Statistical Machine Learning

Christoph Lampert

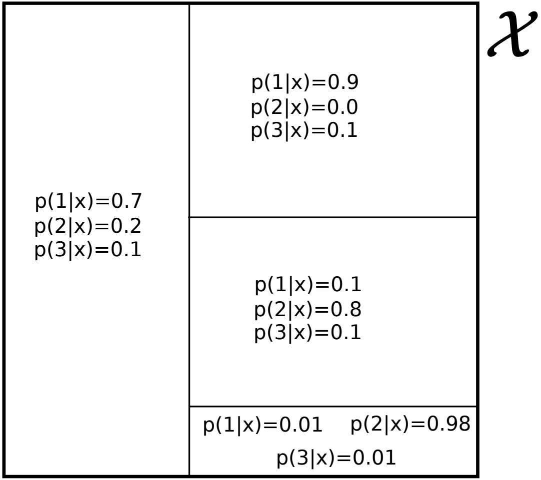
IST AUSTRIA

Institute of Science and Technology

Spring Semester 2013/2014 // Lecture 3

Nonparametric Discriminative Model

Idea: split $\mathcal{X}$ into regions, for each region store an estimate $\hat{p}(y|x)$.



Nonparametric Discriminative Model

Idea: split $\mathcal{X}$ into regions, for each region store an estimate $\hat{p}(y|x)$.

For example, using a **decision tree**:

- training: build a tree
- prediction: for new example x, find its leaf
- output $\hat{p}(y|x) = \frac{n_y}{n}$, where
  - n is the number of examples in the leaf,
  - n_y is the number of example of label y in the leaf.

Nonparametric Discriminative Model

Idea: split $\mathcal{X}$ into regions, for each region store an estimate $\hat{p}(y|x)$.

For example, using a **decision tree**:

- training: build a tree
- prediction: for new example x, find its leaf
- output $\hat{p}(y|x) = \frac{n_y}{n}$, where
  - n is the number of examples in the leaf,
  - n_y is the number of example of label y in the leaf.

Note: prediction rule

$$c(x) = \operatorname*{argmax}_y \hat{p}(y|x)$$

is predicts the most frequent label in each leaf (same as in first lecture).

Parametric Discriminative Model: Logistic Regression

Setting. We assume $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$.

Definition (Logistic Regression (LogReg) Model)

Modeling

$$\hat{p}(y|x; w) = \frac{1}{1 + \exp(-y\langle w, x\rangle)},$$

with parameter vector $w \in \mathbb{R}^d$ is called a *logistic regression* model.

Parametric Discriminative Model: Logistic Regression

Setting. We assume $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$.

Definition (Logistic Regression (LogReg) Model)

Modeling

$$\hat{p}(y|x; w) = \frac{1}{1 + \exp(-y\langle w, x\rangle)},$$

with parameter vector $w \in \mathbb{R}^d$ is called a *logistic regression* model.

Lemma

$\hat{p}(y|x; w)$ *is a well defined probability density w.r.t.* y *for any* $w \in \mathbb{R}^d$*.*

Proof. elementary.

How to set the weight vector w (based on $\mathcal{D}$)

Logistic Regression Training

Given a training set $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\}$, *logistic regression training* sets the free parameter vector as

$$w_{LR} \;=\; \operatorname*{argmin}_{w \in \mathbb{R}^d} \; \sum_{i=1}^{n} \log\left(1 + \exp(-y^i \langle w, x^i \rangle)\right)$$

Lemma (Conditional Likelihood Maximization)

w_{LR} *from Logistic Regression training maximizes the conditional data likelihood w.r.t. the LogReg model,*

$$w_{LR} \;=\; \operatorname*{argmax}_{w \in \mathbb{R}^d} \; \hat{p}(y^1, \dots, y^n | x^1, \dots, x^n, w)$$

Proof.

Maximizing

$$\hat{p}(\mathcal{D}^Y|\mathcal{D}^X, w) \overset{i.i.d.}{=} \prod_{i=1}^{n} \hat{p}(y^i|x^i, w)$$

is equivalent to minimizing its negative logarithm

$$\begin{aligned} -\log\ \hat{p}(\mathcal{D}^Y|\mathcal{D}^X, w) &= -\log \prod_{i=1}^{n} \hat{p}(y^i|x^i, w) = -\sum_{i=1}^{n} \log \hat{p}(y^i|x^i, w) \\ &= -\sum_{i=1}^{n} \log \frac{1}{1+\exp(-y^i\langle w, x^i\rangle)}, \\ &= -\sum_{i=1}^{n} [\log 1 - \log(1+\exp(-y^i\langle w, x^i\rangle))], \\ &= \sum_{i=1}^{n} \log(1+\exp(-y^i\langle w, x^i\rangle)). \end{aligned}$$

$\square$

Alternative Explanation

Definition (Kullback-Leibler (KL) divergence)

Let p and q be two probability distributions (for discrete $\mathcal{Z}$) or probabilitiy densities with respect to a measure dλ (for continuous $\mathcal{Z}$). The **Kullback-Leibler (KL)-divergence** between p and q is defined as

$$\mathrm{KL}(p \,\|\, q) = \sum_{z \in \mathcal{Z}} p(z) \log \frac{p(z)}{q(z)}, \quad \text{or} \quad \mathrm{KL}(p \,\|\, q) = \int_{z \in \mathcal{Z}} p(z) \log \frac{p(z)}{q(z)} \, \mathsf{d}\lambda(\mathsf{z}),$$

(with convention $0 \log 0 = 0$, and $a \log \frac{a}{0} = \infty$ for $a > 0$).

Alternative Explanation

Definition (Kullback-Leibler (KL) divergence)

Let p and q be two probability distributions (for discrete $\mathcal{Z}$) or probabilitiy densities with respect to a measure dλ (for continuous $\mathcal{Z}$). The **Kullback-Leibler (KL)-divergence** between p and q is defined as

$$\mathrm{KL}(p \,\|\, q) = \sum_{z \in \mathcal{Z}} p(z) \log \frac{p(z)}{q(z)}, \quad \text{or} \quad \mathrm{KL}(p \,\|\, q) = \int_{z \in \mathcal{Z}} p(z) \log \frac{p(z)}{q(z)} \, \mathsf{d}\lambda(\mathsf{z}),$$

(with convention $0 \log 0 = 0$, and $a \log \frac{a}{0} = \infty$ for $a > 0$).

KL is a similarity measure between probability distributions. It fulfills

$$0 \leq KL(p \,\|\, q) \leq \infty, \qquad \text{and} \qquad KL(p \,\|\, q) = 0 \;\Leftrightarrow\; p = q.$$

However, KL is **not a metric**.

