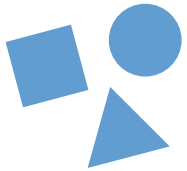
Subgradient Method Support Vector Machines for RNA-seq data

ERIC ZANDER

CS 314: DISTRIBUTED DATA MANAGEMENT

Support Vector Machines

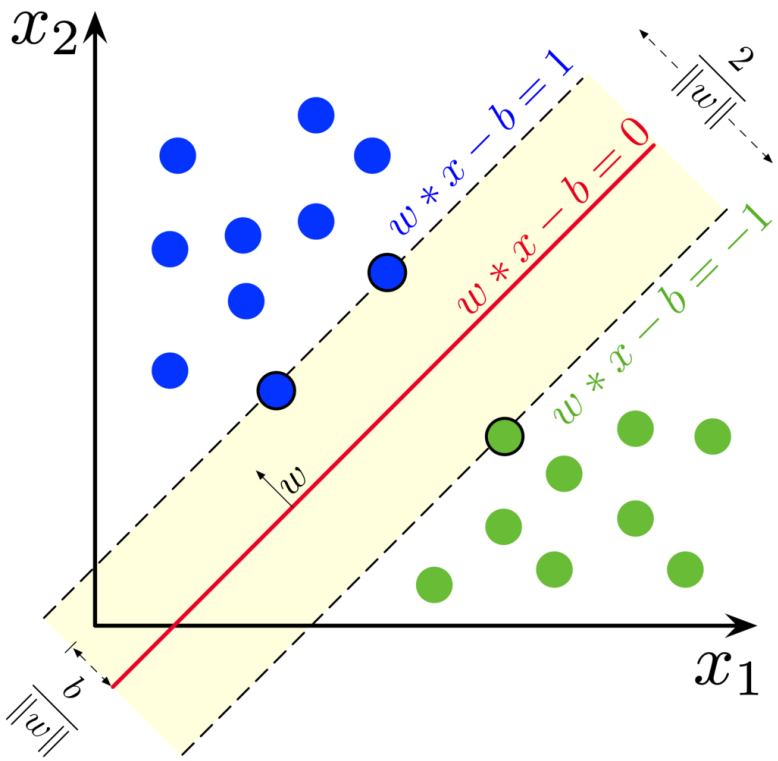
Desirable Model Characteristics



Multiclass predictions

Built-in Regularization

Potentially scalability



Support Vector Machines

- Separates data by hyperplane
- Built-in regularization
- Flexible
  - Primal and dual formulations
  - Multiple kernels
  - Scalable versions
    - Subgradient method

Subgradient Method

- Comparable (but not identical) to gradient descent/ascent
- Alternative to quadratic programming approaches to optimizing SVM
  - Potentially faster
  - Lends itself to parallelization and online learning
  - Can be less stable
- Multiple types
  - Primal vs. dual formulation
  - Batch, mini-batch, and stochastic
  - Different optimizers
- Employed in this project to optimize Lagrange multipliers in dual formulation
  - Using gradient of L wrt. lambda/alpha
  - Hsieh *et al*.

Dual formulation of optimization problem

$$\max_{\lambda} \mathcal{L}(\lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i,j}^{N} \lambda_i \lambda_j \, y_i \, y_j \phi(x_i)^T \phi(x_j)$$

$$= \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i,j}^{N} \lambda_i \lambda_j \, y_i \, y_j \kappa(x_j, x_i)$$

```
# Get gradient of L wrt alpha
grad = 1 - y[i] * np.sum(alpha * y * K[:, i])
```

