How can SVM 'find' an infinite feature space where linear separation is always possible?

Asked 7 years ago Active 4 years, 11 months ago Viewed 11k times

39

What is the intuition behind the fact that an SVM with a Gaussian Kernel has infinite dimensional feature space?

svm feature-selection kernel-trick

35

Share Cite Improve this question Follow

edited Aug 22 '15 at 15:23
John
20.5k 5 44 82

asked Dec 23 '13 at 11:51
user36162
531 1 5 4

1 I don't really understand the question. Do you want an explanation *why* its corresponding feature space is infinite dimensional or an interpretation as to what the resulting hyperplane means? – Marc Claesen Dec 23 '13 at 13:27

1 I wouldn't mind hearing both! – user36162 Dec 23 '13 at 15:07

5 I think this is an interesting question (+1) – user83346 Aug 22 '15 at 8:29

43

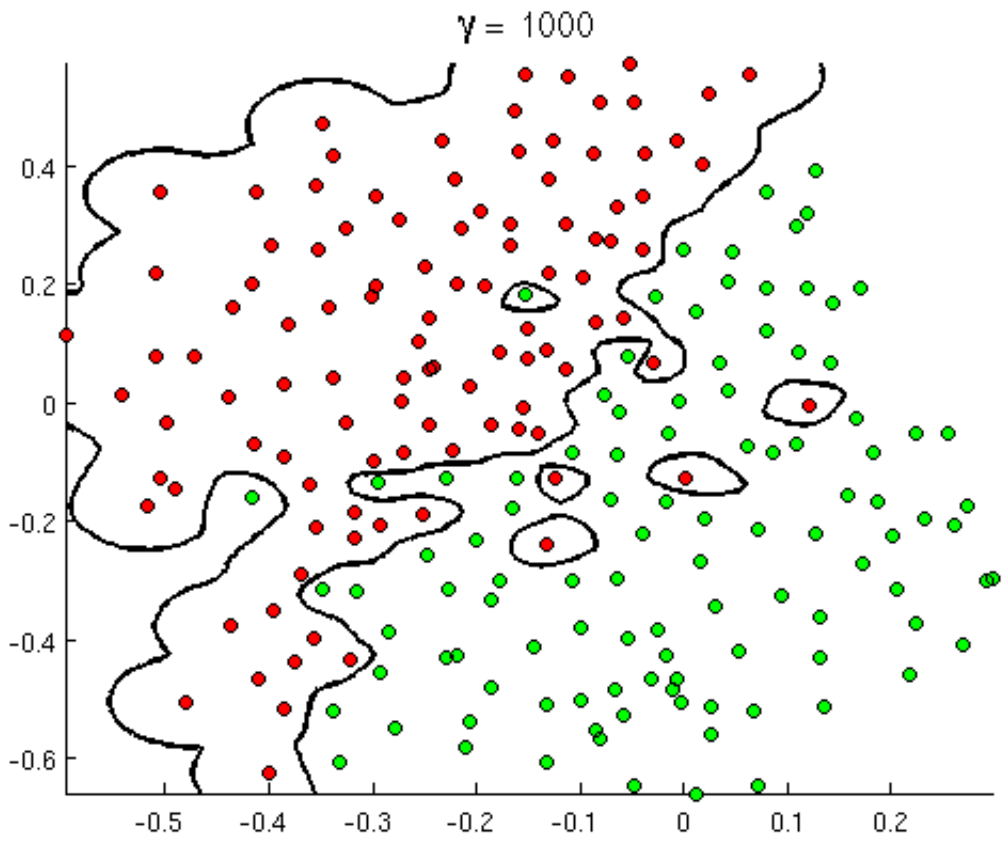
+100

This answer explains the following:

1. Why perfect separation is always possible with distinct points and a Gaussian kernel (of sufficiently small bandwidth)
2. How this separation may be interpreted as linear, but only in an abstract feature space distinct from the space where the data lives
3. How the mapping from data space to feature space is "found". Spoiler: it's not found by SVM, it's implicitly defined by the kernel you choose.
4. Why the feature space is infinite-dimensional.

1. Achieving perfect separation

Perfect separation is always possible with a Gaussian kernel (provided no two points from different classes are ever exactly the same) because of the kernel's locality properties, which lead to an arbitrarily flexible decision boundary. For sufficiently small kernel bandwidth, the decision boundary will look like you just drew little circles around the points whenever they are needed to separate the positive and negative examples:



(Credit: Andrew Ng's online machine learning course).