- it is in general not symmetric, $KL(q \,\|\, p) \neq KL(p \,\|\, q)$,
- it does not fulfill the triangle inequality.

Alternative Explanation of Logistic Regression Training

Definition (Expected Kullback-Leibler (KL) divergence)

Let $p(x, y)$ be a probability distribution over $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and let $\hat{p}(y|x)$ be an approximation of $p(y|x)$.
We measure the approximation quality by the **expected KL-divergence between** p **and** q over all $x \in \mathcal{X}$:

$$\mathrm{KL}_{\textit{exp}}(p \,\|\, q) = \mathbb{E}_{x \sim p(x)} \{ \, \mathrm{KL}(p(\cdot|x) \| q(\cdot|x)) \, \}$$

Theorem

The parameter w_{LR} obtained by logistic regression training approximately minimizes the KL divergence between $\hat{p}(y|x; w)$ and $p(y|x)$.

Proof.

We show how maximimzing the conditional likelihood relates to KL_{exp}:

$$\begin{aligned}\mathrm{KL}_{exp}(p\|\hat{p}) &= \mathbb{E}_{x\sim p(x)} \sum_{y\in\mathcal{Y}} p(y|x) \log \frac{p(y|x)}{\hat{p}(y|x,w)} \\ &= \underbrace{\mathbb{E}_{(x,y)\sim p(x,y)} \log p(y|x)}_{\text{indep. of } w} - \mathbb{E}_{(x,y)\sim p(x,y)} \log \hat{p}(y|x,w)\end{aligned}$$

We can't maximize $\mathbb{E}_{(x,y)\sim p(x,y)} \log \hat{p}(y|x,w)$ directly, because $p(x,y)$ is unknown. But we can maximimize its empirical estimate based on $\mathcal{D}$:

$$\mathbb{E}_{(x,y)\sim p(x,y)} \log \hat{p}(y|x,w) \approx \underbrace{\sum_{(x^i,y^i)\in\mathcal{D}} \log \hat{p}(y^i|x^i,w)}_{\log \text{ of conditional data likelihood}} .$$

The approximation will get better the more data we have. $\square$

Solving Logistic Regression numerically – Optimization I

Theorem

Logistic Regression training,

$$w_{LR} \;=\; \operatorname*{argmin}_{w\in\mathbb{R}^d}\; \mathcal{L}(w) \quad \textit{for} \quad \mathcal{L}(w) = \sum_{i=1}^{n} \log\left(1 + \exp(-y^i\langle w, x^i\rangle)\right),$$

is a C^∞-smooth, unconstrained, convex optimization problem.

Proof.

1. it's an optimization problem,
2. it's unconstrained,
3. it's smooth (the objective function is C^∞ differentiable),
4. remains to show: the objective function is a *convex* function.
   Since $\mathcal{L}$ is smooth, it's enough to show that its *Hessian matrix* (the matrix of 2nd partial derivatives) is everywhere *positive definite.*

We compute first the gradient and then the Hessian of

$$\mathcal{L}(w) = \sum_{i=1}^{n} \log(1 + \exp(-y^i \langle w, x^i \rangle)).$$

$$\nabla_w \; \mathcal{L}(w) = \sum_{i=1}^{n} \nabla \log(1 + \exp(-y^i \langle w, x^i \rangle)).$$

use the chain rule, $\nabla f(g(w)) = \frac{d}{dt}(g(w) \nabla g(w)$, and $\frac{d}{dt} \log(t) = \frac{1}{t}$

$$= \sum_{i=1}^{n} \frac{\nabla [1 + \exp(-y^i \langle w, x^i \rangle]}{1 + \exp(-y^i \langle w, x^i \rangle}$$

$$= \sum_{i=1}^{n} \underbrace{\frac{\exp(-y^i \langle w, x^i \rangle)}{1 + \exp(-y^i \langle w, x^i \rangle)}}_{=\hat{p}(-y^i | x^i, w)} \nabla(-y^i \langle w, x^i \rangle)$$

use the chain rule again, $\frac{d}{dt} \exp(t) = \exp(t)$, and $\nabla_w \langle w, x^i \rangle = x^i$

$$= -\sum_{i=1}^{n} [\hat{p}(-y^i | x^i, w)] \; y^i x^i$$

$$H_w\mathcal{L}(w) = \nabla\nabla^\top\mathcal{L}(w) = -\sum_{i=1}^{n}[\nabla\hat{p}(-y^i|x^i,w)]\ y^i x^i$$

$$\nabla\hat{p}(-y^i|x^i,w) = \nabla\frac{1}{1+\exp(y^i\langle w,x^i\rangle)}$$

$$= -\frac{\nabla[1+\exp(y^i\langle w,x^i\rangle)]}{[1+\exp(y^i\langle w,x^i\rangle)]^2}$$

use quotient rule, $\nabla\frac{1}{f(w)} = -\frac{\nabla f(w)}{f^2(w)}$, and chain rule,

$$= -\frac{\exp(y^i\langle w,x^i\rangle)}{[1+\exp(y^i\langle w,x^i\rangle)]^2}\nabla y^i\langle w,x^i\rangle$$

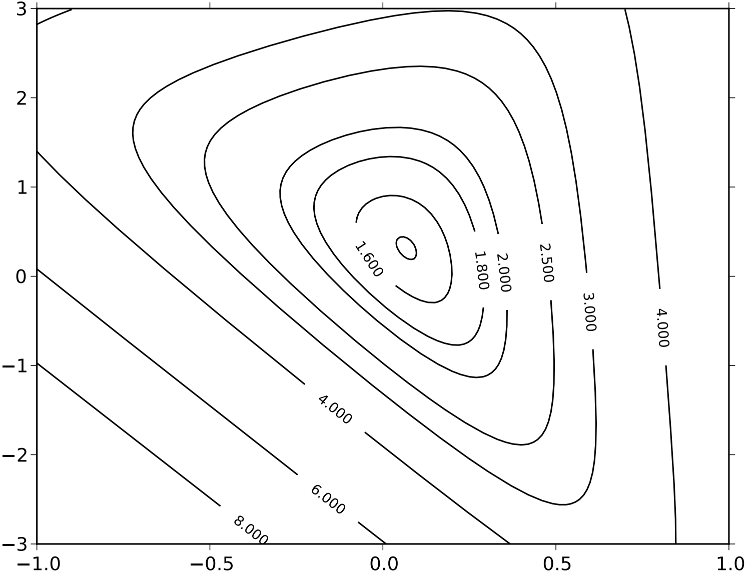
$$= -(\hat{p}(-y^i|x^i))\hat{p}(y^i|x^i,w)y^i x^i$$

insert into above expression for $H_w\mathcal{L}(w)$

$$H = \sum_{i=1}^{n}\underbrace{\hat{p}(-y^i|x^i)\hat{p}(y^i|x^i,w)}_{>0}\ \underbrace{x^i x^{i\top}}_{sym.pos.def.}$$

A positively weighted linear combination of pos.def. matrices is pos.def.

