Microsoft

Microsoft Research Faculty Summit 2013

Microsoft

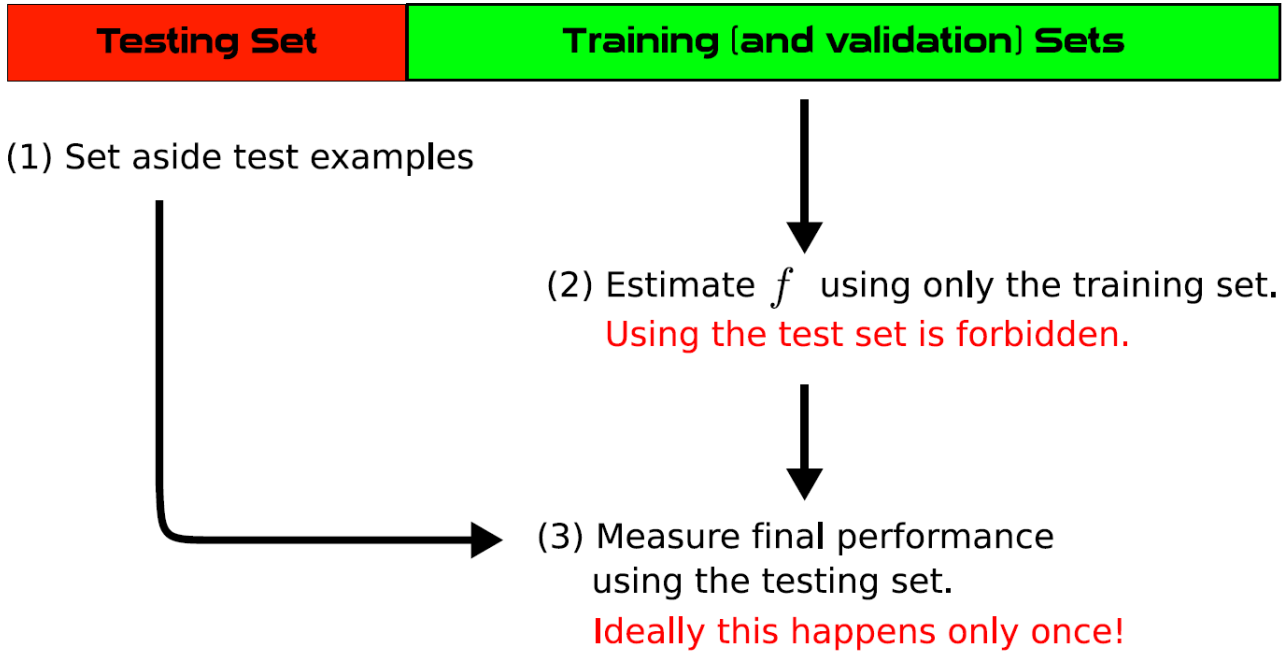
Large-scale learning revisited

Leon Bottou

Microsoft Research
Faculty Summit

1 three influential ideas in machine learning.

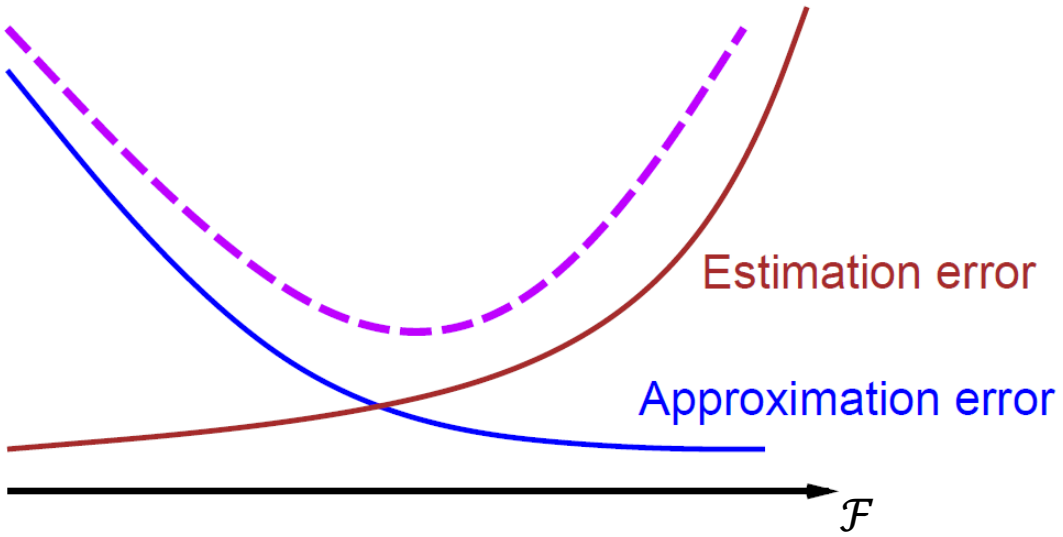
I. independent and identically distributed data



- This experimental paradigm has driven machine learning progress.
- The essential assumption is that training and testing data are exchangeable, e.g., follow the same distribution.

II. model selection tradeoffs

$$E(f_n) - E(f^*) = \left(E(f_F^*) - E(f^*)\right) \quad \text{Approximation Error}$$
$$\quad + \left(E(f_n) - E(f_{\mathcal{F}}^*)\right) \quad \text{Estimation Error}$$



How complex a model can you afford with your data?

III. “Vapnik’s razor”

“ When solving a (learning) problem of interest, do not solve a more complex problem as an intermediate step. ”

How complex a model can you afford with your data? (again)

- To classify patterns, use a model that outputs a class and nothing else.
- To achieve something more complex,
  i. carefully define the problem,
  ii. solve the problem and nothing else.

conceptual viewpoints in machine learning

	same distribution	capacity tradeoff	Vapnik's razor
statistical learning theory (ERM, SRM, SVM, …)	yes	yes	yes
Bayesian learning (generative models, priors, …)	yes	yes	no (1)
algorithmic learning theory (regret bounds, …)	no (2)	yes	yes

1) See the discriminant versus generative debate. On the one hand, some authors see generative models as an implicit form of regularization. On the other hand, generative models are often appealing because they easy to combine, unlike strict discriminant classifiers.

2) Online regret bounds express how a learning algorithm performs relative to a class of competitors. Although they do not depend on i.i.d. assumptions, they lose value when none of the competitors works well, for instance because the data is too far from i.i.d..

2 large-scale learning tradeoffs.

statistics and computation

Statistical Perspective

- It is good to optimize an objective function that ensures a fast estimation rate when the number of examples increases.

Optimization Perspective

- To efficiently solve large problems, it is preferable to choose an optimization algorithm with strong convergence properties.

Incorrect Conclusion

- To address large-scale learning problems, use the best algorithm to optimize an objective function with fast estimation rates. *

learning with approximate optimization

We compute $f_n = \operatorname{argmin}_{f \in \mathcal{F}} E_n(f)$ (best training error)

… which is an approximation of $f_{\mathcal{F}}$, (best test error in $\mathcal{F}$)

… which is an approximation of f^*. (best test error)

No need to compute f_n exactly.