So, why does this occur from a mathematical perspective?

Consider the standard setup: you have a Gaussian kernel $K(\mathbf{x}, \mathbf{z}) = \exp(-||\mathbf{x} - \mathbf{z}||^2/\sigma^2)$ and training data $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})$ where the $y^{(i)}$ values are ± 1. We want to learn a classifier function

$$\hat{y}(\mathbf{x}) = \sum_i w_i y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x})$$

Now how will we ever assign the weights w_i? Do we need infinite dimensional spaces and a quadratic programming algorithm? No, because I just want to show that I can separate the points perfectly. So I make σ a billion times smaller than the smallest separation $||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||$ between any two training examples, and I just set $w_i = 1$. This means that all the training points are a billion sigmas apart as far as the kernel is concerned, and each point completely controls the sign of $\hat{y}$ in its neighborhood. Formally, we have

$$\hat{y}(\mathbf{x}^{(k)}) = \sum_{i=1}^{n} y^{(k)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}) = y^{(k)} K(\mathbf{x}^{(k)}, \mathbf{x}^{(k)}) + \sum_{i \neq k} y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}) = y^{(k)} + \epsilon$$

where ϵ is some arbitrarily tiny value. We know ϵ is tiny because $\mathbf{x}^{(k)}$ is a billion sigmas away from any other point, so for all $i \neq k$ we have

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}) = \exp(-||\mathbf{x}^{(i)} - \mathbf{x}^{(k)}||^2/\sigma^2) \approx 0.$$

Since ϵ is so small, $\hat{y}(\mathbf{x}^{(k)})$ definitely has the same sign as $y^{(k)}$, and the classifier achieves perfect accuracy on the training data.

2. Kernel SVM learning as linear separation

The fact that this can be interpreted as "perfect linear separation in an infinite dimensional feature space" comes from the kernel trick, which allows you to interpret the kernel as an inner product in a (potentially infinite-dimensional) feature space:

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \Phi(\mathbf{x}^{(i)}), \Phi(\mathbf{x}^{(j)}) \rangle$$

where $\Phi(\mathbf{x})$ is the mapping from the data space into the feature space. It follows immediately that the $\hat{y}(\mathbf{x})$ function as a linear function in the feature space:

$$\hat{y}(\mathbf{x}) = \sum_i w_i y^{(i)} \langle \Phi(\mathbf{x}^{(i)}), \Phi(\mathbf{x}) \rangle = L(\Phi(\mathbf{x}))$$

where the linear function $L(\mathbf{v})$ is defined on feature space vectors $\mathbf{v}$ as

$$L(\mathbf{v}) = \sum_i w_i y^{(i)} \langle \Phi(\mathbf{x}^{(i)}), \mathbf{v} \rangle$$

This function is linear in $\mathbf{v}$ because it's just a linear combination of inner products with fixed vectors. In the feature space, the decision boundary $\hat{y}(\mathbf{x}) = 0$ is just $L(\mathbf{v}) = 0$, the level set of a linear function. This is the very definition of a hyperplane in the feature space.

3. Understanding the mapping and feature space

Note: In this section, the notation $\mathbf{x}^{(i)}$ refers to an arbitrary set of n points and not the training data. This is pure math; the training data does not figure into this section at all!

Kernel methods never actually "find" or "compute" the feature space or the mapping Φ explicitly. Kernel learning methods such as SVM do not need them to work; they only need the kernel function K.

That said, it is possible to write down a formula for Φ. The feature space that Φ maps to is kind of abstract (and potentially infinite-dimensional), but essentially, the mapping is just using the kernel to do some simple feature engineering. In terms of the final result, the model you end up learning, using kernels is no different from the traditional feature engineering popularly applied in linear regression and GLM modeling, like taking the log of a positive predictor variable before feeding it into a regression formula. The math is mostly just there to help make sure the kernel plays well with the SVM algorithm, which has its vaunted advantages of sparsity and scaling well to large datasets.

