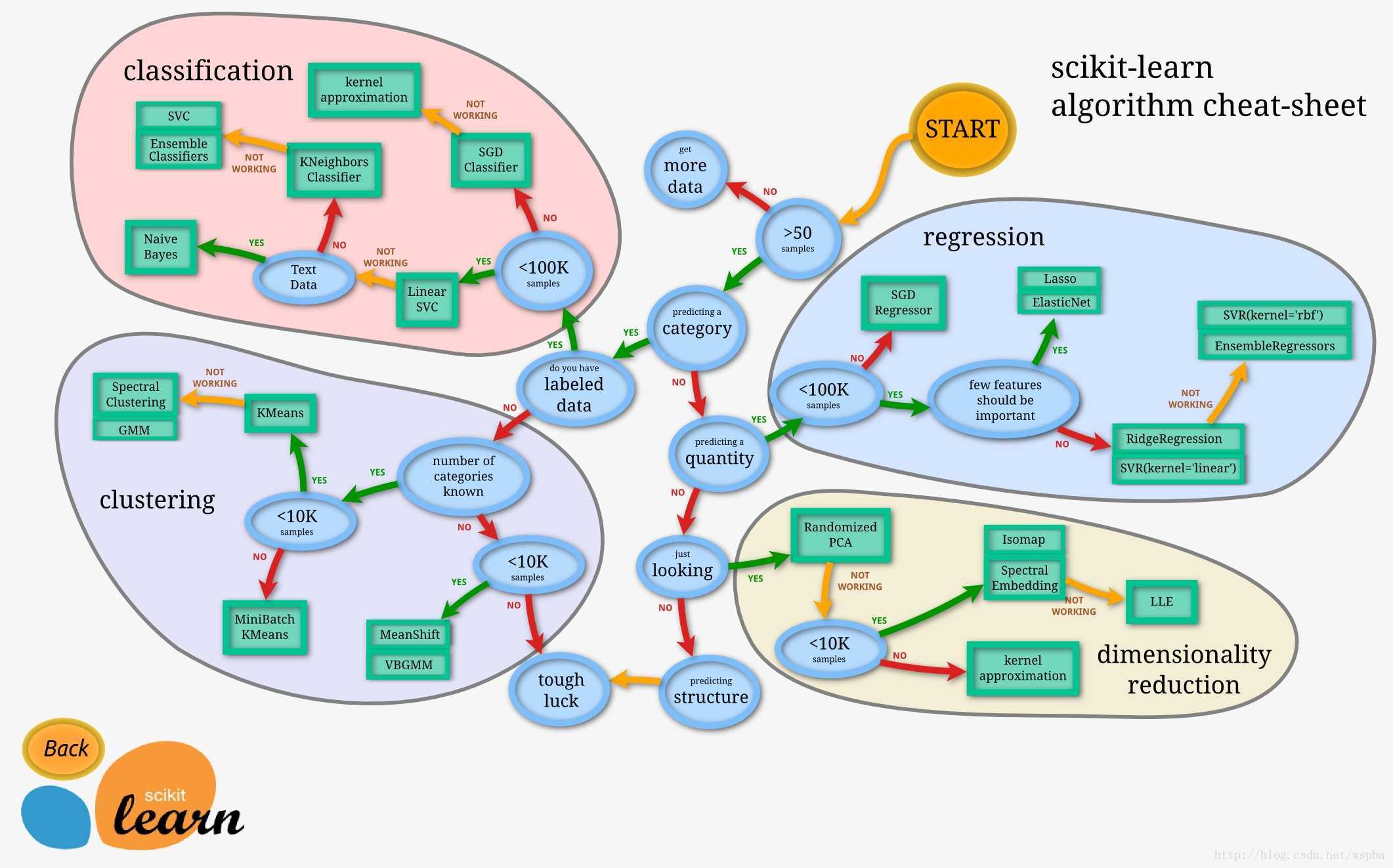
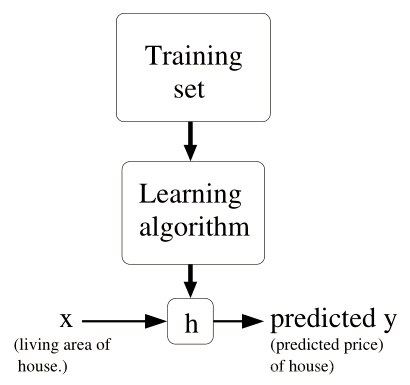
ML



**Lesson 1**

① Supervised Learning (SL)



Continuous 🡪 Regression Question (to predict)

Discrete 🡪 Classification Question (to ‘chose’ whether or not)

② Unsupervised Learning (UL)

No comparative sets or no certain answer 🡪 Clustering

③ Reinforcement Learning (RL)

‘Good Dog’ & ‘Bad Dog’ Question 🡪 Maximize ‘Reward Signal’ (decision making, like SLAM)

**Lesson 2**

① Optimization Question 🡪 least square method 🡪 ***Gradient Descent Algorithm***

② GDA associated

called ‘learning rate’, need to set by experience because too large can lead to go over the minimum, accordingly, too small can lead to convergent slowly.

(1) Batch gradient descent (e.g. linear regression)

.

**🡪** :

🡪🡪:

∴ *Batch Gradient Descent called*

(2) Stochastic gradient descent (suit large quantities of training sets)

‘accelerate’ algorithm, local minimize feature not as good as (1), just swing around it.

*(Q: How to check convergence? Any optimized solution?)*

③ Normal Equation

GDA iteration complex 🡪 Matrix-related presentation (trace properties & equation deductions)

①

🡪

②

🡪

③

🡪 ,

④

try to concise iteration by upwards definition & properties

cannot inverse?

1. features have extra linear relation 🡪delete!

2. m<n 🡪regularize training examples!