Let's compute an approximate optimum $\tilde{f}_n$ such that

$$E_n(\tilde{f}_n) \leq E_n(f_n) + \rho$$

error decomposition

$$E(\tilde{f}_n) - E(f^*) = E(f^*_{\mathcal{F}}) - E(f^*) \quad \text{Approximation error} \quad \mathcal{E}_{app}$$
$$+ E(f_n) - E(f^*_{\mathcal{F}}) \quad \text{Estimation error} \quad \mathcal{E}_{est}$$
$$+ E(\tilde{f}_n) - E(f_n) \quad \text{Optimization error} \quad \mathcal{E}_{opt}$$

Problem:

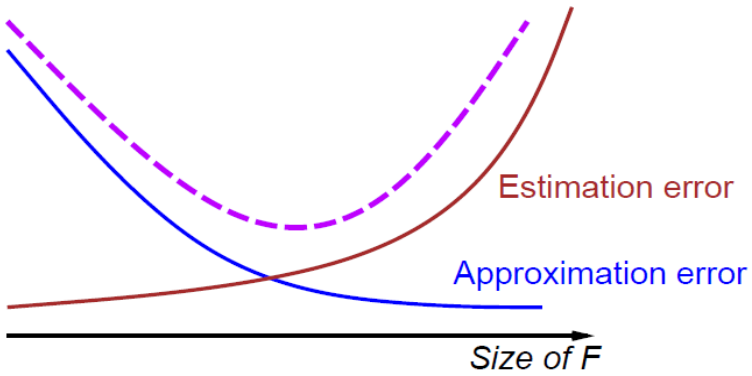
Choose $\mathcal{F}$, n, and ρ to make this as small as possible,

subject to budget constraints $\begin{cases} \text{max number of examples } n \\ \text{max computing time } T \end{cases}$

small-scale learning

The active budget constraint is the number of examples.

- To reduce the estimation error, take n as large as the budget allows.
- To reduce the optimization error to zero, take $\rho = 0$.
- We need to adjust the size of $\mathcal{F}$.



See Structural Risk Minimization (Vapnik 74) and later works.

large-scale learning

The active budget constraint is the computing time.

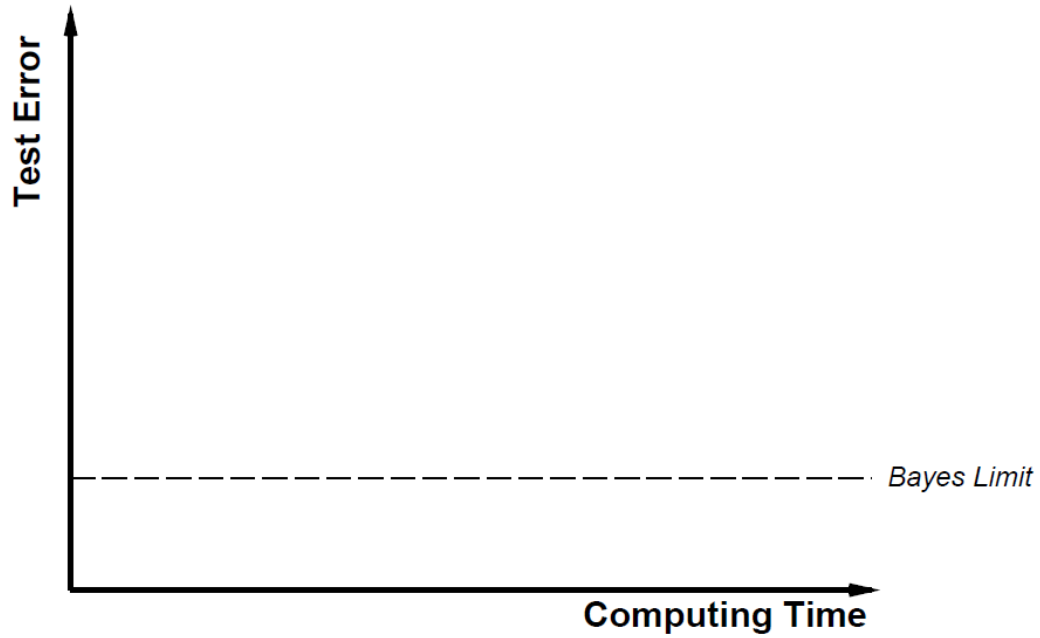
- More complicated tradeoffs.
  The computing time depends on the three variables: $\mathcal{F}$, n, and ρ.

- Example.
  If we choose ρ small, we decrease the optimization error. But we must also decrease $\mathcal{F}$ and/or n with adverse effects on the estimation and approximation errors.

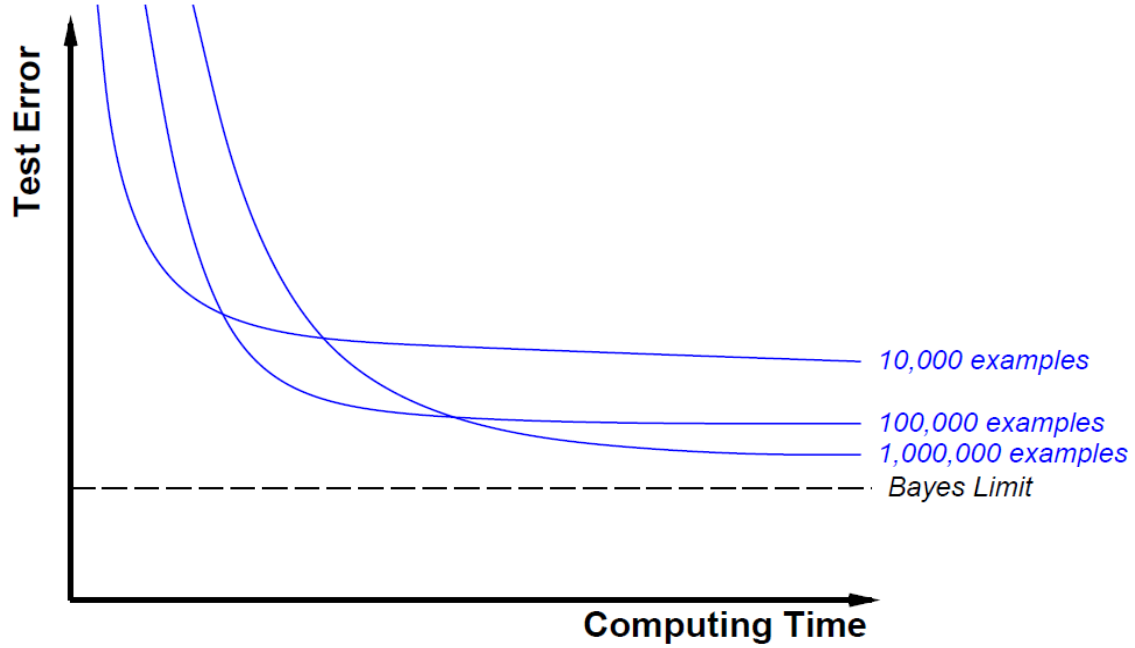
- The exact tradeoff depends on the optimization algorithm.