Example plot: LogReg objective for three examples in $\mathbb{R}^2$



Numeric Optimization

Convex optimization is a well understood field. We can use, e.g., *gradient descent* will converge to the globally optimal solution!

Steepest Descent Minimization with Line Search

input $\epsilon > 0$ tolerance (for stopping criterion)

$w \leftarrow 0$
repeat
  $v \leftarrow -\nabla_w \mathcal{L}(w)$ {descent direction}
  $\eta \leftarrow \operatorname{argmin}_{\eta > 0} \mathcal{L}(w + \eta v)$ {1D line search}
  $w \leftarrow w + \eta d$
until $\|v\| < \epsilon$

output $w \in \mathbb{R}^d$ learned weight vector

Faster conference from methods that use second-order information, e.g., *conjugate gradients* or *(L-)BFGS* $\rightarrow$ *convex optimization lecture*

Binary classification with a LogReg Models

A discriminative probability model, $\hat{p}(y|x)$, is enough to make decisions:

$$c(x) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\, \hat{p}(y|x) \quad \text{or} \quad c(x) = \underset{y\in\mathcal{Y}}{\operatorname{argmin}}\, \mathbb{E}_{\bar{y}\sim\hat{p}(y|x)} \ell(\bar{y}, y).$$

For Logistic Regression, this is particularly simple:

Lemma

The LogReg classification rule for $0/1$-loss is

$$c(x) = \operatorname{sign} \langle w, x\rangle.$$

For a loss function $\ell = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ the rule is

$$c_\ell(x) = \operatorname{sign}[\ \langle w, x\rangle + \log \frac{c-d}{b-a}\ ],$$

In particular, ***the decision boundaries is linear (or affine).***

Proof. Elementary, since $\log \frac{\hat{p}(+1|x;w)}{p(-1|x;w)} = \langle w, x\rangle$

Multiclass Logistic Regression

For $\mathcal{Y} = \{1, \dots, M\}$, we can do two things:

- Parametrize $\hat{p}(y|x;\vec{w})$ using $M-1$ vectors, $w_1, \dots, w_{M-1} \in \mathbb{R}^d$, as

$$\hat{p}(y|x,w) = \frac{\exp(\langle w_y, x\rangle)}{1 + \sum_{j=1}^{M-1} \exp(\langle w_j, x\rangle)} \quad \text{for } y = 1, \dots, M-1,$$
$$\hat{p}(M|x,w) = \frac{1}{1 + \sum_{j=1}^{M-1} \exp(\langle w_j, x\rangle)}.$$

- Parametrize $\hat{p}(y|x;\vec{w})$ using M vectors, $w_1, \dots, w_M \in \mathbb{R}^d$, as

$$\hat{p}(y|x,w) = \frac{\exp(\langle w_y, x\rangle)}{\sum_{j=1}^{M} \exp(\langle w_j, x\rangle)} \quad \text{for } y = 1, \dots, M,$$

Second is more popular, since it's easier to implement and analyze.

Decision boundaries are still *piecewise linear*, $c(x) = \operatorname{argmax}_y \langle w_y, x\rangle$.

Summary: Discriminative Models

Discriminative models treats the input data, x, as fixed and only model the distribution of the output labels $p(y|x)$.

Discriminative models, in particular LogReg, are popular, because

- they often need less training data than generative models,
- they provide an estimate of the uncertainty of a decision $p(c(x)|x)$,
- training them is often efficient,
  e.g. Yahoo trains LogReg models routinely from billions of examples.

But: they also have drawbacks

- often $\hat{p}_{LR}(y|x) \not\to p(y|x)$, even for $n \to \infty$,
- they usually are good for *prediction*, but they do not reflect the actual *mechanism*.

Note: there are much more complex discriminative models than LogReg, e.g. Conditional Random Fields (maybe later).

Maximum Margin Classifiers

$\boxed{\textbf{Let's use } \mathcal{D} \textbf{ to estimate a classifier } c : \mathcal{X} \to \mathcal{Y} \textbf{ directly.}}$

Maximum Margin Classifiers

Let's use $\mathcal{D}$ to estimate a classifier $c : \mathcal{X} \to \mathcal{Y}$ directly.

For a start, we fix

- $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$,
- $\mathcal{Y} = \{+1, -1\}$,
- we look for classifiers with linear decision boundary.

Several of the classifiers we saw had *linear* decision boundaries:

- Perceptron
- Generative classifiers for Gaussian class-conditional densities with shared covariance matrix
- Logistic Regression

What's the **best linear classifier**?

Linear classifiers

Definition

Let

$$\mathcal{F} = \{ \ f : \mathbb{R}^d \to \{\pm 1\} \text{ with } f(x) = b + a_1 x_1 + \cdots + a_d x_d = b + \langle w, x \rangle \ \}$$

be the set of linear (affine) function from $\mathbb{R}^d \to \mathbb{R}$.

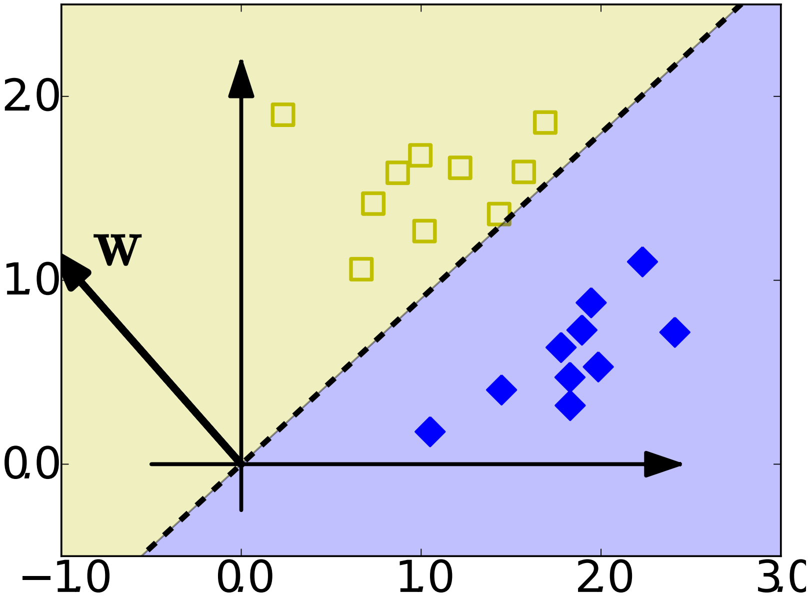
A classifier $g : \mathcal{X} \to \mathcal{Y}$ is called **linear**, if it can be written as