**Lesson 3**

*Outline this Lesson:*

1. Linear regression🡪locally weighted regression

↓

2. Probabilistic interpretation ( probabilities’ expression)

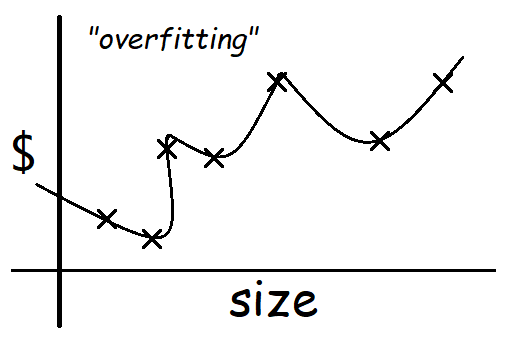
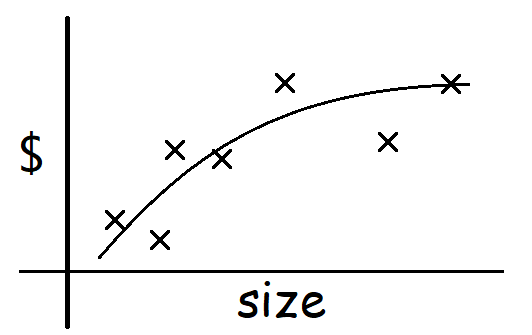
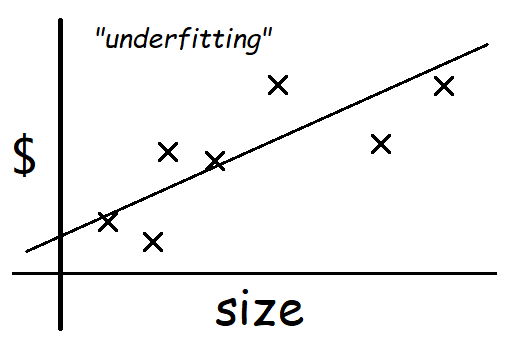
↓

3. Logistic regression 🡪perceptron algorithm

① linear regression extension (polynomial regression)

linear regression parameter can be represent like this:

🡪*underfitting* & *overfitting*



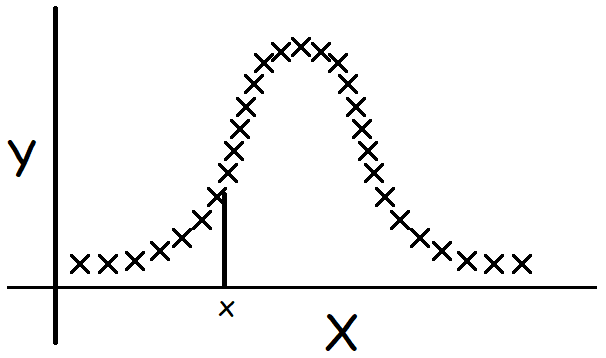
”Parametric” learning algorithm: training examples - get s - fixed set of parameters

🡪”Non-parametric” learning algorithm:

In another word, parameters rely the whole training sets, even after learning.

🡪 *locally weighted regression*(loess/lowess)

e.g.



② Probabilistic interpretation

🡪

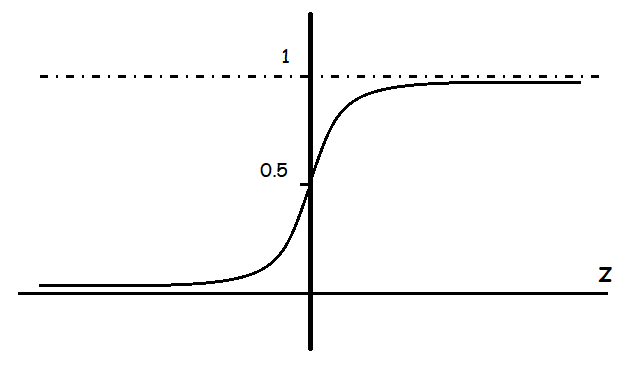
PS: means the probability of given (independent variables);

means the probability of given and parameterized by

**🡪** :

③ Logistic regression (Classification)

🡪

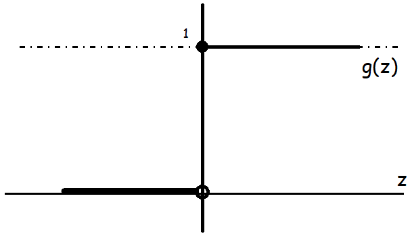


🡪 ,

🡪

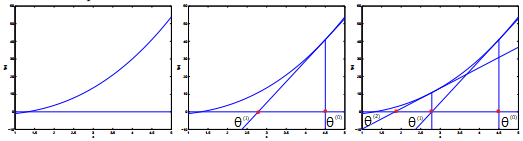
🡪

perceptron algorithm



**Lesson 4**

① Logistic regression – Newton’s Method



🡪

② Exponential Family

🡪

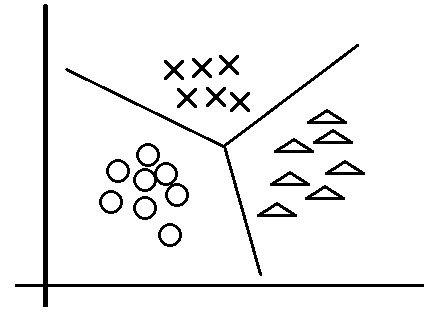
🡪

🡪

③ Generalized Linear Models (GLMs)

🡪

🡪

🡪

🡪

Softmax regression:

🡪

🡪

**Lesson 5**

*Outline this Lesson:*

1. Generative learning algorithms

2. GDA (Gaussian discriminant analysis)🡪digression: Gaussians

3. Generative & Discriminative comparison (GDA & logistic regression comparison)

4. Naive Bayes

5. Laplace Smoothing

① generative learning algorithms (e.g. logistic regression)

🡪

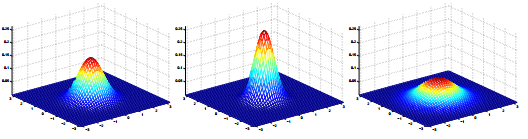
-

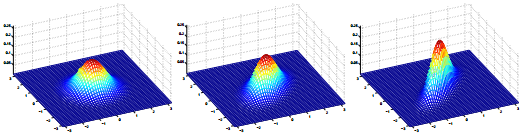
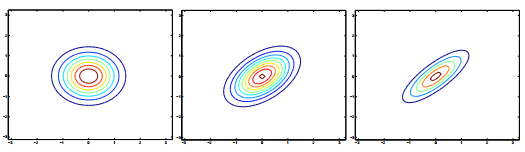
-