- We can compare optimization algorithms rigorously.

test error versus training time

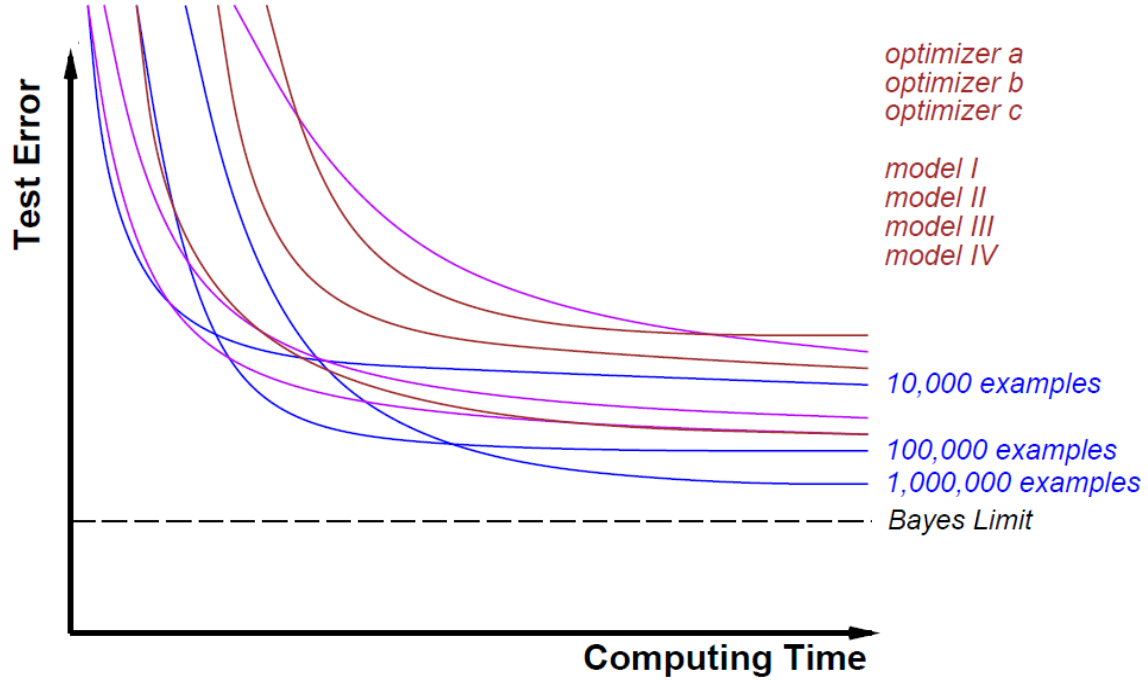


test error versus training time



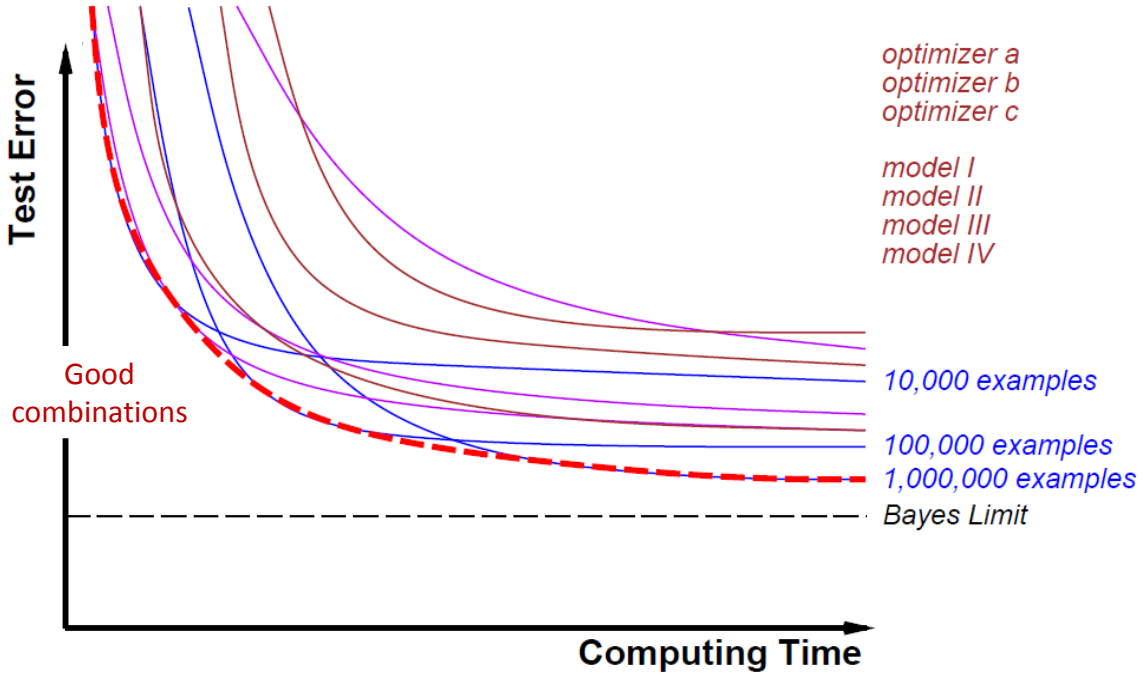
- Vary the number of examples

test error versus training time



- Vary the number of examples, the model, the algorithm

test error versus training time



- Optimal combination depends on training time budget.

analysis of a simple case

Fix the family $\mathcal{F}$ of functions

- Linearly parameterized, smooth convex loss, …

Compare four iterative optimization algorithms

1. Gradient descent (GD)
2. Second order gradient descent (2GD)
3. Stochastic gradient descent (SGD)
4. Second order stochastic gradient descent (2SGD)

analysis of a simple case

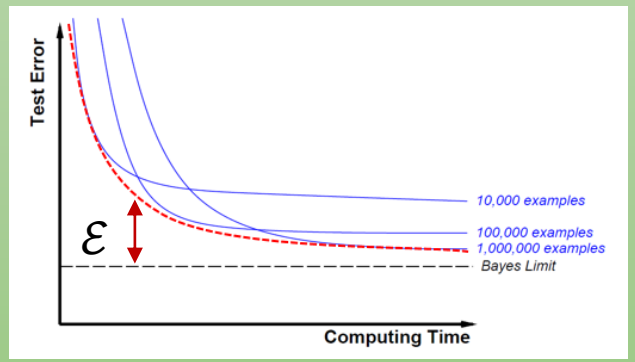
	GD	2GD	SGD	2SGD
Time per iteration $\sim$	n	n	1	1
Iteration to accuracy ρ $\sim$	$\log\frac{1}{\rho}$	$\log\log\frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to accuracy ρ $\sim$	$n\log\frac{1}{\rho}$	$n\log\log\frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$

Newton (2GD) optimizes much faster than simple gradient descent (GD).