$$g(x) = \operatorname{sign} f(x)$$

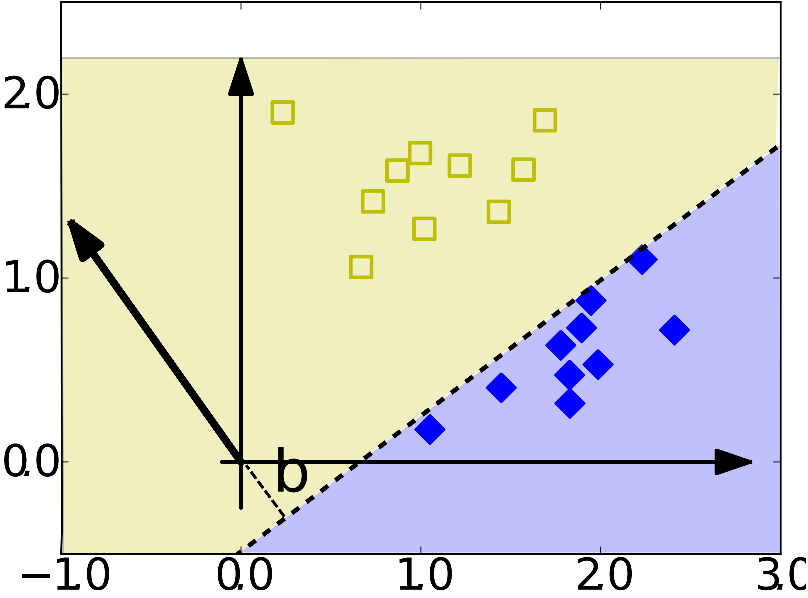
for some $f \in \mathcal{F}$.

We write $\mathcal{G}$ for the set of all linear classifiers.

A linear classifier, $g(x) = \mathrm{sign}\langle w, x\rangle$, with $b = 0$



A linear classifier $g(x) = \mathrm{sign}(\langle w, x\rangle + b)$, with $b > 0$



Definition

We call a classifier, g, **correct** (for a training set $\mathcal{D}$), if it predicts the correct labels for all training examples:

$$g(x^i) = y^i \qquad \text{for } i = 1, \ldots, n.$$

Example (Perceptron)

- if the *Perceptron* converges, the result is an *correct* classifier.
- any classifier with zero training error is *correct*.

Linear classifiers

Definition

We call a classifier, g, **correct** (for a training set $\mathcal{D}$), if it predicts the correct labels for all training examples:

$$g(x^i) = y^i \qquad \text{for } i = 1, \dots, n.$$

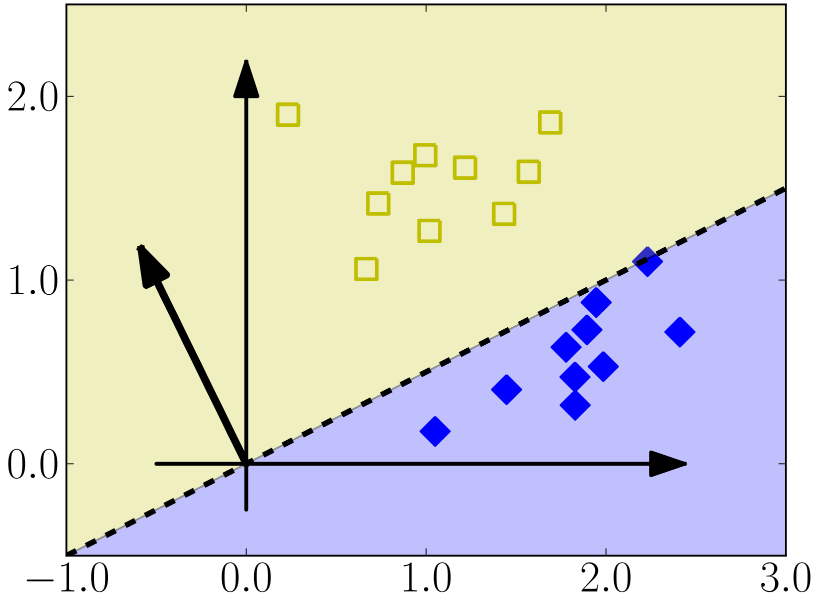
Example (Perceptron)

- if the *Perceptron* converges, the result is an *correct* classifier.
- any classifier with zero training error is *correct*.

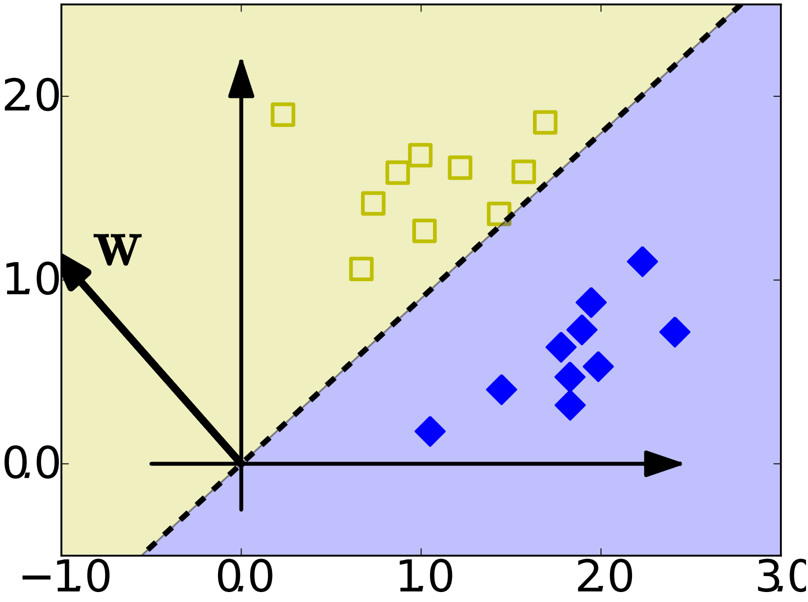
Definition (Linear Separability)

A training set $\mathcal{D}$ is called **linearly separable**, if it allows a correct linear classifier (i.e. the classes can be separated by a hyperplane).