If you're still interested, here's how it works. Essentially we take the identity we want to hold, $\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle = K(\mathbf{x}, \mathbf{y})$, and construct a space and inner product such that it holds by definition. To do this, we define an abstract vector space V where each vector is a function from the space the data lives in, $\mathcal{X}$, to the real numbers $\mathbb{R}$. A vector f in V is a function formed from a finite linear combination of kernel slices:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}^{(i)}, \mathbf{x})$$

It is convenient to write f more compactly as

$$f = \sum_{i=1}^{n} \alpha_i K_{\mathbf{x}^{(i)}}$$

where $K_{\mathbf{x}}(\mathbf{y}) = K(\mathbf{x}, \mathbf{y})$ is a function giving a "slice" of the kernel at $\mathbf{x}$.

The inner product on the space is not the ordinary dot product, but an abstract inner product based on the kernel:

$$\langle \sum_{i=1}^{n} \alpha_i K_{\mathbf{x}^{(i)}}, \sum_{j=1}^{n} \beta_j K_{\mathbf{x}^{(j)}} \rangle = \sum_{i,j} \alpha_i \beta_j K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

With the feature space defined in this way, Φ is a mapping $\mathcal{X} \to V$, taking each point $\mathbf{x}$ to the "kernel slice" at that point:

$$\Phi(\mathbf{x}) = K_{\mathbf{x}}, \quad \text{where} \quad K_{\mathbf{x}}(\mathbf{y}) = K(\mathbf{x}, \mathbf{y}).$$

You can prove that V is an inner product space when K is a positive definite kernel. See this paper for details. (Kudos to f coppens for pointing this out!)

4. Why is the feature space infinite-dimensional?

This answer gives a nice linear algebra explanation, but here's a geometric perspective, with both intuition and proof.

Intuition

For any fixed point $\mathbf{z}$, we have a kernel slice function $K_{\mathbf{z}}(\mathbf{x}) = K(\mathbf{z}, \mathbf{x})$. The graph of $K_{\mathbf{z}}$ is just a Gaussian bump centered at $\mathbf{z}$. Now, if the feature space were only finite dimensional, that would mean we could take a finite set of bumps at a fixed set of points and form any Gaussian bump anywhere else. But clearly there's no way we can do this; you can't make a new bump out of old bumps, because the new bump could be really far away from the old ones. So, no matter how many feature vectors (bumps) we have, we can always add new bumps, and in the feature space these are new independent vectors. So the feature space can't be finite dimensional; it has to be infinite.

Proof

We use induction. Suppose you have an arbitrary set of points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$ such that the vectors $\Phi(\mathbf{x}^{(i)})$ are linearly independent in the feature space. Now find a point $\mathbf{x}^{(n+1)}$ distinct from these n points, in fact a billion sigmas away from all of them. We claim that $\Phi(\mathbf{x}^{(n+1)})$ is linearly independent from the first n feature vectors $\Phi(\mathbf{x}^{(i)})$.

Proof by contradiction. Suppose to the contrary that

$$\Phi(\mathbf{x}^{(n+1)}) = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}^{(i)})$$

Now take the inner product on both sides with an arbitrary $\mathbf{x}$. By the identity $\langle \Phi(\mathbf{z}), \Phi(\mathbf{x}) \rangle = K(\mathbf{z}, \mathbf{x})$, we obtain

$$K(\mathbf{x}^{(n+1)}, \mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}^{(i)}, \mathbf{x})$$

Here $\mathbf{x}$ is a free variable, so this equation is an identity stating that two functions are the same.

In particular, it says that a Gaussian centered at $\mathbf{x}^{(n+1)}$ can be represented as a linear combination of Gaussians at other points $\mathbf{x}^{(i)}$. It is obvious geometrically that one cannot create a Gaussian bump centered at one point from a finite combination of Gaussian bumps centered at other points, especially when all those other Gaussian bumps are a billion sigmas away. So our assumption of linear dependence has led to a contradiction, as we set out to show.