Stochastic algorithms seem hopeless

analysis of a simple case

	GD	2GD	SGD	2SGD
Time per iteration ~	n	n	1	1
Iteration to accuracy ρ ~	$\log\frac{1}{\rho}$	$\log\log\frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to accuracy ρ ~	$n\log\frac{1}{\rho}$	$n\log\log\frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to excess error $\mathcal{E}$ ~	$\frac{1}{\mathcal{E}^\alpha}\log\frac{1}{\mathcal{E}}$	$\frac{1}{\mathcal{E}^\alpha}\log\frac{1}{\mathcal{E}}\log\log\frac{1}{\mathcal{E}}$	$\frac{1}{\mathcal{E}}$	$\frac{1}{\mathcal{E}}$



$1 \le \alpha \le 2$

Stochastic algorithms now look very good!

SGD and 2SGD only differ by constants

text categorization with a linear svm

- **Dataset**
  - Reuters RCV1 document corpus.
  - 781,265 training examples, 23,149 testing examples.
  - 47,152 TF-IDF features.
- **Task**
  - Recognizing documents of category `CCAT`.
  - Minimize $\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\lambda}{2}w^2 + \ell(w\,x_i + b,\, y_i)\right)$.
  - Update $w \leftarrow w - \eta_t \nabla(w_t, x_t, y_t) = w - \eta_t \left(\lambda w + \frac{\partial \ell(w\,x_t + b,\, y_t)}{\partial w}\right)$

Same setup as (Shalev-Schwartz et al., 2007) but plain SGD.

text categorization with a linear svm

- **Results: Linear SVM**

  $\ell(\hat{y}, y) = \max\{0, 1 - y\hat{y}\}$ $\lambda = 0.0001$

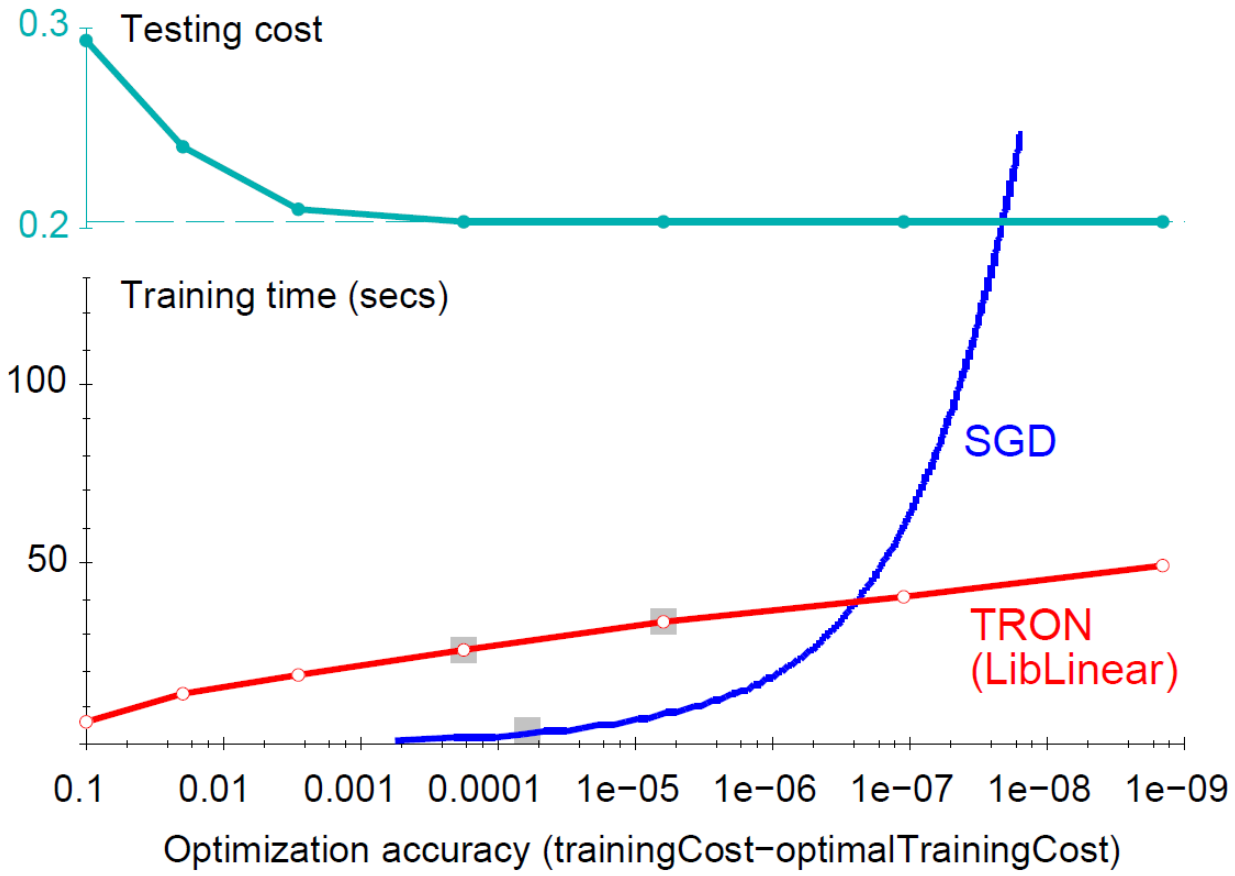
	Training Time	Primal cost	Test Error
SVMLight	23,642 secs	*0.2275*	*6.02%*
SVMPerf	66 secs	0.2278	6.03%
SGD	1.4 secs	*0.2275*	*6.02%*

- **Results: Log-Loss Classifier**

  $\ell(\hat{y}, y) = log(1 + exp(-y\hat{y}))$ $\lambda = 0.00001$

	Training Time	Primal cost	Test Error
TRON(LibLinear, $\varepsilon = 0.01$)	30 secs	0.18907	*5.68%*
TRON(LibLinear, $\varepsilon = 0.001$)	44 secs	*0.18890*	5.70%
SGD	2.3 secs	0.18893	*5.66%*