🡪

-

② GDA (Gaussian Discriminant Analysis)



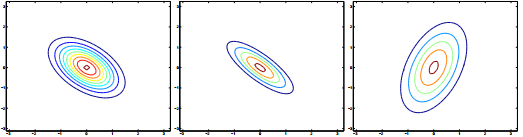


figure: different covariance of Gaussian functions (2-D)

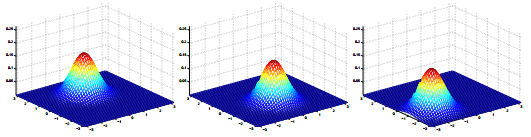
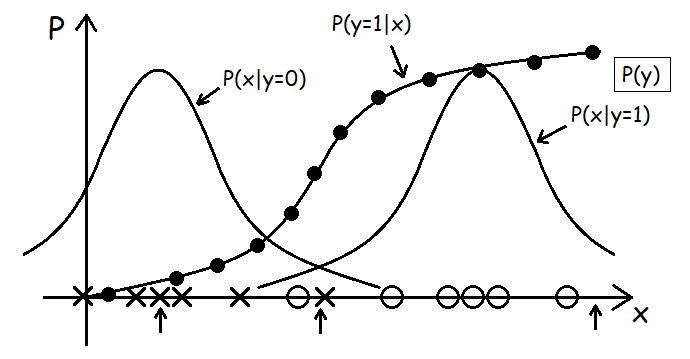


figure: different means of Gaussian functions (2-D)

🡪



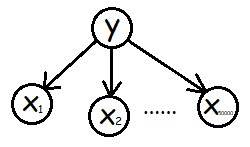
③ GDA & logistic regression comparison

It shows that logistic regression is robust for different classification questions.

GDA makes stronger modeling assumptions, and is more data efficient (i.e., requires less training data to learn “well”) when the modeling assumptions are (approximately) correct.

Logistic regression makes weaker assumptions, and is significantly more robust to deviations from modeling assumptions.

④ Naive Bayes (e.g. text classification: anti-spam)



⑤ Laplace Smoothing

When a brand new word (e.g. “NIPS”, which is the 30,000th word in dictionary) inputs, the prior probabilities are as follows:

To avoid this problem, we use Laplace Smoothing to replace estimate with

**Lesson 6**

*Outline this Lesson:*

1. Naive Bayes – event models

2. Neural Networks (Intro.)

3. Support Vector Machines

① event models

called *“Multi-variate Bernoulli* *event model”*

(hererepresents one of samples,is one of words in dictionary whether inor not)

*without using number information?* 🡪

*Multinomial event model*

e.g.



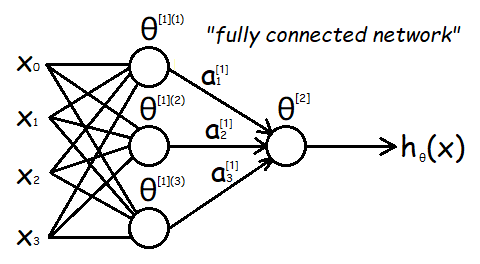
back to primal example, we can change the model with

(is a vector with different length in emails,represents the number where in dictionary)

🡪

Because Naive Bayes algorithm is built by Bernoulli or Multinomial (i.e. or other distributions), it still belongs to exponential family (GLMs), so finally, it is still a *linear classifier*.

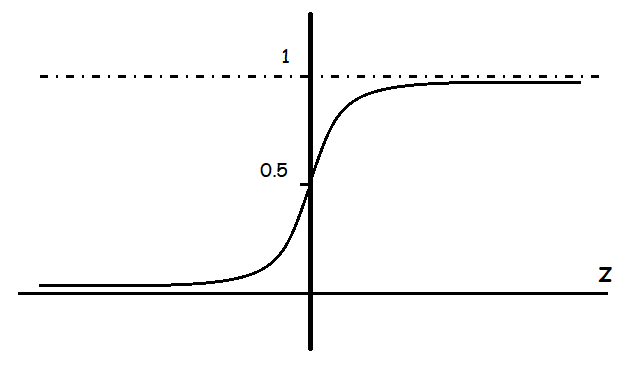
② Neural Networks



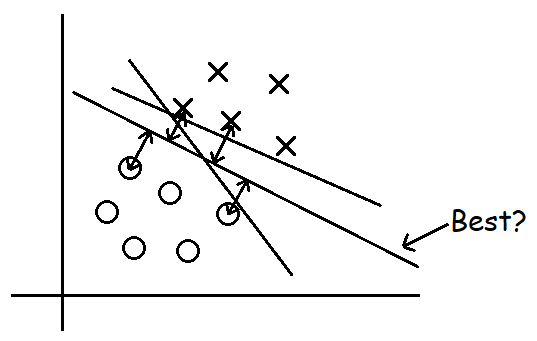
*Q: How to build a basic neural network and compute all the parameters?*

*- references: cs229-notes-deep\_learning & cs229-notes-backprop*

③ Support Vector Machines

(1) Functional margins (e.g. logistic regression)

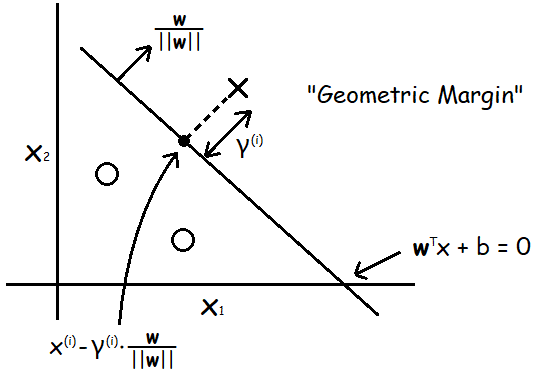
(2) Geometric margins



(3) notation

(4) definition & deduction

🡪



🡪

So to the whole samples, we have the definition of “geometric margin” as follows:

**Lesson 7**

*Outline this Lesson:*

1. Optimal Margin Classifier

2. Primal/Dual Optimization problem (KKT)

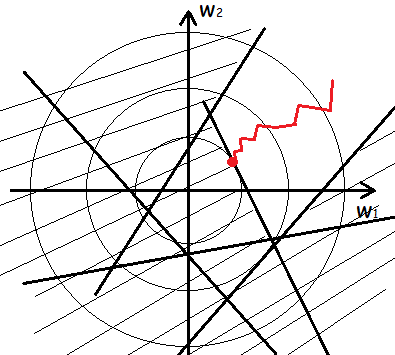
3. SVM dual

4. Kernels

① Optimal margin classifier

from last lesson, we know that change the constraints like:

will not change the original optimization problem. so,

🡪

so, original problem can re-describe into:

② Primal/Dual Optimization problem

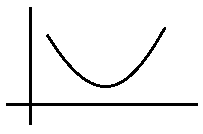
here's a function of that

we can build a *“Lagrange equation”* to re-describe it as:

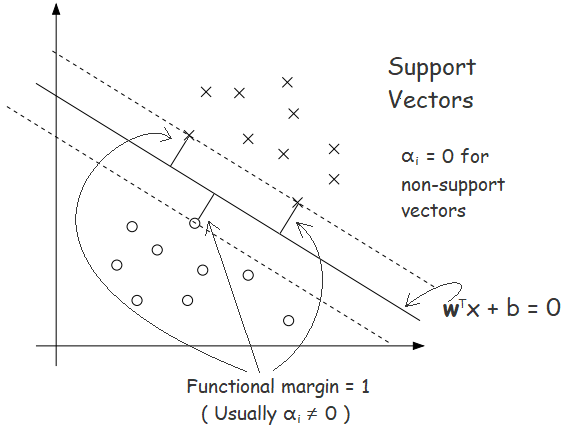
🡪

- *why?*

🡪



③ SVM dual (& KKT conditions)



SVM cost function as:

Lagrange duality

From above, we know that just associate with **.** So,

Once we solve the dual problem, we can easily solve for :

\*Finally, to predict new sample(s), we just need calculate:

④ Kernels

*\*Lagrange duality problem must be a convex optimization problem!*

proof as follows:

(<https://www.cnblogs.com/xubing-613/p/5941549.html> )

**Lesson 8**

*Outline this Lesson:*

1. Kernels

2. Soft margin

3. SMO algorithm

① Kernels

e.g.

e.g. 2

e.g. 3

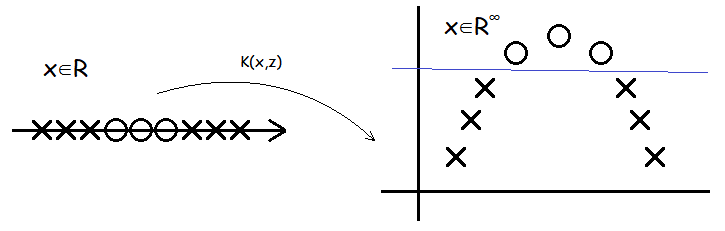
e.g. 4 (polynomial kernel)

e.g. 5 (Gaussian kernel/radial basis function)

*- ?*

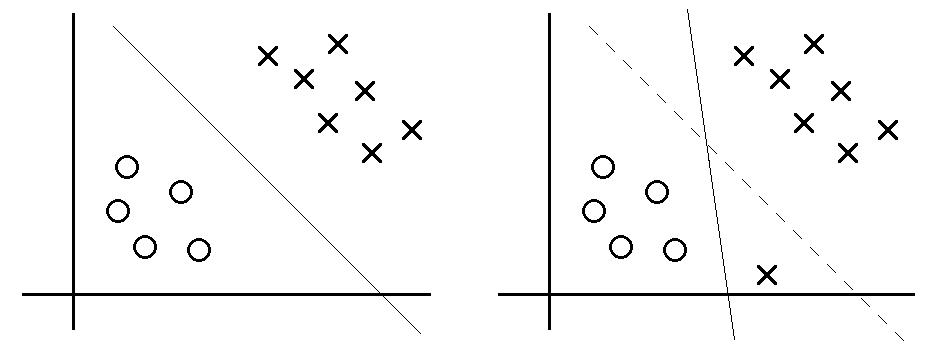
e.g.

*- How to use kernels?*



② Soft margin (Regularization)

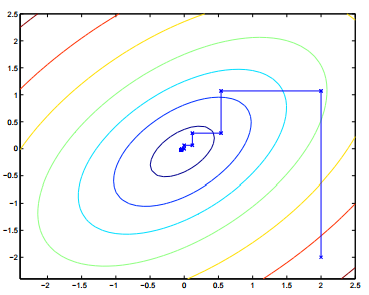
“L1 (or L2) norm soft margin SVM”



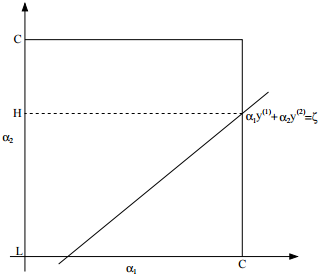
③ SMO algorithm

- Digression: Coordinate ascent

Considering an unconstrained optimization problem:



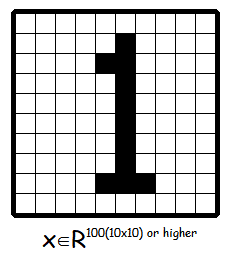
- Sequential Minimal Optimization (SMO)



reference of how to choose& update: *John Platt’s paper of SMO*

- two brief examples of using SVM

#1 Handler’s Integer Recognition



#2 classify protein sequences

*(even though only 20 to human’s protein XD)*

- *how to represent?*

**Lesson 9**

*Outline this Lesson:*

Learning Theory

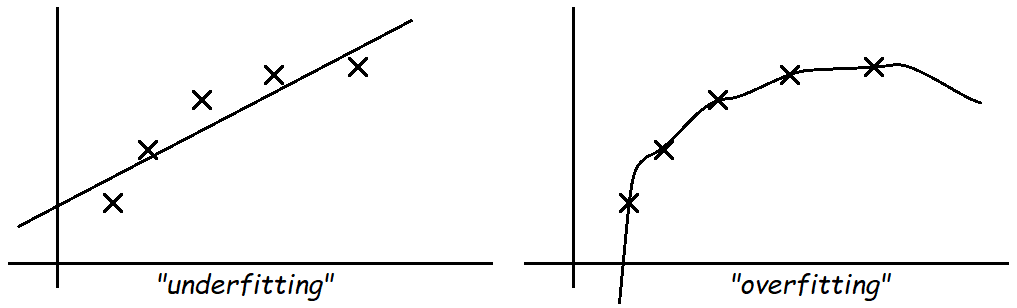
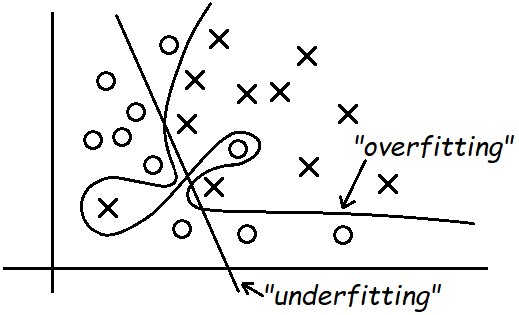
1. Bias/Variance

