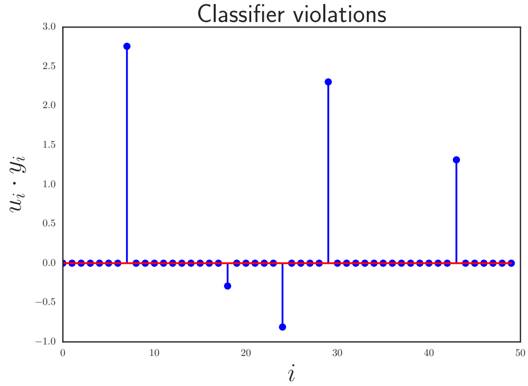
```
a = Variable(n)
b = Variable()
u = Variable(N)

obj = Minimize(sum_entries(u))
constr = [mul_elemwise(y, X*a - b) >= 1 - u, u >= 0]
Problem(obj, constr).solve()
```

Example



- solution depends only on points inside of, tangent to, or on wrong side of slab

Example



- "$+$" class has 3 misclassified points
- "$-$" class has 2 correctly classified, but inside slab

Support Vector Classifier

Outline

Support Vector Classifier

- idea: combine aspects of last two classifiers
  - sparse violations of SpVC
  - robustness of large separating slab in MMC
- optimize both:

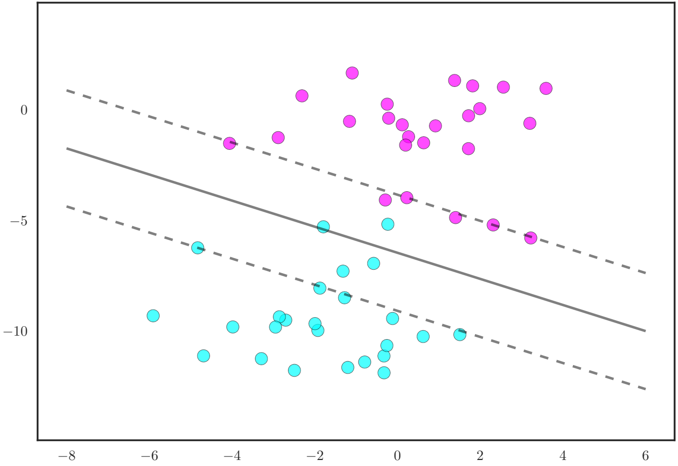
$$
\begin{array}{ll}
\text{minimize} & \|a\|_2 + \rho \mathbf{1}^T u \\
\text{subject to} & y_i \left(a^T x_i - b \right) \geq 1 - u_i \text{ for } i = 1, \ldots, N \\
& u \geq 0
\end{array}
$$

- $\rho > 0$ trades-off between margin $2/\|a\|_2$ and classification violations $\mathbf{1}^T u$ (multi-objective optimization)
- **support vector classifier** (SVC)

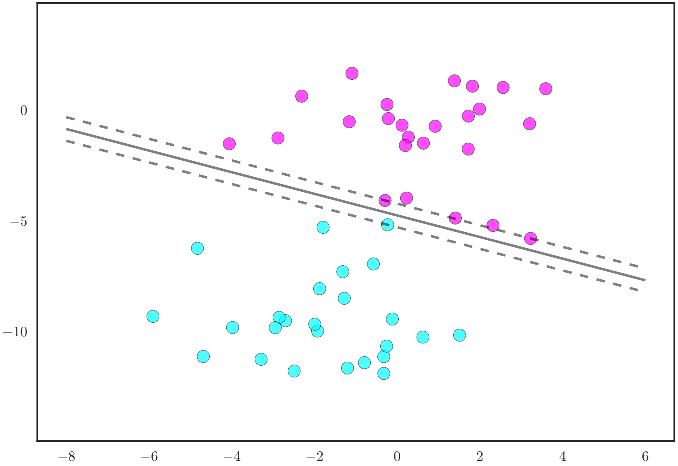
CVXPY

```
a = Variable(n)
b = Variable()
u = Variable(N)
rho = .1

obj = Minimize(norm(a) + rho*sum_entries(u))
constr = [mul_elemwise(y, X*a - b) >= 1 - u, u >= 0]
Problem(obj, constr).solve()
```

Example with $\rho = .1$



Example with $\rho = 10$



Loss Functions

Outline

Hinge Loss

- in SpVC, it follows from $y_i\left(a^T x_i - b\right) \geq 1 - u_i$, $u_i \geq 0$, that

$$u_i = \begin{cases} 0 & y_i\left(a^T x_i - b\right) \geq 1 \\ 1 - y_i\left(a^T x_i - b\right) & y_i\left(a^T x_i - b\right) < 1 \end{cases}$$