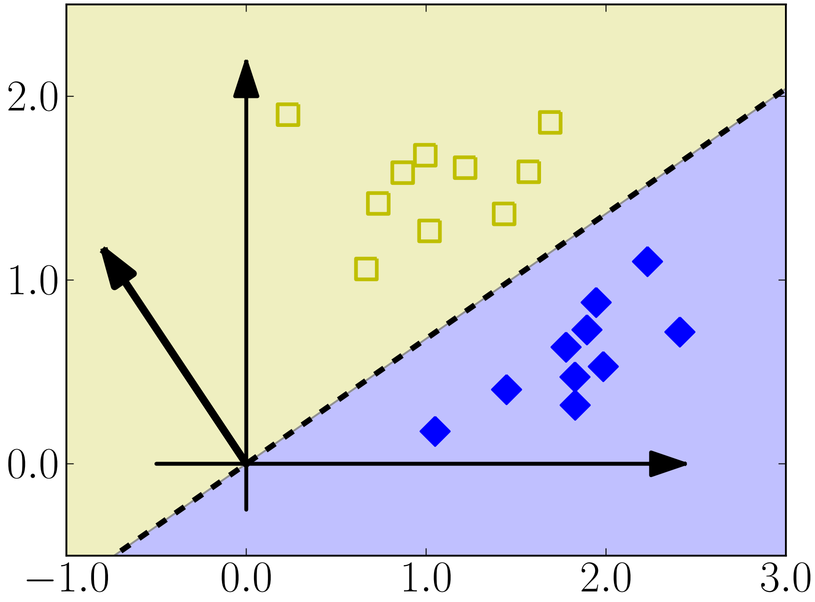
A linearly separable dataset and a correct classifier



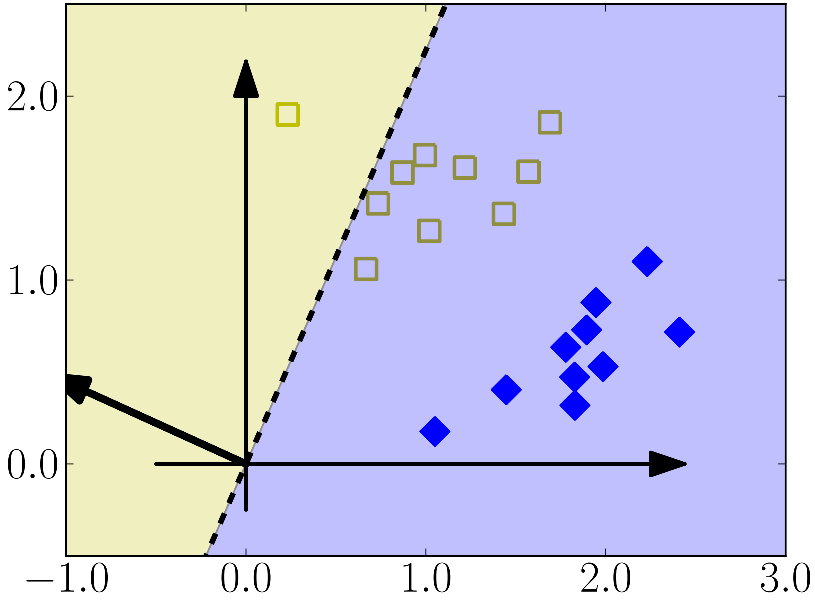
A linearly separable dataset and a correct classifier



A linearly separable dataset and a correct classifier



An incorrect classifier



Definition

The **robustness** of a classifier g (with respect to $\mathcal{D}$) is the largest amount, ρ, by which we can perturb the training samples without changing the predictions of g.

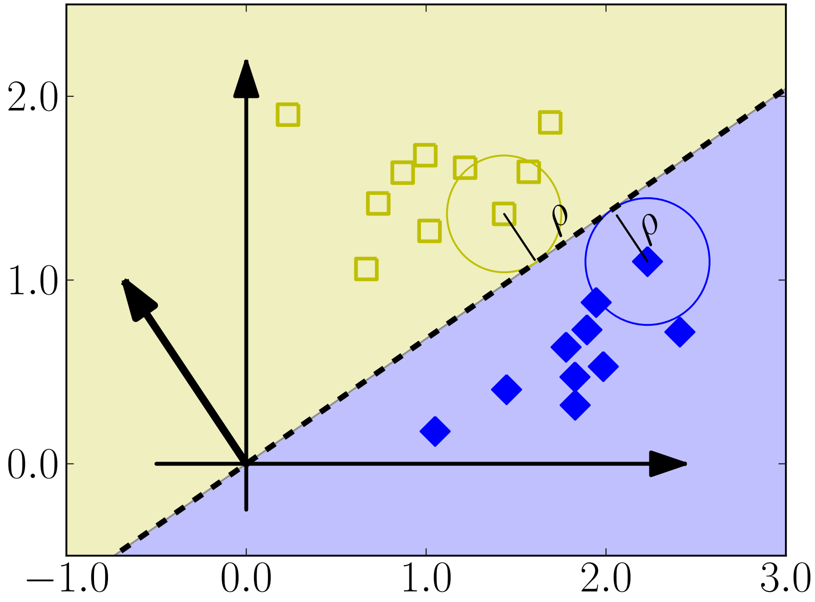
$$g(x^i + \epsilon) = g(x^i), \qquad \text{for all } i = 1, \dots, n.$$

for any $\epsilon \in \mathbb{R}^d$ with $\|\epsilon\| < \rho$.

Example

- constant classifier, e.g. $c(x) \equiv 1$: very robust ($\rho = \infty$), (but it is not *correct*, in the sense of the previous definition)
- robustness of the *Perceptron*: can be arbitrarily small (see Exercise...)

Robustness, ρ, of a linear classifier



Definition (Margin)

Let $f(x) = \langle w, x \rangle + b$ define a *correct* linear classifier.
Then the smallest (Euclidean) distance of any training example from the decision hyperplane is called the **margin** of f (with respect to $\mathcal{D}$).

Lemma

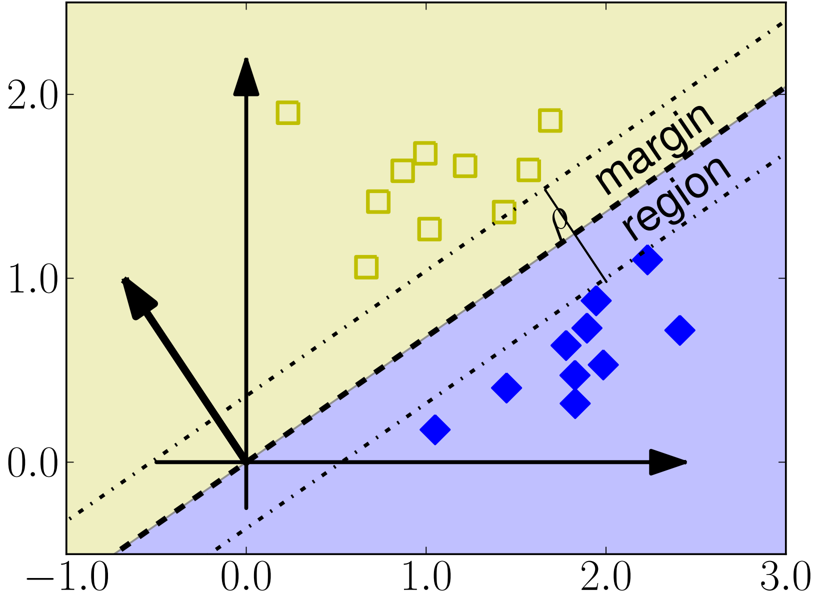
We can compute the margin of a linear classifier $f = \langle w, x \rangle + b$ *as*

$$\rho = \mathbf{min}_{i=1,\ldots,n} \left| \langle \frac{w}{\|w\|}, x^i \rangle + b \right|.$$

Proof.