2. Empirical Risk Minimization (ERM)

3. Union Bound & Hoeffding inequality

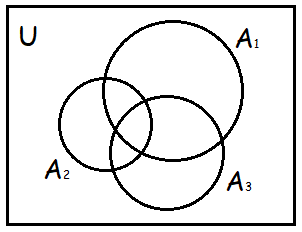
4. The case of finite (Uniform convergence)

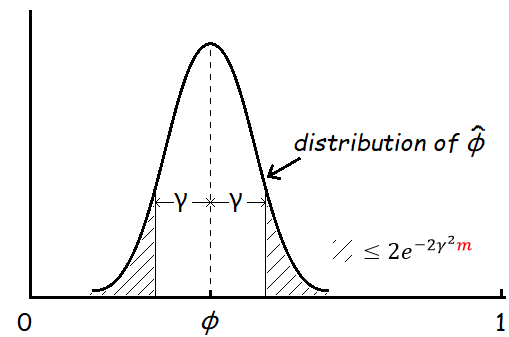
① Bias/Variance

② Empirical Risk Minimization (ERM)

All above is just curious about training with training sets, but our ultimate goal is the prediction, which means the accuracy of what we want to classify or seize. So we define

③ Union Bound & Hoeffding inequality



④ The case of finite (Uniform convergence)

So we have the *uniform convergence*:

From above, we can know that there are 3 quantities of interest: #samples , bound and the probability of error. Here’s some other ways to re-describe this solution where we can bound either one in terms of the other two!

(1)

Transform to solve m with this equation, we have the *sample complexity* bound:

In practical, almost for finite ’s length k in Computer Science, it has “”, which means we don’t worry too much about of vary kinds of hypothesis functions.

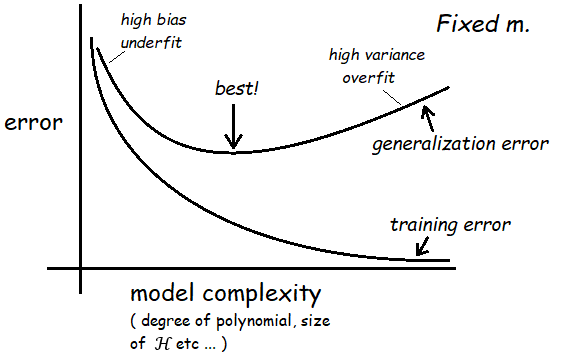
(2)

Similarly, we have the *error bound*:

From above, we have proved a so-called *bias variance tradeoff* theorem:

E.g., we have some larger hypothesis class , switching to means

(bias) can only decrease, meanwhile, the term will increase (variance) by increasing k.



PS: is the *size* of training set, is the *gap* between training error & generalization error, is the *probability* of this condition isn’t satisfied.

**Lesson 10**

*Outline this Lesson:*

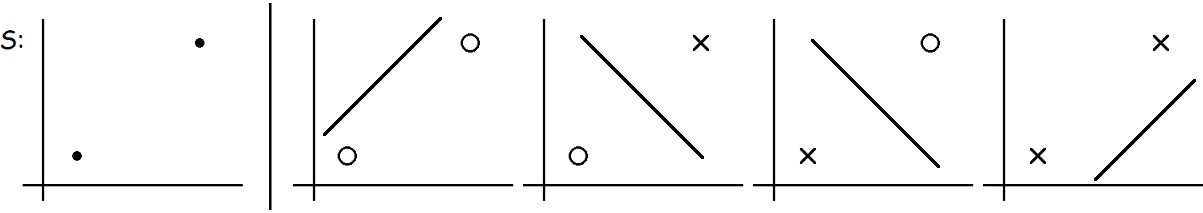
Learning Theory

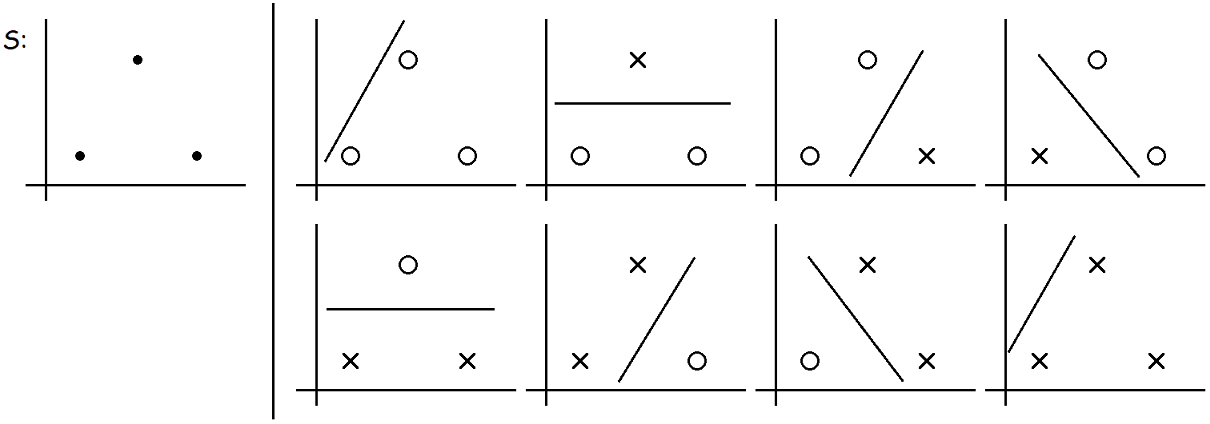
1. The case of infinite (VC dimension)🡪digression: tie up some loose facts

2. Model selection: Cross validation & Feature Selection

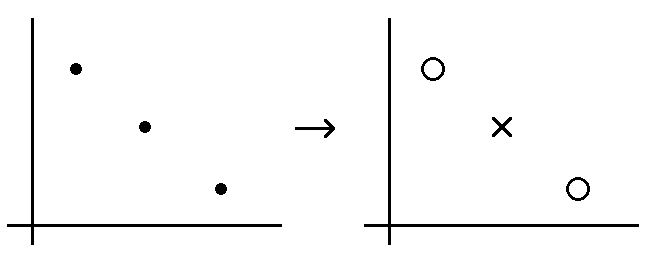
① The case of infinite (VC dimension)

In this case, the number of training set almost linear with the parameter, but not all the hypothesis class own this linear feature (generally, only suit for linear hypothesis).