Share Cite Improve this answer Follow

edited Apr 13 '17 at 12:44
Community ♦
1

answered Aug 22 '15 at 17:28
Paul
9,332 1 24 49

7 **Perfect separation is impossible.** Counterexample: (0,0,ClasssA), (0,0,ClassB). Good luck separating this data set! – Has QUIT--Anony-Mousse Feb 25 '16 at 17:42

4 That's... technically correct, the best kind of correct! Have an upvote. I'll add a note in the post. – Paul Feb 25 '16 at 17:45

3 (I do think your point makes sense if you require a minimum distance between samples of different classes. It may be worth pointing out that in this scenario, the SVM becomes a nearest-neighbor classifier) – Has QUIT--Anony-Mousse Feb 25 '16 at 17:52

1 I'm only addressing the finite training set case, so there's always a minimum distance between points once we are given a training set of n distinct points to work with. – Paul Feb 25 '16 at 17:57

@Paul Regarding your section 2, I have a question. Let k_i be the representer in our RKHS for training point $x^{(i)}$ and k_x for arbitrary new point x so that $\hat{y}(x) = \sum_i w_i y^{(i)} \langle k_i, k_x \rangle = \sum_i w_i y^{(i)} k_i(x)$ so the function $\hat{y} = \sum_i z_i k_i$ for some $z_i \in \mathbb{R}$. To me this is like the function space version of $\hat{y}$ being in the column space of X for linear regression and is where the linearity really comes from. Does this description seem accurate? I'm still very much learning this RKHS stuff. – jld Sep 19 '16 at 15:57

13

The kernel matrix of the Gaussian kernel has always full rank for distinct $\mathbf{x}_1, \ldots, \mathbf{x}_m$. This means that each time you add a new example, the rank increases by 1. The easiest way to see this if you set σ very small. Then the kernel matrix is almost diagonal.

The fact that the rank always increases by one means that all projections $\Phi(\mathbf{x})$ in feature space are linearly independent (not orthogonal, but independent). Therefore, each example adds a new dimension to the span of the projections $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_m)$. Since you can add uncountably infinitely many examples, the feature space must have infinite dimension. Interestingly, all projections of the input space into the feature space lie on a sphere, since $||\Phi(\mathbf{x})||^2_{\mathcal{H}} = k(\mathbf{x}, \mathbf{x}) = 1$. Nevertheless, the geometry of the sphere is flat. You can read more on that in

Burges, C. J. C. (1999). Geometry and Invariance in Kernel Based Methods. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.), Advances in Kernel Methods Support Vector Learning (pp. 89–116). MIT Press.

Share Cite Improve this answer Follow

answered Dec 23 '13 at 13:33

fabee
2,323 12 18

I still don't understand it, but you earned an upvote anyway :) – stmax Aug 13 '14 at 13:29

You mean, you don't understand why the geometry is flat or why it is infinite dimensional? Thanks for the upvote. – fabee Aug 13 '14 at 14:56

1 If I have 100 examples, is my feature space 100-dimensional or already infinitely dimensional? Why can I add "uncountably" infinitely many examples? Isn't that a countable infinity? Why does countable/uncountable matter here? I didn't even try thinking about the "flat sphere" yet :D Thanks for your explanations! – stmax Aug 13 '14 at 16:50

6 I hope you believe me that every new example is linearly independent from all the ones before (except for the same x). In $\mathbb{R}^n$ you cannot do that: Every point beyond n must be linearly dependent on the others. For the Gaussian RKHS, if you have 100 different examples, they span a 100 dimensional subspace of the infinite dimensional space. So the span is finite dimensional, but the features space they live in is infinite dimensional. The infinity is uncountable, because every new point in $\mathbb{R}^n$ is a new dimension and there are uncountably many points in $\mathbb{R}^n$. – fabee Aug 13 '14 at 18:53

@fabee: I tried it in a different way, you seem yo know a lot about it, can you take a look at my answer whether I got it more or less 'right' ? – user83346 Aug 22 '15 at 8:26

7

For the background and the notations I refer to the answer How to calculate decision boundary from support vectors?.

So the features in the 'original' space are the vectors x_i, the binary outcome $y_i \in \{-1, +1\}$ and the Lagrange multipliers are α_i.

It is known that the Kernel can be written as $K(x, y) = \Phi(x) \cdot \Phi(y)$ ('$\cdot$' represents the inner product.) Where Φ is an (implicit and unknown) transformation to a new feature space.

I will try to give some **'intuitive' explanation** of what this Φ looks like, so this answer is no formal proof, it just wants to **give some feeling of how I think that this works.** Do not hesitate to correct me if I am wrong. The basis for my explanation is section 2.2.1 of this pdf

I have to 'transform' my feature space (so my x_i) into some 'new' feature space in which the linear separation will be solved.