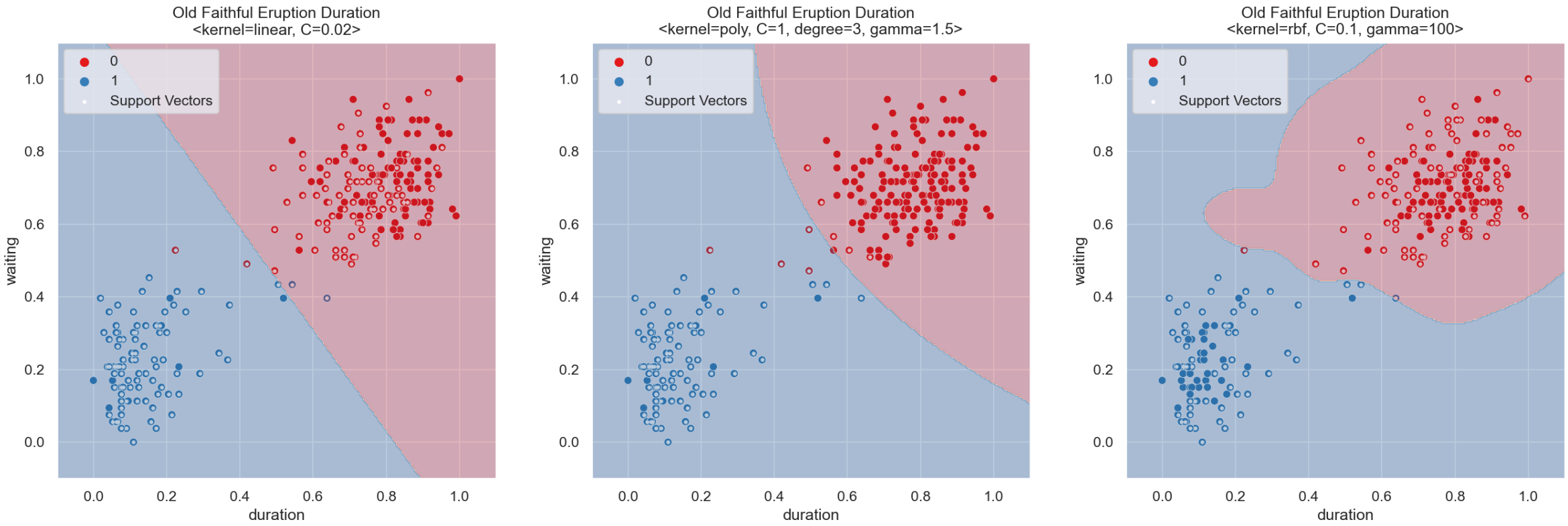
```python
# Perform optimization with subgradient method for dual formulation
for epoch in range(self.iters):
    prev_alpha = alpha.copy()

    # Get indices of random sample for stochastic/mini-batch
    batch_indices = np.random.permutation(num_samples)[:self.batch_size]

    for i in batch_indices:
        # Shrinking (leave out zero'd support vector candidates)
        if alpha[i] >= self.shrink_thresh:
            # Get gradient of L wrt alpha
            grad = 1 - y[i] * np.sum(alpha * y * K[:, i])

            # Punish non-zero lagrangian multipliers to encourage sparsity
            grad -= self.l1_reg * np.abs(alpha[i])

            # Update moment estimations for ADAM
            t += 1
            m[i] = self.beta1 * m[i] + (1 - self.beta1) * grad
            v[i] = self.beta2 * v[i] + (1 - self.beta2) * grad ** 2
            m_hat = m[i] / (1 - self.beta1 ** t)
            v_hat = v[i] / (1 - self.beta2 ** t)

            # Perform update
            alpha[i] += self.lr * m_hat / (np.sqrt(v_hat) + 1e-8)

            # Restrict alpha to [0, C]
            alpha[i] = max(0, min(self.C, alpha[i]))

    # Break if converged based on tolerance
    if np.linalg.norm(alpha - prev_alpha) < self.eps:
        if self.verbose:
            print(f"converged (epoch={epoch})")
        break
```

Partial Implementation

Goal: Optimize support vectors' Lagrange multipliers (alpha)