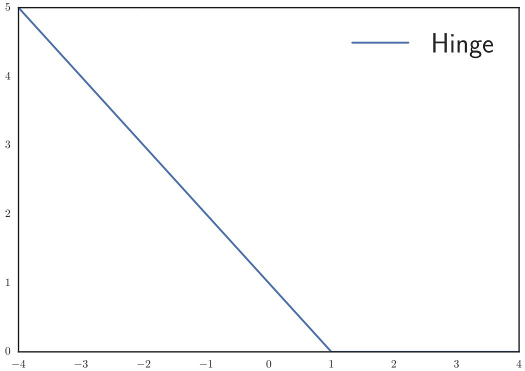
- rewrite as $u_i = \ell_h\left[y_i\left(a^T x_i - b\right)\right]$, where

$$\ell_h(z) = \begin{cases} 0 & z \geq 1 \\ 1 - z & z < 1 \end{cases}$$

is the **hinge loss** function, equivalently: $\max(0, 1 - z)$ or $(1 - z)_+$

Hinge Loss

- $u_i = \ell_h \left[y_i \left(a^T x_i - b \right) \right]$
- no penality if $y_i \left(a^T x_i - b \right) \geq 1$
- linear penalty otherwise



Hinge Loss SpVC

- note that ℓ_h is convex, so we can rewrite SpVC as the **equivalent problem**

$$\text{minimize} \quad \sum_{i=1}^{N} \ell_h \left[y_i \left(a^T x_i - b \right) \right]$$

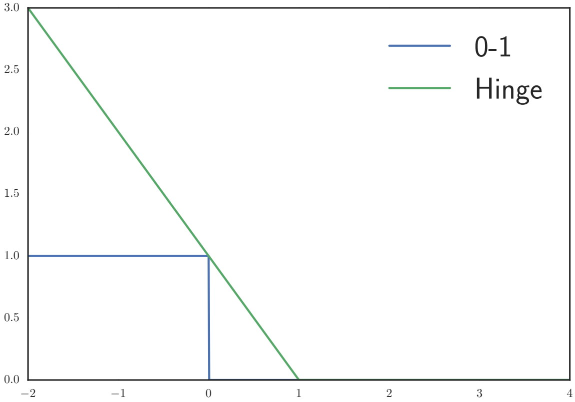
- unconstrained (non-differentiable) convex problem
- in CVXPY:

```
def hinge(z):
    return pos(1-z)

r = mul_elemwise(y, X*a - b)
obj = Minimize(sum_entries(hinge(r)))
Problem(obj).solve()
```

Why Hinge Loss?

- "0-1" loss: $\ell_{0-1}(z) = \begin{cases} 0 & z \geq 0 \\ 1 & z < 0 \end{cases}$
- can't solve nonconvex, combinatorial problem to minimize (discrete) number of violations with
$$\text{minimize} \quad \sum_{i=1}^{N} \ell_{0-1}\left[y_i\left(a^T x_i - b\right)\right]$$
- hinge loss gives a **convex** approximation to 0-1 loss

Why Hinge Loss?



- but not the **only** convex approximation

Hinge Loss SVC

- can rewrite SVC as the **unconstrained** problem

$$\text{minimize} \quad \|a\|_2 + \rho \sum_{i=1}^{N} \ell_h \left[y_i \left(a^T x_i - b \right) \right]$$

- completely **equivalent** to the SVC formulation from before
- common form for classification problems:

$$\text{minimize} \quad r(a) + \rho \sum_{i=1}^{N} \ell \left[y_i \left(a^T x_i - b \right) \right]$$

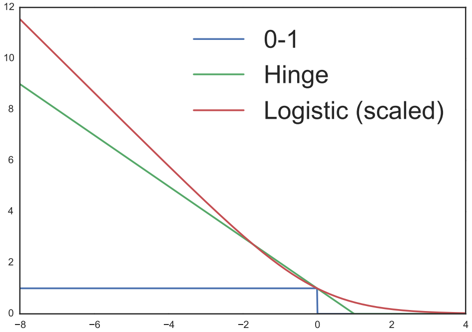
  - ℓ is a **loss function** (fit to data)
  - r is a **regularizer** (prior on parameters)
- **mix and match** regularizers and loss functions for different types of classification

Logistic Loss

- **logistic loss** is an alternative to hinge loss:

$$\ell_L(z) = \log(1 + \exp(-z))$$

- convex, but not immediately obvious (2nd derivative test)



Logistic Regression

- get classic **logistic regression** with

$$\text{minimize} \quad r(a) + \rho \sum_{i=1}^{N} \ell \left[y_i \left(a^T x_i - b \right) \right]$$

  when:

  - $r(a) \equiv 0$
  - $\ell(z) = \ell_L(z)$

- nice probabilistic interpretation
- **regularized** logistic regression when $r(a)$ is $\|a\|_2$ or $\|a\|_1$ (sparsity)

Logistic Loss in CVXPY

- $\ell_L(z) = \log(1 + \exp(-z))$ doesn't follow convex composition rules
- to represent in CVXPY, use existing convex atom **log-sum-exp**:

$$f(x) = \log\left(e^{x_1} + \cdots + e^{x_n}\right)$$

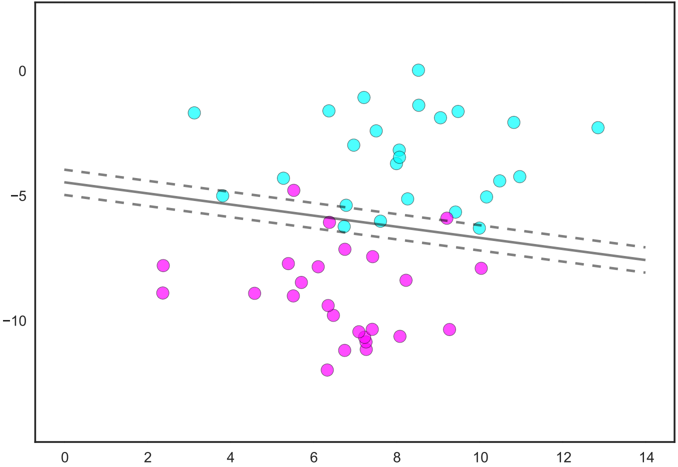
- convexity follows from Hessian argument
- $$\ell_L(z) = \log(1 + \exp(-z)) = \log\left(e^0 + e^{-z}\right)$$

```
def logistic(x):
    elems = []
    for xi in x:
        elems += [cvx.log_sum_exp(cvx.vstack(0, xi))]

    return cvx.vstack(*elems)
```

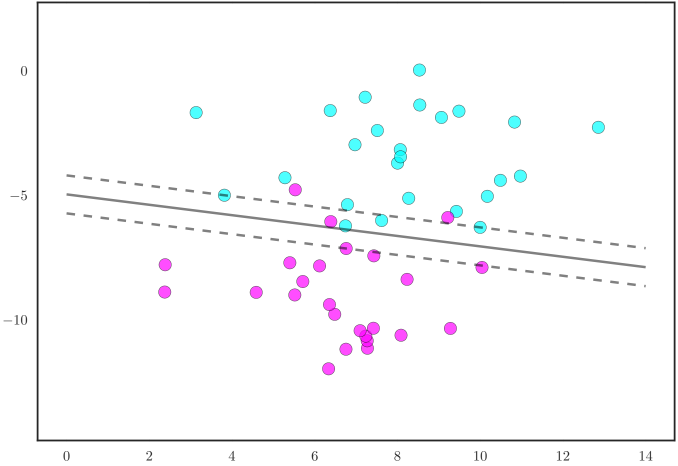
Logistic Regression in CVXPY

```
a = Variable(n)
b = Variable()

r = mul_elemwise(y, X*a - b)
obj = Minimize(sum_entries(logistic(r)))
Problem(obj).solve()
```

Logistic Regression in CVXPY



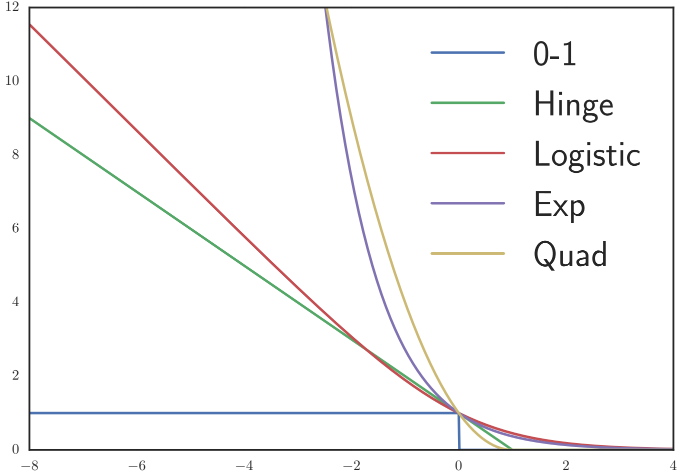
SpVC (for comparison)