text categorization with a linear svm



the tradeoffs of large-scale learning

Small-scale learning ≠ large-scale learning

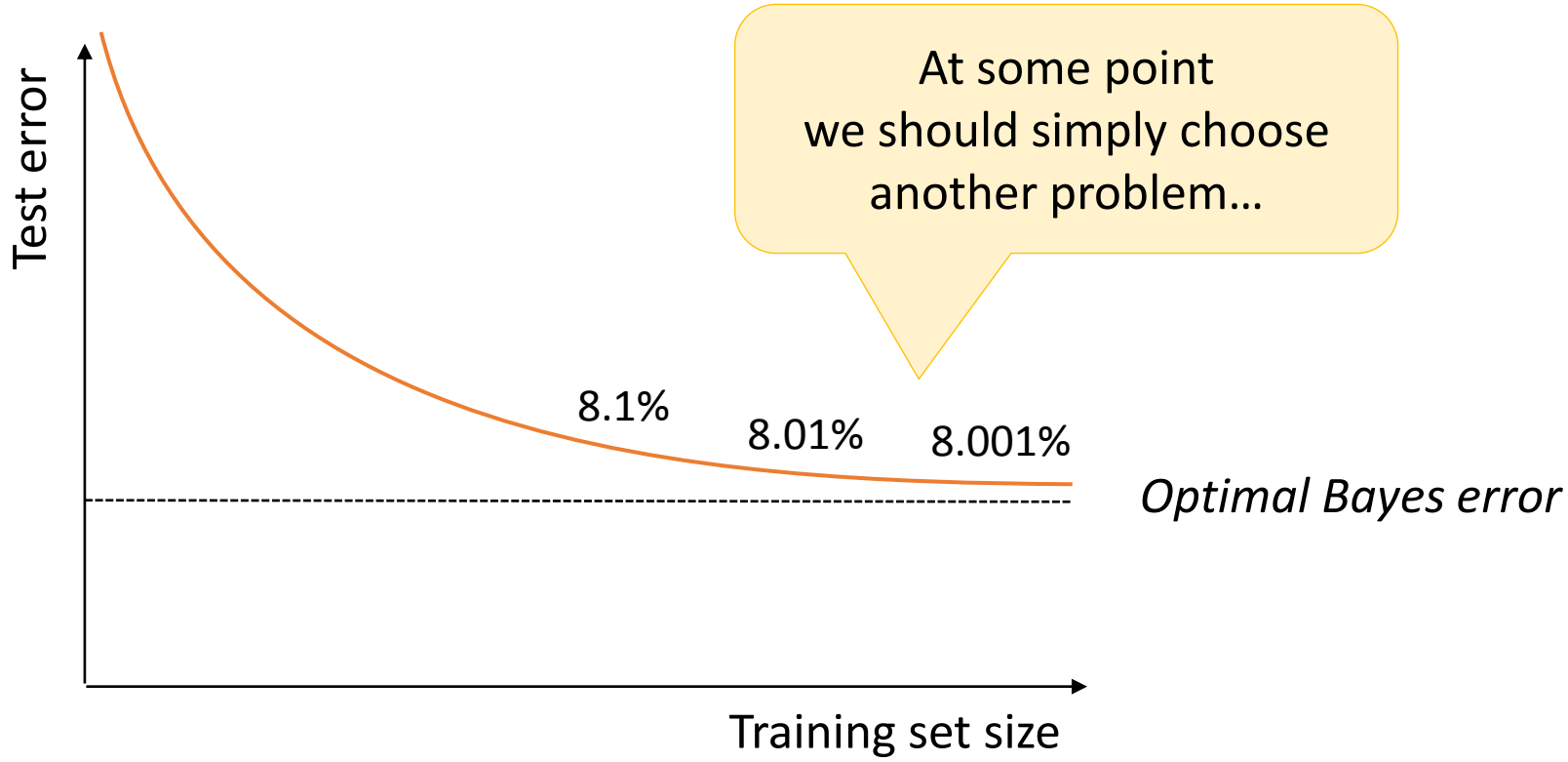
- Large-scale learning involves more complex tradeoffs
  that depends on the properties of the optimization algorithm.

Good optimization algorithm ≠ good learning algorithm

- Mediocre optimization algorithms (e.g., SGD)
  can outperform sophisticated optimization algorithms
  on large-scale learning problems