*Q: this cannot shattered bysituation?*



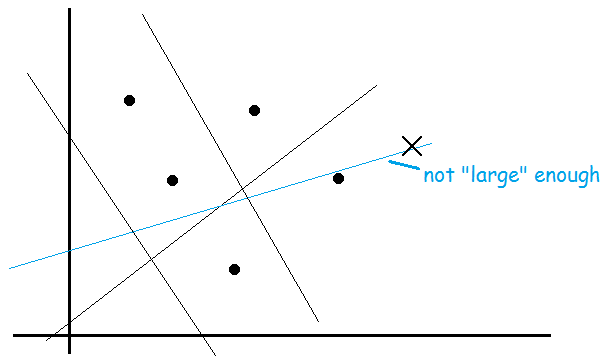
It doesn’t matter. Under the definition of the VC dimension, in order to prove that is at least , we need to show only that there’s one set of size that can shatter.

With the definition of VC dimension, we can prove a theorem of the infinite (the progress of proof quite complicated, here just conclusion).

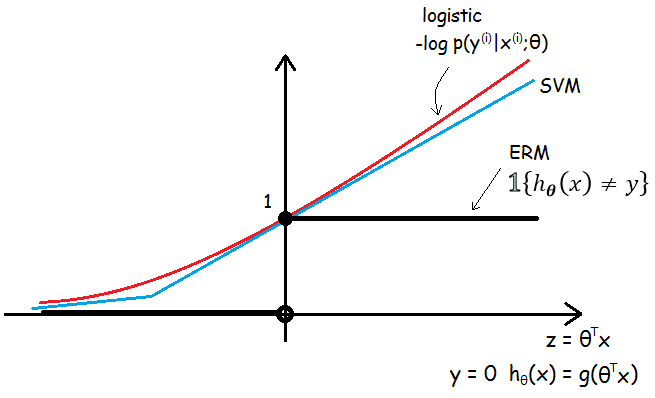
It means that, if we want to minimize the gap between training error with generalization error whether the hypothesis is finite or infinite, the least number of training set must be the *same order* with (i.e. change with the same rate, cause it’s just a very loose bound that has implicitly considered worse or even the worst condition) the number of hypothetical features (VC dimension).

- digression: tie up some loose facts

#1 Large margin linear classifier (e.g. SVM), i.e.,



#2 ERM refers to logistic regression and SVM, i.e.,

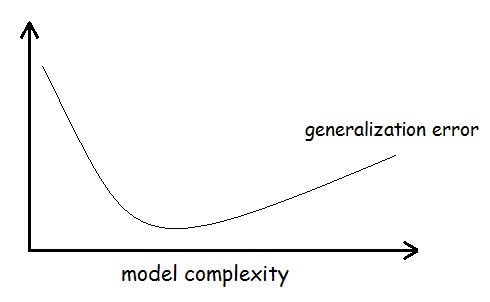


Considering only one parameter’s condition, logistic regression and SVM can be both viewed as the convex approximation of ERM(can also generalize), because the linear classifiers minimizing the training error is an NP-hard problem(not convex).

② Model selection: Cross validation & Feature Selection

(1) cross validation

Relationship between generalization error and model complexity is re-drawn as:



Consider 3 kinds of problems of selecting among several models for a learning problem:

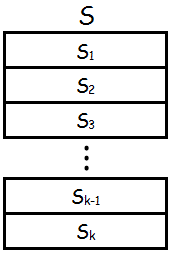
🡪 Practically, these kinds of problems we assume we have some finite set of models

Here can consist of SVM, neural network, logistic regression etcetera.

🡪 *Hold-out cross validation* (or *simple cross validation*):

- Easy and fast to use especially in quite large sets, but not sufficiently enough.

🡪 *-fold cross validation*:



- Suit for data-scarce sets (e.g.) and be more sufficient in using data, but inevitably accompany with more computational expense.

🡪 *Leave-one-out cross validation:* when, with the extremely rare data.

(2) feature selection

Some special learning problems could be described as: the number of featuresis (extremely) larger than the number of training set, but there is only a small number of features that are “(strongly) relevant” to the learning task. It means that

However we almost cannot search for the very features through comparing allmodels, here we have got some heuristic search procedure as follows.

🡪 *Forward search*:

- The algorithm can be finally terminated in 2 situations: all features are in, exceeds some pre-set threshold (e.g. pre-set).

PS: It’s a kind of *“wrapper” model feature selection*.

*Backward search*:

🡪 *”Filter” feature selection*:

E.g.

Here we can define *mutual information* (mainly in text problems):

is all the results thatcan get (usually), is the same. The probabilities can be estimated from training set.

Through information theory, we can also re-write the mutual information as a *KL(Kullback-Leibler)-divergence*:

Informally, this equation gives a measure of how different the probability distributions are. It is clearly that if and are independent random variables, the KL-divergence between will be 0.

Finally when we pick top features, we can also choose them using cross validation as measure!