Other Loss Functions

- many loss functions are available to modeler
- hard loss: $\ell_{\text{hard}}(z) = \begin{cases} 0 & z \geq 0 \\ +\infty & z < 0 \end{cases}$
- exponential loss: $\ell_{\exp}(z) = \exp(-z)$
- quadratic loss: $\ell_2(z) = (1-z)_+^2$

Other Loss Functions



Unified Models

- many classification models fall into the form

$$\text{minimize} \quad r(a) + \rho \sum_{i=1}^{N} \ell \left[y_i \left(a^T x_i - b \right) \right]$$

- unified way to think about many of these models

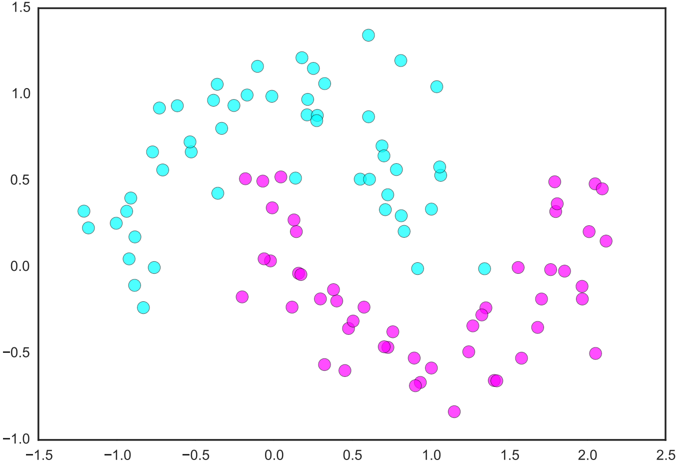
Unified Models

- linear separator feasibility problem: ℓ_{hard}, $r \equiv 0$
- MMC: ℓ_{hard}, $r(a) = \|a\|_2$
- SpVC: ℓ_h, $r \equiv 0$
- SVC: ℓ_h, $r(a) = \|a\|_2$
- Logistic regression: ℓ_L, $r \equiv 0$
- "boosting": ℓ_{exp}, $r \equiv 0$
- many other options for modeling
  - loss for max violation instead of sum
  - sparse a for feature selection
  - one-sided Huber loss for outliers?

Nonlinear Separators

Outline

Nonlinear Separators



Nonlinear Separators

- don't expect a linear separator to work
- to get nonlinear separators, we need to generalize our classification function

$$f(x) = a^T x + b$$

- consider polynomials of $x \in \mathbf{R}^n$ of degree d:

$$f(x) = \sum_{j_1+\cdots+j_n \le d} a_{j_1\cdots j_n} x_1^{j_1} \cdots x_n^{j_n}$$

- a little messy, but **still linear** in decision variable a

Support Vector Machines

- follow the same setup as before: data x_i with labels $y_i \in \{+1, -1\}$
- positive and negative examples on opposite "sides" of the classification function

$$f_a(x_i) > 0 \text{ if } y_i = +1$$
$$f_a(x_i) < 0 \text{ if } y_i = -1$$

which we simplify to

$$y_i f_a(x_i) > 0$$

Support Vector Machines

- **quantify** our dislike of violations with **any** loss function we like

$$\ell\left[y_i f_a(x_i)\right]$$

- add regularization to get the same general set up as before:

$$\text{minimize}_a \quad r(a) + \rho \sum_{i=1}^{N} \ell\left[y_i f_a(x)\right]$$

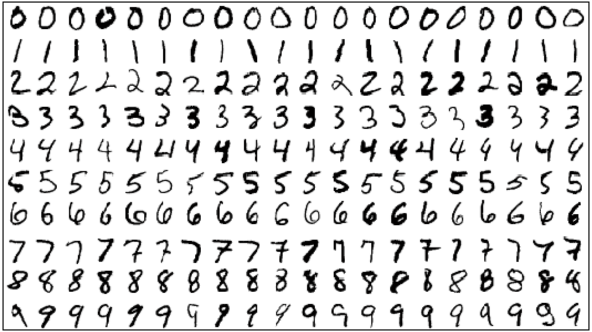
- not really different from linear classification problem before
- we've just expanded the number of **features** for each point by considering polynomials
- **support vector machine** is usually ℓ_h, $r(a) = \|a\|_2$

Multiclass SVM

Outline

Multiclass SVM

- what if you have more than 2 labels?
- example: handwritten digit classification



Multiclass SVM Approaches

- one-vs-one
  - for each pair of classes, train an SVM
  - $\frac{K(K-1)}{2}$ problems!
  - for a new observation, choose most frequently predicted label among all SVMs
- one-vs-all
  - train K SVMs: one single class vs. all others grouped
  - for new observation, choose label furthest away from separating hyperplane

Single-model Multiclass SVM

- TODO