3- breadth versus accuracy

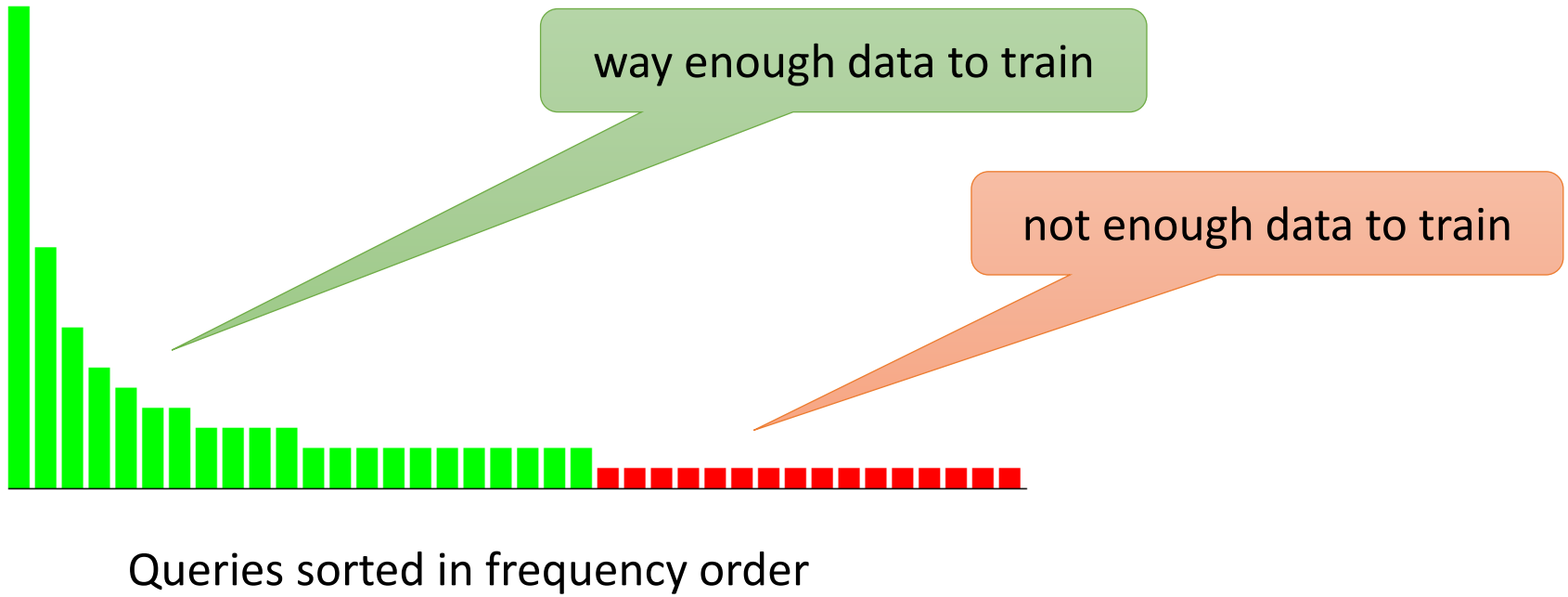
diminishing returns



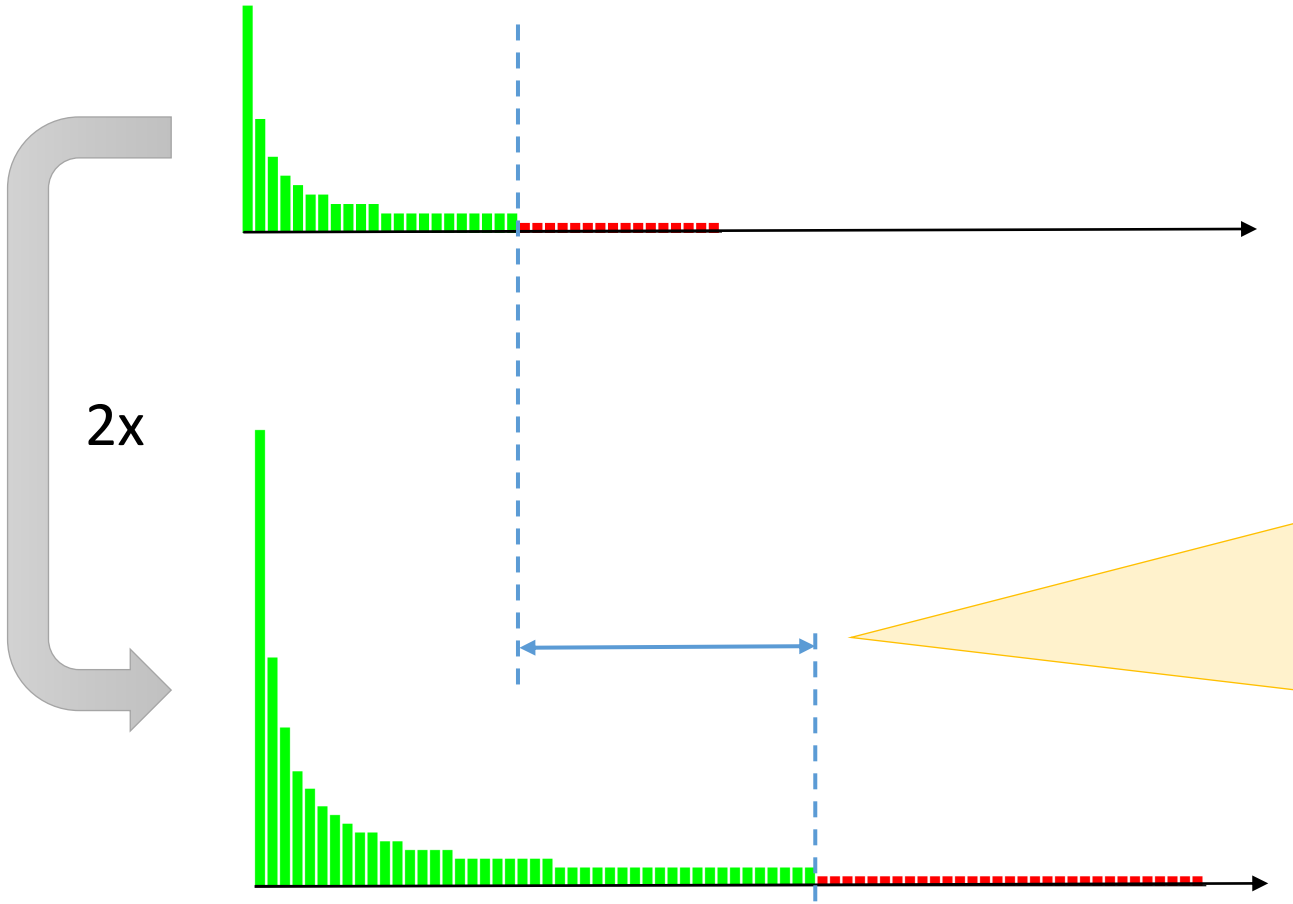
- Accuracy improvements cannot justify the computational cost forever.
- Why then use very large training sets ?

Zipf distributed data

- Roughly half of the search queries are unique.



doubling the size of the training set



Diminishing returns
for average accuracy improvements.

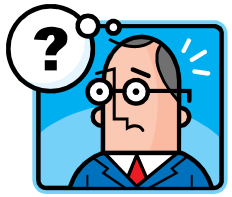
No diminishing returns
on number of queries for which we can learn correct answers.

the value of big data in machine learning

Accuracy improvements are subject to diminishing returns.
Breadth improvements are not subject to diminishing returns.

"How accurately do we recognize an object category?"
vs. "How many categories do we recognize well enough?"

Should we optimize a different criterion?



How does this helps if average accuracy is what we care about ?

identically distributed examples ?

Data collection in traditional machine learning

- Training data collection for real-life machine learning is difficult.
  The data distribution must reflect the operational conditions.
- The i.i.d. assumption is not automatically satisfied.
  It happens through manual data curation.

Data collection in big data machine learning

- Big data exists because data collection is automated.
- No manual curation to enforce the identical distribution assumption.
- The output of the machine learning system frequently
  impacts the distribution of future training data.

dealing with covariate shifts

$$P(X,Y) = P(Y|X)\ P(X)$$

We want to model $Y \approx f(X)$.
We must assume that the training data describes $P(Y|X)$ well enough.

We cannot trust $P(X)$.
We want to train a system robust to $P(X)$ changes.

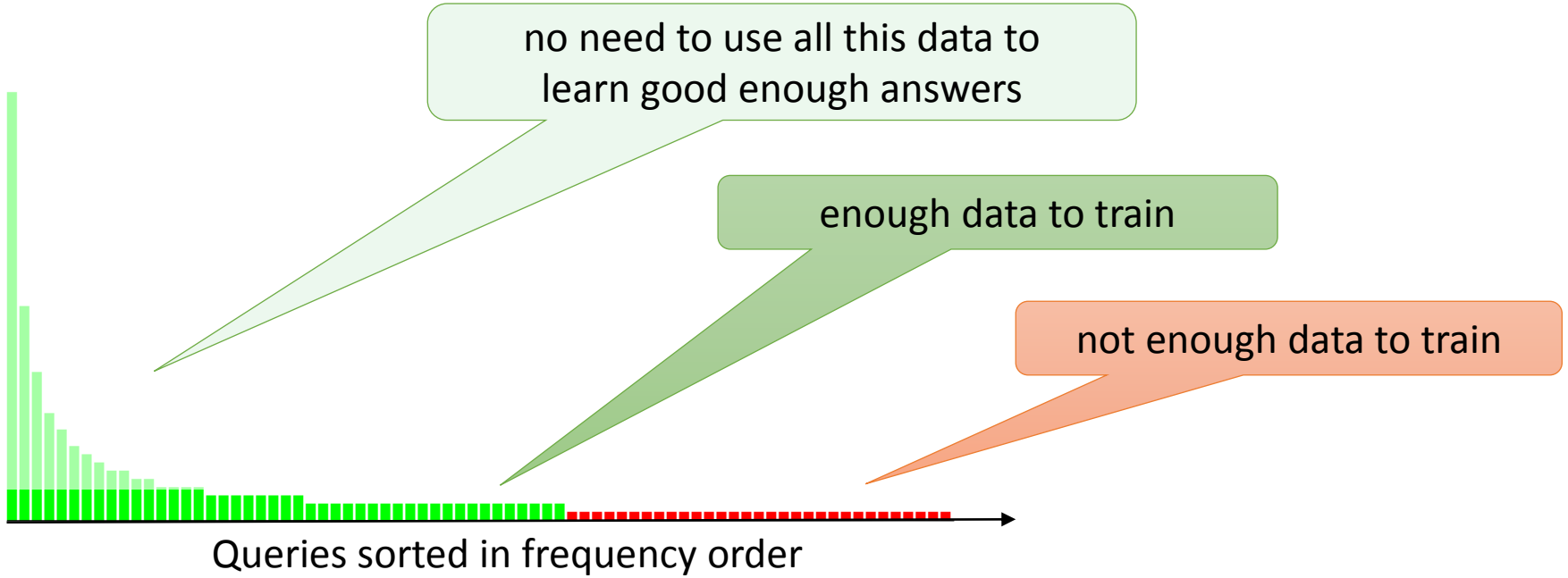
Minimizing the training set error

- Approximation errors are pushed towards patterns X with low probability.
- What if these patterns occur more frequently at testing time?