**Lesson 11**

*Outline this Lesson:*

Learning Theory

1. Bayesian statistics & regularization

2. Digression: Online learning

3. Advice for applying ML algorithms

① Bayesian statistics & regularization

*“Frequentist”* view (e.g. Linear regression):

*\*“Bayesian”* view:

Practically,

*\*Regularization can always be transformed by “loss + prior” type!*

proof as follows:

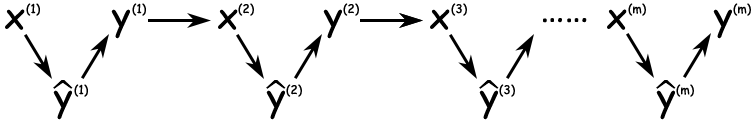
through **Lesson 3**, we can accordingly know that

🡪

🡪

🡪

② Digression: Online learning



Online error can also solve some incredible problem, such as when we use perceptron algorithm but, if it is proved that positive and negative examples are separated by a margin, perceptron algorithm can still converge to digital dimensional space. Total online error is at most

\*③ Advice for applying ML algorithms

Notes here are experiential for applying, might not suitable for researching. Some of these advice is debatable, but still make a dent in parts of problems.

(1) diagnostics for debugging learning algorithms

#1 Someone used Bayesian logistic regression to build up an anti-spam, which contains a small set of words as features, but got unacceptable 20% (or more) test error.

a) Common approach: Just try in different ways (quite randomly)

- Try getting more training examples.

- Try a smaller set of features.

- Try a larger set of features.

- Try changing the features: Email header vs. email body features.

- Run GD for more iterations.

- Try Newton’s method.

- Use a different value for .

- Try using an SVM.

This approach is time-consuming, gambly, sometimes it might work though.

b) Better approach: Diagnose and repair

- Run diagnostics to figure out what the problem truly is.

- Fix (or precisely optimize) whatever the problem is.

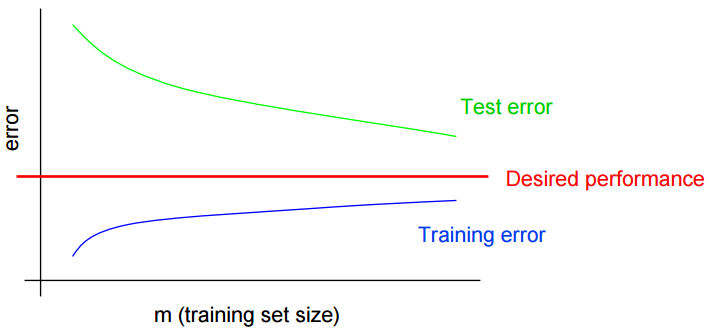
🡪 1) Suppose the problem is either (*bias vs. variance diagnostics*):

- Overfitting (high variance).

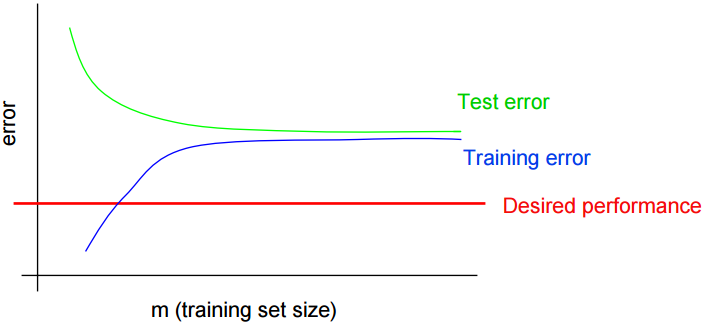
- Too few features to classify spam (high bias).

2) Diagnostic:

- Variance: Training error will be much lower than test error.



- Bias: Training error will also be high (although both are very closed of each other).



here, we can know what fixes to try:

- Try getting more training examples. Fixes high variance.

- Try a smaller set of features. Fixes high variance.

- Try a larger set of features. Fixes high bias.

- Try email header features. Fixes high bias.

For other problems, it’s usually up to ingenuity to design effective and unique diagnostics to figure out what’s wrong.

#2 Someone uses Bayesian logistic regression to build up an anti-spam, which gets 2% error on spam, and 2% error on non-spam. (Unacceptable error on non-spam.) He also applies SVM using a linear kernel, which gets 10% error on spam, and 0.01% error on non-spam (acceptable). Because of some reasons (like computational efficiency etc.), he chooses to use the former algorithm.

We can figure out lots of potential improvements (or questions) like:

- Is the algorithm converging?

- Optimize the right function?

- If using weight, need weights higher for non-spam than spam?

- Correct value for in Bayesian logistic regression?

- Correct value for in SVM? …

And whatever reason, we really want to deploy Bayesian logistic regression, even though SVM does much better for this application.

*Q: What to do next?*

BLR tries to maximize:

Diagnostics:

- Problem is with optimization algorithm.

It means that BLR tries to maximizebut fails, because using SVM’s parameterscan be larger than BLR’s. Obviously algorithm doesn’t converge well.

- Problem is with optimization objective function.

It shows that the SVM, which does worse on, actually does better on weighted accuracy. In other words, maximizingdoesn’t really correspond that well to maximizing. This confirms that if you care about,is the wrong function to be maximizing.

here, we can know what fixes to try:

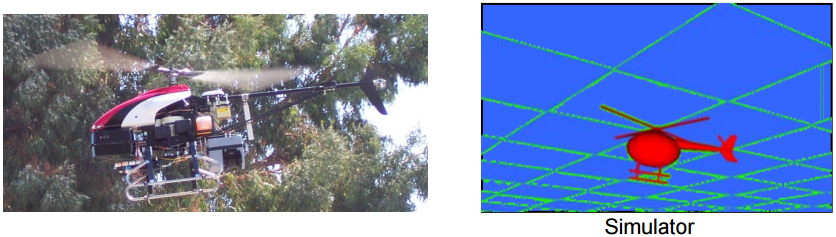
- Run GD for more iterations. Fixes optimization algorithm.

- Try Newton’s method. Fixes optimization algorithm.

- Use a different value for . Fixes optimization objective.

- Try using an SVM. Fixes optimization objective.

#3 Andrew Ng’s project on reinforcement learning: The Stanford Autonomous Helicopter.



Build a simulator of helicopter 🡪 Choose a cost function🡪 minimizeto get

Suppose the resulting controller parameters gives much worse performance than human pilot. *What to do next?*

- Improve simulator?

- Modify cost function?

- Modify RL algorithm?

Diagnostics:

- Ifflies well in simulation, but not in reality. 🡪 Problem in simulation.

Letbe the human control policy.