High school maths: distance between a points and a hyperplane in *Hessian normal form.*

Margin, ρ, of a linear classifier



Theorem

The robustness of a linear classifier function $g(x) = \operatorname{sign} f(x)$ with $f(x) = \langle w, x \rangle + b$ is identical to the margin of f.

Correction: this only works for classifiers with $b = 0$ for now:

Proof (blackboard). For any $i = 1, \dots, n$ and any $\epsilon \in \mathbb{R}^d$

$$f(x^i + \epsilon) = \langle w, x^i + \epsilon \rangle = \langle w, x^i \rangle + \langle w, \epsilon \rangle = f(x^i) + \langle w, \epsilon \rangle,$$

so it follows (Cauchy-Schwarz inequality) that

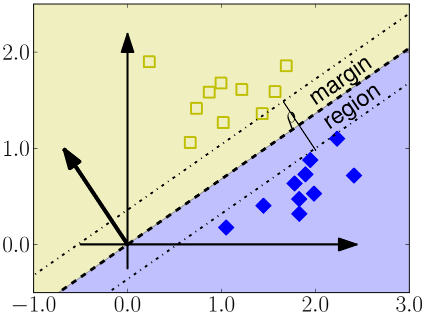
$$f(x^i) - \|w\| \|\epsilon\| \quad \leq \quad f(x^i + \epsilon) \quad \leq \quad f(x^i) + \|w\| \|\epsilon\|.$$

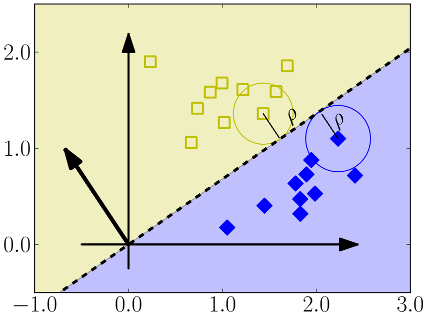
Checking the cases $\epsilon = \pm \frac{\|\epsilon\|}{\|w\|} w$, we see that these inequalities are sharp.

To ensure $g(x^i + \epsilon) = g(x^i)$ for all training samples, $f(x^i)$ and $f(x^i + \epsilon)$ have the same sign for all ϵ, i.e. $|f(x^i)| \geq \|w\| \|\epsilon\|$ for $i = 1, \dots, n$.

This inequality holds for all samples, so in particular it holds for the one of minimal score. $\square$

Proof by Picture





Maximum-Margin Classifier

Theorem

Let $\mathcal{D}$ be a linearly separable training set. Then the ***most robust, correct classifier*** *is given by* $g(x) = \operatorname{sign}\langle w^*, x\rangle + b^*$ *where* (w^*, b^*) *are the solution to*

$$\mathbf{min}_{w \in \mathbb{R}^d} \ \frac{1}{2}\|w\|^2$$

subject to

$$y^i(\langle w, x^i\rangle + b) \geq 1, \quad \textit{for } i = 1, \ldots, n.$$

Remark

- The classifier defined above is call **Maximum (Hard) Margin Classifier**, or **Hard-Margin Support Vector Machine (SVM)**
- It is unique (follows from strictly convex optimization problem).

Proof.

1. All (w, b) that fulfill the inequalities yield *correct* classifiers.
2. Since $\mathcal{D}$ is linearly separable, there exists some (v, b) with

$$\operatorname{sign}(\langle v, x^i \rangle + b) = y_i, \quad \text{i.e.} \quad y_i(\langle v, x^i \rangle + b) \geq \gamma > 0.$$

   for $\gamma = \mathbf{min}_i\, y_i(\langle v, x^i \rangle + b)$. So $\tilde{v} = v/\gamma$, $\tilde{b} = b/\gamma$ fulfills the inequalities and we see that the constraint set is at least not empty.

Proof.

1. All (w, b) that fulfill the inequalities yield *correct* classifiers.
2. Since $\mathcal{D}$ is linearly separable, there exists some (v, b) with

$$\operatorname{sign}(\langle v, x^i\rangle + b) = y_i, \quad \text{i.e.} \quad y_i(\langle v, x^i\rangle + b) \geq \gamma > 0.$$

   for $\gamma = \mathbf{min}_i\, y_i(\langle v, x^i\rangle + b)$. So $\tilde{v} = v/\gamma$, $\tilde{b} = b/\gamma$ fulfills the inequalities and we see that the constraint set is at least not empty.

3. Now we check (for all $i = 1, \dots, n$):

$$\begin{aligned}
& \mathbf{min}_{w\in\mathbb{R}^d}\, \frac{1}{2}\|w\|^2 \text{ sb.t. } y^i\langle w, x^i\rangle \geq 1 \\
\Leftrightarrow\quad & \mathbf{max}_{w\in\mathbb{R}^d}\, \frac{1}{\|w\|} \quad \text{sb.t. } y^i\langle w, x^i\rangle \geq 1 \\
\Leftrightarrow\quad & \mathbf{max}_{\{w':|w'\|=1\},\rho\in\mathbb{R}} \quad \rho \quad \text{sb.t. } y^i\langle \frac{w'}{\rho}, x^i\rangle \geq 1 \\
\Leftrightarrow\quad & \mathbf{max}_{\{w':|w'\|=1\},\rho\in\mathbb{R}} \quad \rho \quad \text{sb.t. } y^i\langle w', x^i\rangle \geq \rho \\
\Leftrightarrow\quad & \mathbf{max}_{\{w':|w'\|=1\},\rho\in\mathbb{R}} \quad \rho \quad \text{sb.t. } |\langle w', x^i\rangle| \geq \rho \text{ and } \operatorname{sign}\langle w', x^i\rangle = y_i
\end{aligned}$$

Proof.

1. All (w, b) that fulfill the inequalities yield *correct* classifiers.
2. Since $\mathcal{D}$ is linearly separable, there exists some (v, b) with

$$\operatorname{sign}(\langle v, x^i \rangle + b) = y_i, \quad \text{i.e.} \quad y_i(\langle v, x^i \rangle + b) \geq \gamma > 0.$$

for $\gamma = \mathbf{min}_i\, y_i(\langle v, x^i \rangle + b)$. So $\tilde{v} = v/\gamma$, $\tilde{b} = b/\gamma$ fulfills the inequalities and we see that the constraint set is at least not empty.