Maximize the "diversity" of patterns that are recognized well enough.

- Yields a solution that is more robust to $P(X)$ changes.

scalability opportunities



- No need to consider all examples of already known queries.
- Best is to focus on queries near the boundary of the known area.
- Curriculum learning and active learning come naturally in this context.
- Scalability gains across the board.

3. deep learning, and transfer learning

engineering machine learning

Reading check amounts

– Input $\mathcal{X}$: Scanned check images.
– Output $\mathcal{Y}$: Positive numbers with two decimals.

Training a model

– Can we train a model using examples $(x_1, y_1) \ldots (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$?
– Possibly (we did not really try.)
– This would require excessive numbers of labeled examples.
– This would require excessive computation time.

engineering machine learning

Identify subproblems

- Locate amount fields.
- Segment characters in amount fields.
- Recognize isolated characters.
- Translate character strings into an amount.

Define a submodel for each subproblem

- Fairly complicated recognition models with large parameter vectors.
- Highly engineered location and segmentation models with only a few adjustable thresholds.

Collect and label data for each subproblem

- Lots of manual work.
- Manual labor is not very expensive. . .

engineering machine learning

Training strategies

– Independent training
  Train each submodel separately.
– Sequential training (better)
  Label outputs of submodel n and train submodel $n+1$.
– Global training (even better)
  Pre-train with sequential training.
  Simultaneously train all submodels with examples from $\mathcal{X} \times \mathcal{Y}$.

Issues

– The structure of the global model changes with the data.
  e.g. managing field location and segmentation hypotheses.
– The composition of submodels has nontrivial aspects.
  e.g. the label-bias problem.

deep learning

Deep learning (simplified)

– Pre-train with sequential unsupervised training
  Collect outputs of submodel n
  Train submodel $n+1$ with unsupervised criterion
– Tune with global training
  Consider all submodels as a single statistical model.
  Train with examples from $\mathcal{X} \times \mathcal{Y}$.

The deep learning surprise:

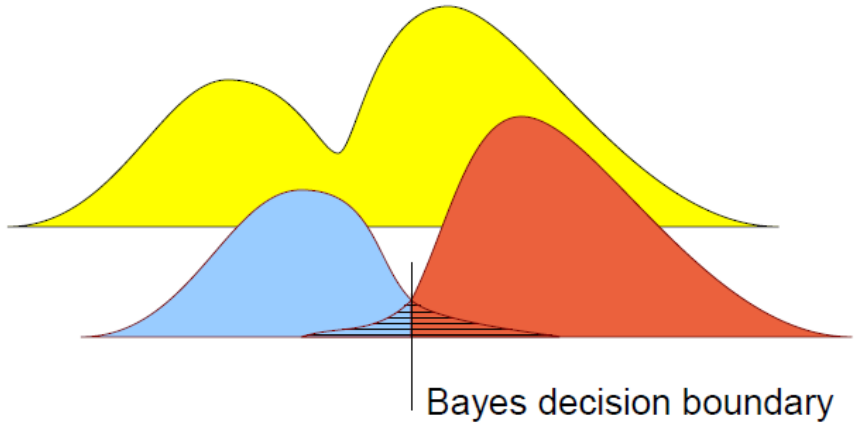
– Generic unsupervised subtasks work remarkably well.
– Little need to define subtasks using engineering knowledge.
– Little need to collect labeled data for all submodels.

Engineering learning systems is easier than we thought!

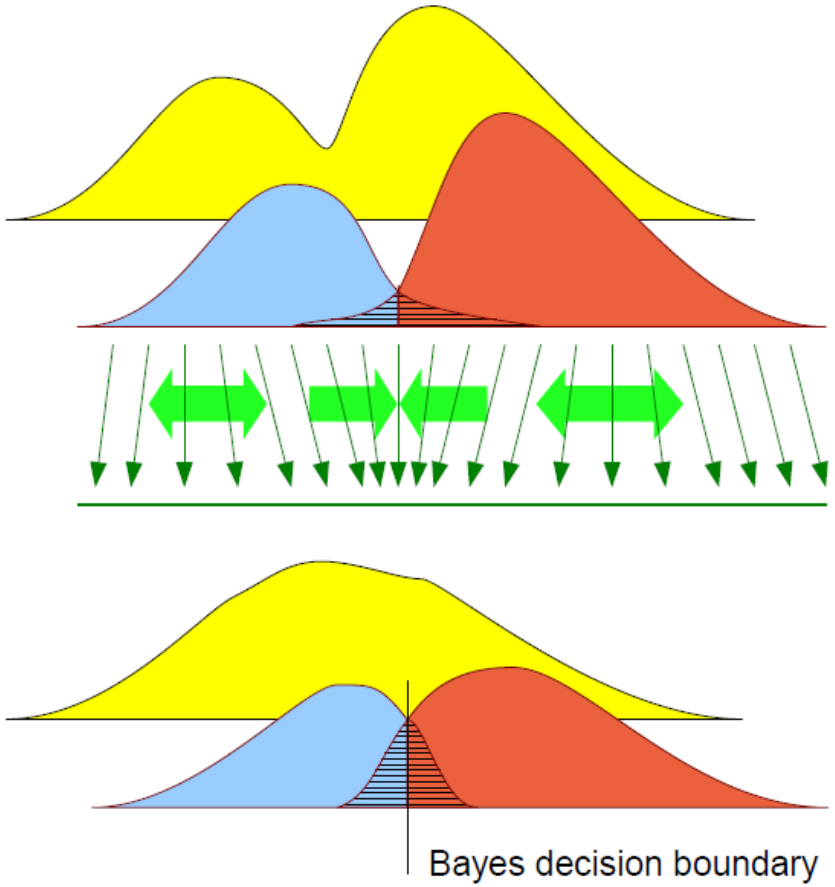
unsupervised learning

What is a cluster?

– Assumption: the shape of the density reveals the underlying categories.