- If, cost function is failed to minimize. 🡪 Problem in RL algorithm.

- If, it means that minimizing doesn’t correspond to good autonomous flight. 🡪 Problem in cost function.

Conclusions:

We’ve got 3 different diagnostics of different learning problems. Practically, it’s just a little bit useful in some ways because it is quite often to come up with our own diagnostics (via ingenuity) to figure out what’s happening in different learning problems.

Moreover, even if a learning algorithm is working well, we can also run diagnostics to help us understand the algorithm further. This is useful for:

- Understanding application problems further, even getting an intuitive understand of what works and what doesn’t work.

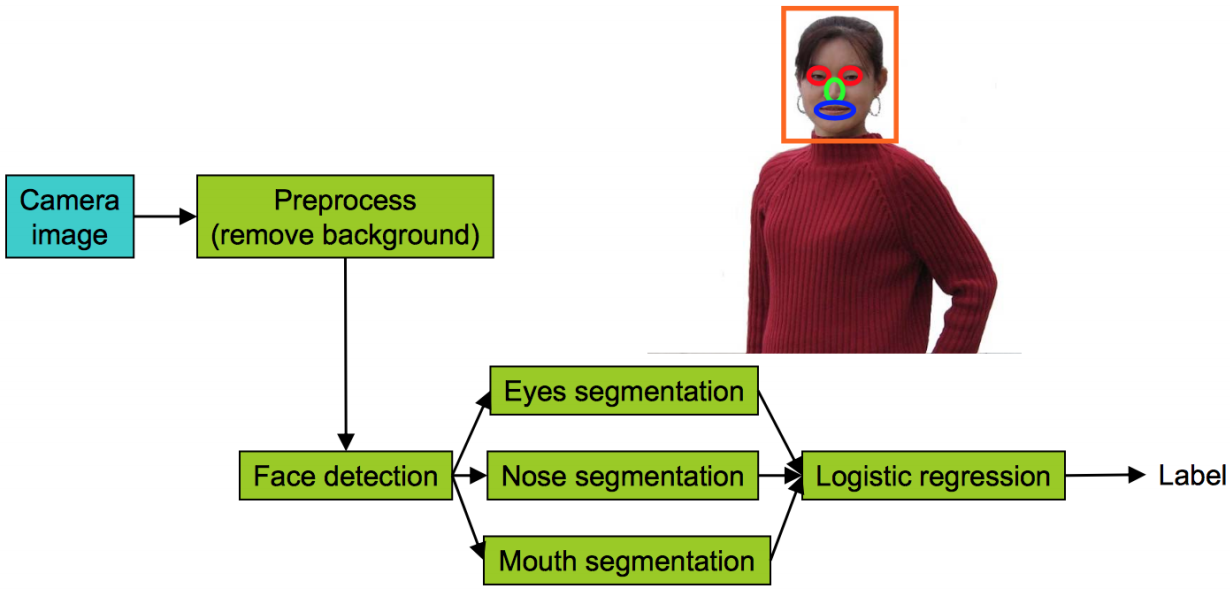
- Writing research papers: enrich insight about the problem and justify research claims.

- Making sense and being convincible when explain the core algorithm to others.

🡪 So, we need *error analysis* to understand what sources of error are!

(2) error analysis & ablative analysis

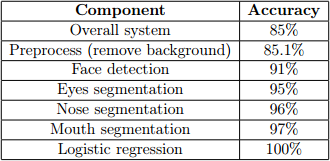
a) error analysis



E.g. A pipeline of Face recognition from images (not quite formal)

Suppose we have a pipeline like upward picture. Now the recognition accuracy of overall system is only 85%.

What we can do in brief, plug in *ground-truth* for each component, which means the perfect output of each component however it’s got (like using PS, coding by hand etc.), see how accuracy changes. Perhaps the changes can be noted as:



And then, find the several maximum gaps between and improve them first. In this case, we know that we have most room for improvement in face detection and eyes segmentation.

b) ablative analysis

Compared with error analysis, ablative analysis is the opposite strategy which tries to explain the difference between some baseline and current performance.

E.g. A good anti-spam classifier by adding lots of clever features to logistic regression:

- Spelling correction.

- Sender host features.

- Email header features.

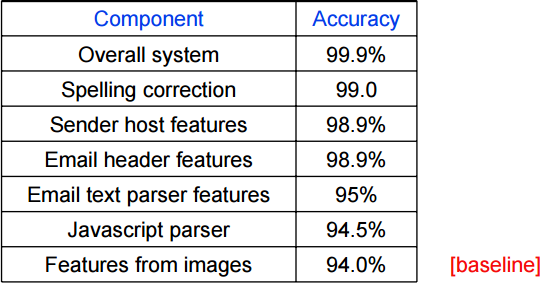
- Email text parser features.

- Javascript parser.

- Features from images.

Q: How much did each of these components really help?

Suppose we apply a simple logistic regression without any clever features get 94% performance. In ablative analysis, just remove components from the system **one at a time** (not only one-by-one, one-out-others-in if suitable is ok) to see how it breaks. Perhaps the changes can be noted as:



Accordingly, we can also find out the several maximum gaps between and figure out what is the core component of the system. In this case, it shows that email text parser features contribute for the most of the improvement.

From the discussion above, we can reorganize the both analysis briefly as:

|  |  |  |
| --- | --- | --- |
| Type | **Error Analysis** | **Ablative Analysis** |
| How | 1. Select suitable components  2. Make each “perfect” (*ground-truth*), record the changes  3. Find out the most improving parts, focus on those | 1. Select suitable components  2.Remove each in some rules, record the changes  3. Find out the most decreasing parts, pay attention on those |
| Situation | Improve the algorithm or debug it. | Analysis improved algorithm, find out the core change(s). |

(3) how to get started on a learning problem

There are two typical approaches to applying learning algorithms:

#1 Design very carefully, then implement it.

Benefit: Nicer, perhaps more scalable algorithms. May come up with new, elegant, learning algorithms; contribute to basic research in machine learning.

Risk: *Premature* (*statistical*) *optimization*.

#2 Build a quick-and-dirty prototype, diagnose and fix it.

Benefit: Will often get application problem working more quickly. Faster time to market.

Risk: Time-consuming debugging and testing (if not quite experienced in some part).