Binary SVM & Old Faithful

Geyser Data: Härdle, 1991



Binary SVM

High gamma for exaggerated kernel behavior

Multiclass SVM & Penguins

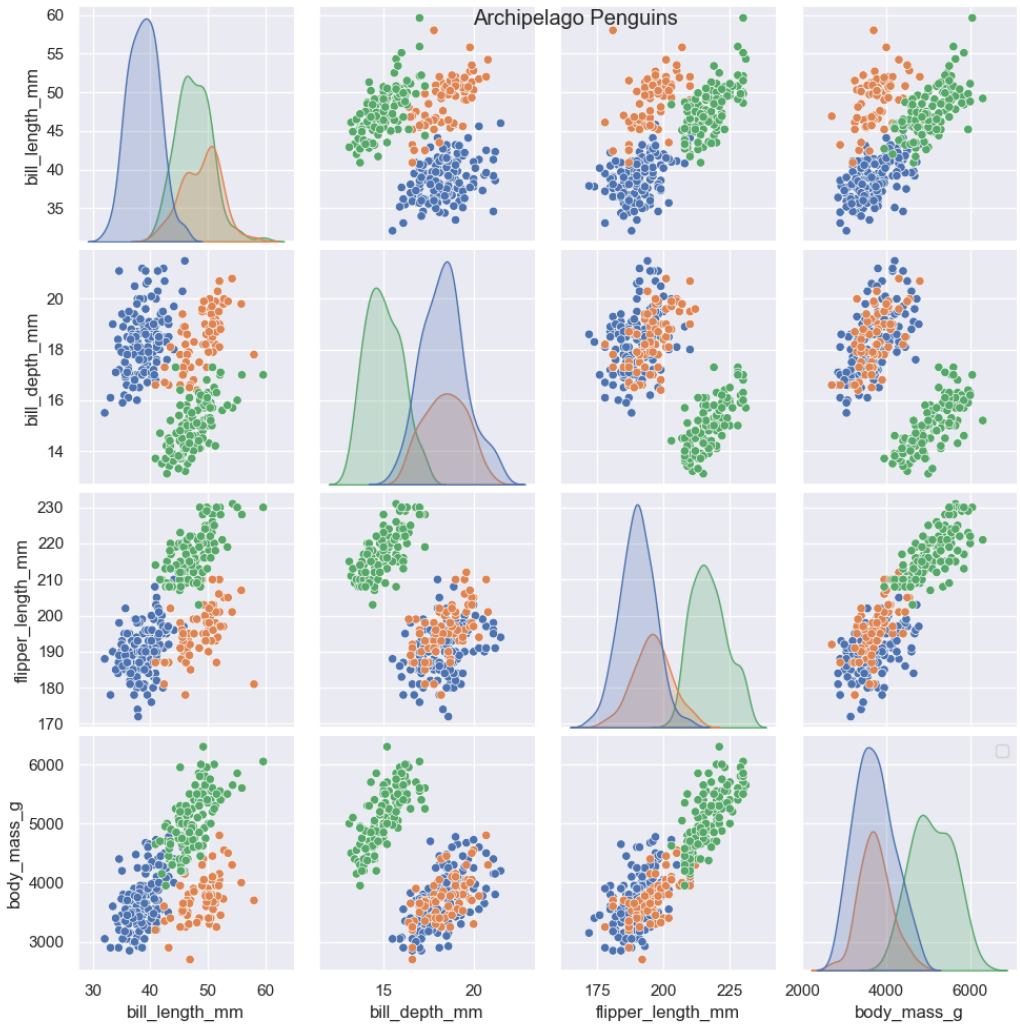
Palmer Archipelago penguin data:
Gorman et al., 2014