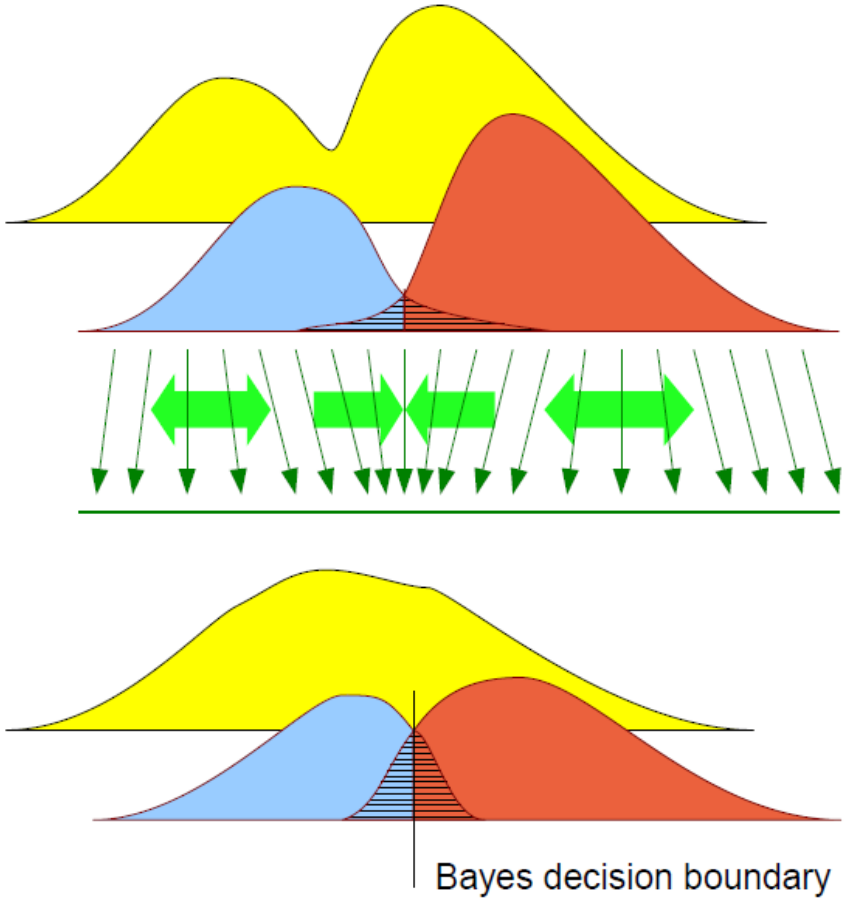
unsupervised learning



Input space transforms

- Categories are invariant.
- Bayes rate is invariant.
- Clustering is not invariant.

unsupervised learning



Clustering revisited

– Clustering is the expression of the prior knowledge encoded by our choice of input representation.

Unsupervised learning

– Comparable to using really cheap labels:
"x_1 and x_2 are close".
"x_1 and x_3 are not close".

auxiliary tasks

The price of labels

Interesting task $\iff$ Scarce labeled examples.
Uninteresting task $\iff$ Abundant labeled examples.

Auxilliary tasks

– **"In the vicinity of an interesting task** (with expensive labels,)
there often are less interesting tasks (with cheap labels,)
that we can put to good use."

– Unsupervised learning is just one of them (with trivial labels.)

– **Deep-learning, semi-supervised learning, and transfer learning are three facets of the same thing.**

e.g. (Weston et al., 2008)

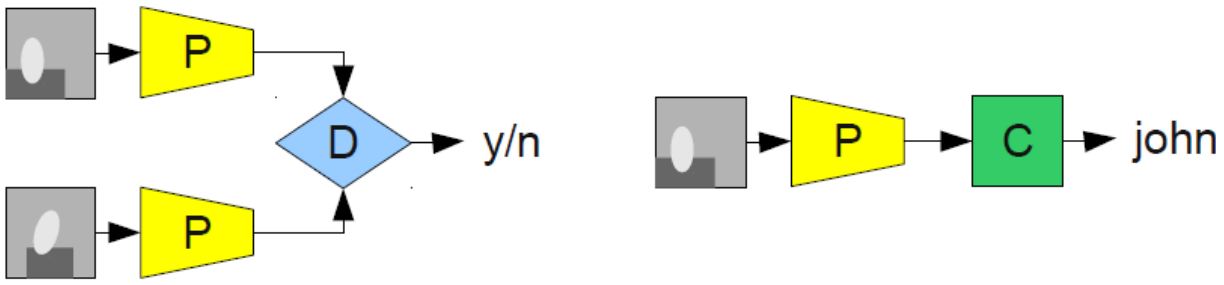
example – face recognition

Interesting problem

– Recognizing the faces of one million persons.
– How many labeled images per person will we get?

Related but less interesting problem

– Are two face images representing the same person?
– Abundant (but noisy) examples:
  ◇ Two faces in the same image are likely to be different persons.
  ◇ Faces in successive frames are likely to be the same person.

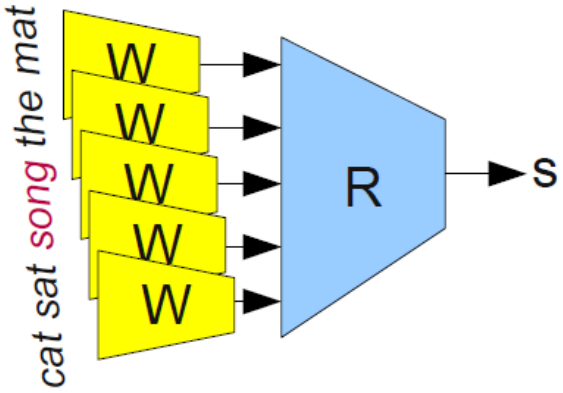


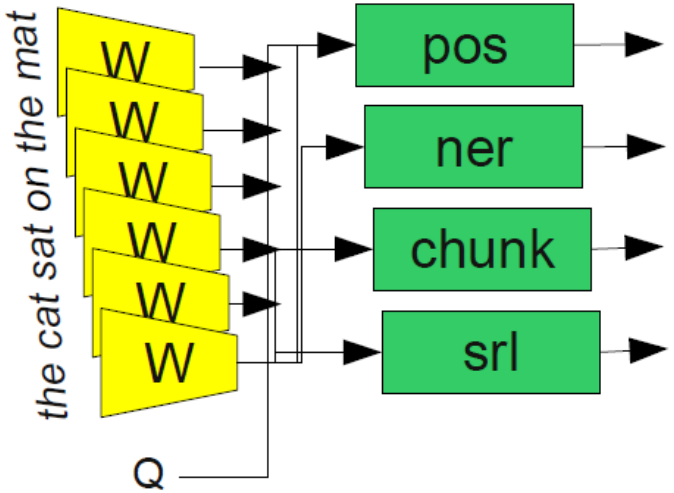
(Matt Miller, NECLA, 2006)

example – natural language tagging

Interesting problems
– Standard NLP tagging tasks.

Related but less interesting problem
– Positive examples are legal sentence segments.
– Negative examples are created by substituting the central word.
– Ranking loss.





(Collobert et al., 2008-2011)

revisiting Vapnik's razor

" When solving a (learning) problem of interest, do not solve a more complex problem as an intermediate step. "

Rationale: how complex a model can we afford with our data?

However, solving a more complex task and transferring features often allows us to leverage more data of a different nature.

- Lots of implications… ("From machine learning to machine reasoning", L.B., 2011.)

conclusion

Large-scale changes everything!