3. Now we check (for all $i = 1, \ldots, n$):

$$\mathbf{min}_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|^2 \text{ sb.t. } y^i \langle w, x^i \rangle \geq 1$$

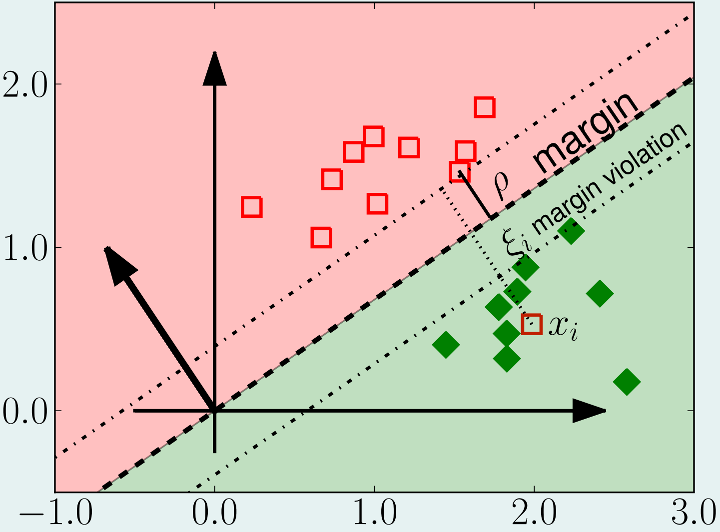
$$\Leftrightarrow \quad \mathbf{max}_{w \in \mathbb{R}^d} \frac{1}{\|w\|} \quad \text{sb.t. } y^i \langle w, x^i \rangle \geq 1$$

$$\Leftrightarrow \quad \mathbf{max}_{\{w' : |w'\|=1\}, \rho \in \mathbb{R}} \quad \rho \quad \text{sb.t. } y^i \langle \frac{w'}{\rho}, x^i \rangle \geq 1$$

$$\Leftrightarrow \quad \mathbf{max}_{\{w' : |w'\|=1\}, \rho \in \mathbb{R}} \quad \rho \quad \text{sb.t. } y^i \langle w', x^i \rangle \geq \rho$$

$$\Leftrightarrow \quad \underbrace{\mathbf{max}_{\{w' : |w'\|=1\}, \rho \in \mathbb{R}} \quad \rho \quad \text{sb.t. } |\langle w', x^i \rangle| \geq \rho}_{\text{maximal robustness}} \text{ and } \underbrace{\operatorname{sign}\langle w', x^i \rangle = y_i}_{\text{and correct}}$$

Observation (Not all training sets are linearly separable.)



Definition (Maximum Soft-Margin Classifier)

Let $\mathcal{D}$ be a training set, not necessarily linearly separable. Let $C > 0$. Then the classifier $g(x) = \operatorname{sign}\langle w^*, x\rangle$ where (w^*, b^*) are the solution to

$$\min_{w\in\mathbb{R}^d, \xi\in\mathbb{R}^n} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi^i$$

subject to

$$y^i(\langle w, x^i\rangle + b) \geq 1 - \xi^i, \quad \text{for } i = 1, \dots, n.$$
$$\xi^i \geq 0, \quad \text{for } i = 1, \dots, n.$$

is called **Maximum (Soft-)Margin Classifier** or **Linear Support Vector Machine**.

Theorem

The maximum soft-margin classifier exists and is unique for any $C > 0$.

Proof. optimization problem is strictly convex

Remark

The constant $C > 0$ is called **regularization** parameter.

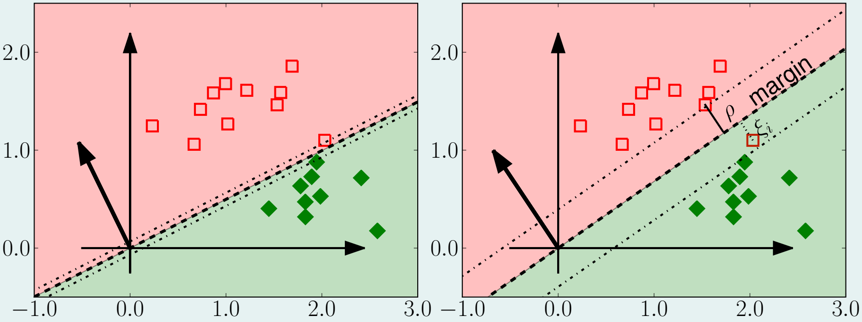
It balances the wishes for robustness and for correctness

- $C \to 0$: mistakes don't matter much, emphasis on short w
- $C \to \infty$: as few errors as possible, might not be robust

We'll see more about this tomorrow.

Remark

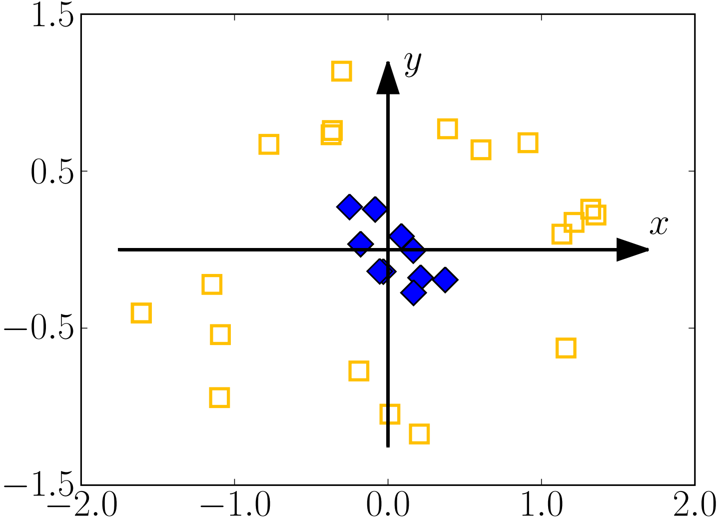
Sometimes, a soft margin is better even for linearly separable datasets!