For each observation x_i, I define functions $\phi_i(x) = K(x_i, x)$, so I have a function ϕ_i for each element of my training sample. These functions ϕ_i span a vector space. The vector space spanned by the ϕ_i, note it $V = span(\phi_{i,i=1,2,\dots N})$. ($N$ is the size of the training sample).

I will try to argue that this vector space V is the vector space in which linear separation will be possible. By definition of the span, each vector in the vector space V can be written as as a linear combination of the ϕ_i, i.e.: $\sum_{i=1}^N \gamma_i \phi_i$, where γ_i are real numbers. So, in fact, $V = \{v = \sum_{i=1}^N \gamma_i \phi_i | (\gamma_1, \gamma_2, \dots \gamma_N) \in \mathbb{R}^N\}$

Note that $(\gamma_1, \gamma_2, \dots \gamma_N)$ are the coordinates of vector v in the vector space V.

N is the size of the training sample and therefore the dimension of the vector space V can go up to N, depending on whether the ϕ_i are linear independent. As $\phi_i(x) = K(x_i, x)$ (see supra, we defined ϕ in this way), this means that **the dimension of V depends on the kernel used and can go up to the size of the training sample.**

If the kernel is 'complex enough' then the $\phi_i(x) = K(x_i, x)$ will all be independent and then the dimension of V will be N, the size of the training sample.

The transformation, that maps my original feature space to V is defined as

$\Phi : x_i \to \phi_i(x) = K(x_i, x)$.

This map Φ maps my original feature space onto a vector space that can have a dimension that goes up to the size of my training sample. So Φ maps each observation in my training sample into a vector space where the vectors are functions. The vector x_i from my training sample is 'mapped' to a vector in V, namely the vector ϕ_i with coordinates all equal to zero, except the i-th coordinate is 1.

Obviously, this transformation (a) depends on the kernel, (b) depends on the values x_i in the training sample and (c) can, depending on my kernel, have a dimension that goes up to the size of my training sample and (d) the vectors of V look like $\sum_{i=1}^N \gamma_i \phi_i$, where γ_i are real numbers.

Looking at the function $f(x)$ in How to calculate decision boundary from support vectors? it can be seen that $f(x) = \sum_i y_i \alpha_i \phi_i(x) + b$. The decision boundary found by the SVM is $f(x) = 0$.

In other words, $f(x)$ is a linear combination of the ϕ_i **and $f(x) = 0$ is a linear separating hyperplane in the V-space** : it is a particular choice of the γ_i namely $\gamma_i = \alpha_i y_i$!

The y_i are known from our observations, the α_i are the Lagrange multipliers that the SVM has found. In other words SVM find, through the use of a kernel and by solving a quadratic programming problem, a linear separation in the V-spave.

This is my intuitive understanding of how the 'kernel trick' allows one to 'implicitly' transform the original feature space into a new feature space V, with a different dimension. This dimension depends on the kernel you use and for the RBF kernel this dimension can go up to the size of the training sample. As training samples may have any size this **could go up to 'infinite'**. Obviously, in very high dimensional spaces the risk of **overfitting** will increase.

So kernels are a technique that allows SVM to transform your feature space , see also What makes the Gaussian kernel so magical for PCA, and also in general?

Share Cite Improve this answer Follow

edited Apr 13 '17 at 12:44
Community ♦
1

answered Aug 22 '15 at 7:49
user83346

+1 this is solid. I translated this material into my own expository style and added it to my answer. – Paul Aug 23 '15 at 13:38

5

Unfortunately, fcop's explanation is quite incorrect. First of all he says "It is known that the Kernel can be written as... where ... is an (implicit and unknown) transformation to a new feature space." It's NOT unknown. This is in fact the space the features are mapped to and this is the space that could be infinite dimensional like in the RBF case. All the kernel does is take the inner product of that transformed feature vector with a transformed feature vector of a training example and applies some function to the result. Thus it implicitly represents this higher dimensional feature vector. Think of writing (x+y)^2 instead of x^2+2xy+y^2 for example. Now think what infinite series is represented implicitly by the exponential function... there you have your infinite feature space. This has absolutely nothing to do with the fact that your training set could be infinitely large.

The right way to think about SVMs is that you map your features to a possibly infinite dimensional feature space which happens to be implicitly representable in yet another finite dimensional "Kernel" feature space whose dimension could be as large as the training set size.

Share Cite Improve this answer Follow

answered Feb 1 '16 at 16:11

salvador
137 1 6