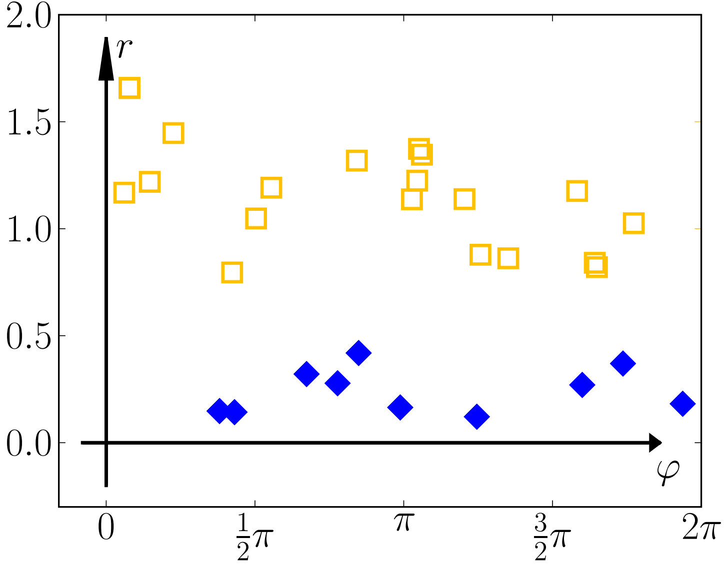
Left: small margin, no errors)

Right: large margin, but 1 error

What, if a linear classifier is really not a good choice?



What, if a linear classifier is really not a good choice?



Change the data representation, e.g. Cartesian $\rightarrow$ polar coordinates

Definition (Max-margin Generalized Linear Classifier)

Let $C > 0$. Assume a necessarily linearly separable training set

$$\mathcal{D} = \{(x^1, y^1), \dots x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}.$$

Let $\phi : \mathcal{X} \to \mathcal{H}$ be a feature map from $\mathcal{X}$ into a Hilbert space $\mathcal{H}$.

Then we can form a new training set

$$\mathcal{D}^{\phi} = \{\ (\phi(x^1), y^1),\ \dots,\ (\phi(x^n), y^n)\ \} \subset \mathcal{H} \times \mathcal{Y}.$$

The maximum-(soft)-margin linear classifier in $\mathcal{H}$,

$$g(x) = \operatorname{sign}\langle w, \phi(x)\rangle_{\mathcal{H}} + b,$$

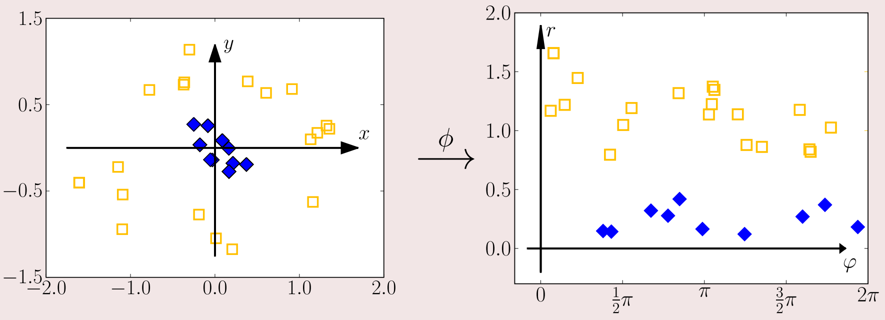
for $w \in \mathcal{H}$ and $b \in \mathbb{R}$ is called **max-margin generalized linear classifier**.

It is still *linear* w.r.t w, but (in general) nonlinear with respect to x.

Example (Polar coordinates)

Left: dataset $\mathcal{D}$ for which no good linear classifier exists.
Right: dataset $\mathcal{D}^{\phi}$ for $\phi : \mathcal{X} \to \mathcal{H}$ with $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{H} = \mathbb{R}^2$

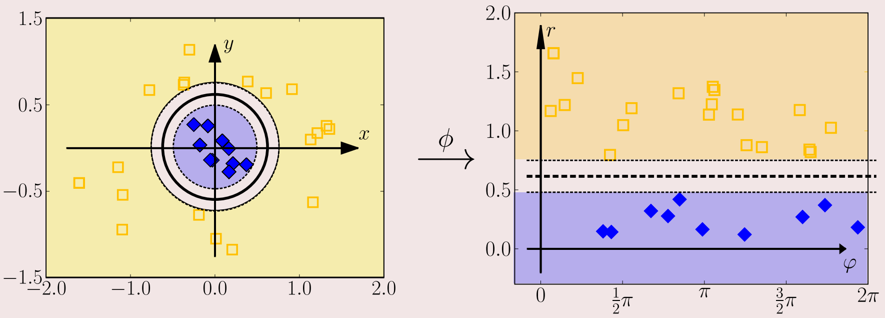
$$\phi(x, y) = (\sqrt{x^2 + y^2}, \arctan \frac{y}{x}) \qquad \text{(and } \phi(0,0) = (0,0)\text{)}$$



Example (Polar coordinates)

Left: dataset $\mathcal{D}$ for which no good linear classifier exists.
Right: dataset $\mathcal{D}^\phi$ for $\phi : \mathcal{X} \to \mathcal{H}$ with $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{H} = \mathbb{R}^2$

$$\phi(x, y) = (\sqrt{x^2 + y^2}, \arctan \frac{y}{x}) \qquad \text{(and } \phi(0,0) = (0,0)\text{)}$$



Any classifier in $\mathcal{H}$ induces a classifier in $\mathcal{X}$.

Other popular feature mappings, ϕ

Example (d-th degree polynomials)

$$\phi : \left(x_1, \dots, x_n\right) \mapsto \left(1, x_1, \dots, x_n, x_1^2, \dots, x_n^2, \dots, x_1^d, \dots, x_n^d\right)$$

Resulting classifier: d-th degree polynomial in $x . g(x) = \operatorname{sign} f(x)$ with

$$f(x) = \langle w, \phi(x) \rangle = \sum\nolimits_j w_j \phi(x)_j = \sum\nolimits_i a_i x_i + \sum\nolimits_{ij} b_{ij} x_i x_j + \dots$$

Example (Distance map)

For a set of prototype $p_1, \dots, p_N \in \mathcal{H}$:

$$\phi : \vec{x} \mapsto \left(e^{-\|\vec{x}-\vec{p}_i\|^2}, \dots, e^{-\|\vec{x}-\vec{p}_N\|^2}\right)$$

Classifier: combine weights from close enough prototypes

$$g(x) = \operatorname{sign}\langle w, \phi(x) \rangle = \operatorname{sign} \sum\nolimits_{i=1}^{n} a_i e^{-\|\vec{x}-\vec{p}_i\|^2}.$$

Finding the Maximum Margin Classifier numerically – Optimization II

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}\xi\in\mathbb{R}^n} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi^i$$

subject to

$$y^i\langle w, \phi(x^i)\rangle \geq 1 - \xi^i, \quad \text{for } i = 1, \ldots, n,$$
$$\xi^i \geq 0. \quad \text{for } i = 1, \ldots, n.$$

How to solve numerically?

- off-the-shelf Quadratic Program (QP) solver
  only for small dimensions and training sets (a few hundred),
- variants of gradient descent,
  high dimensional data, large training sets (millions)
- by convex duality,
  for very high dimensional data and not so many examples ($d \gg n$)