Penguins

- Predicted 3 species
  - Adelie, Chinstrap, Gentoo
- Used 4 numerical features

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

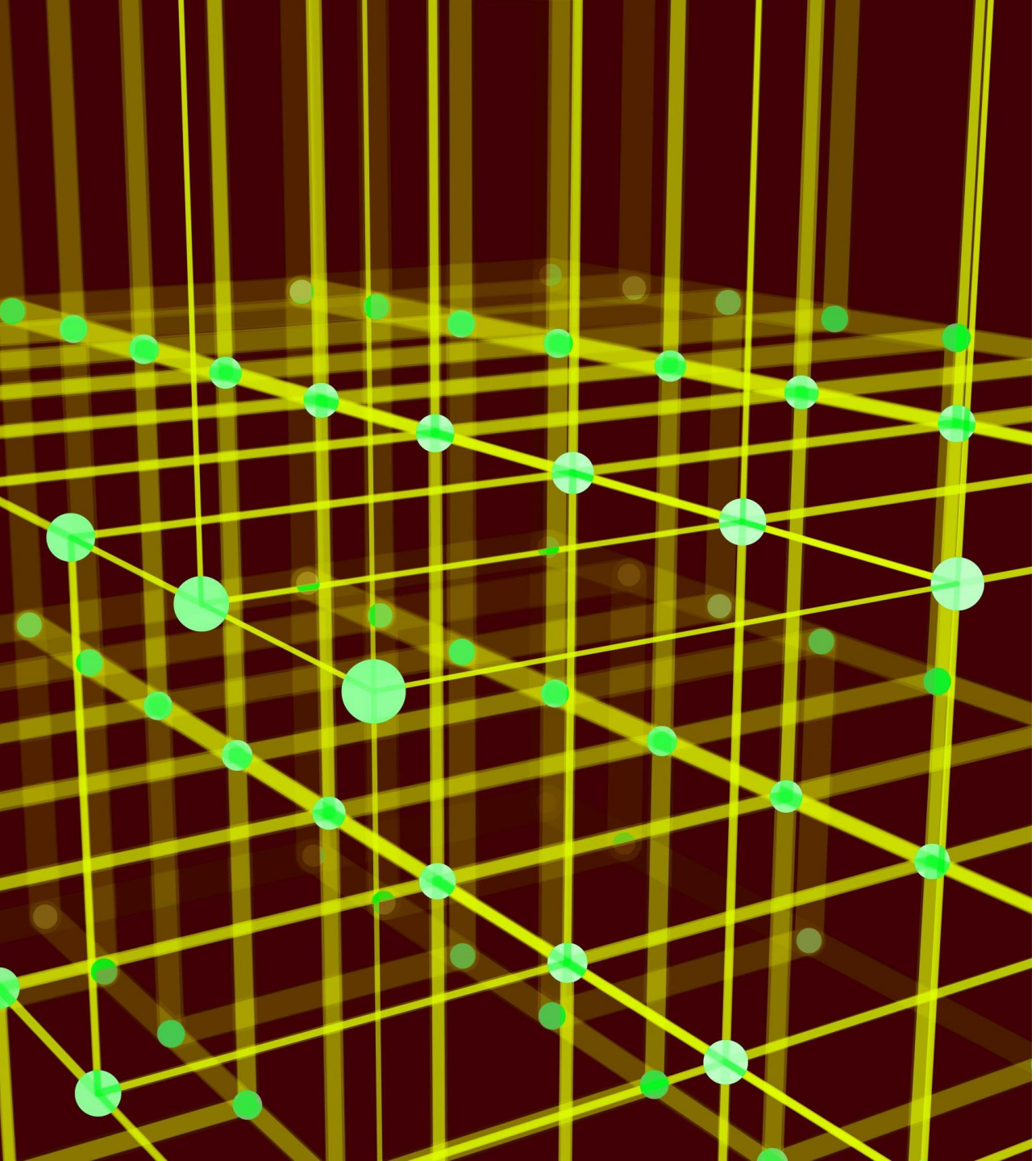
344 rows × 7 columns



```
linear
converged (epoch=364)
converged (epoch=343)
converged (epoch=223)
time: 0.2200
sup vecs: 165
accuracy: 1.0000

poly
converged (epoch=293)
converged (epoch=287)
converged (epoch=211)
time: 0.2147
sup vecs: 176
accuracy: 1.0000

rbf
converged (epoch=287)
converged (epoch=272)
converged (epoch=240)
time: 0.2116
sup vecs: 175
accuracy: 1.0000
```

Multiclass Classification

- Multiple options
  - Went with One vs. All (OvA)
    - Ease of implementation
    - Less sub-classifiers required than One vs. One
    - Example: 1 binary classifier per species
  - Uses max distance from a hyperplane
- Success with minimal tuning



TCGA RNA-Seq

TCGA data: Weinstein et al., 2013

Dataset

- RNA-seq data describes gene expression levels
- Many potential applications in medicine and elsewhere
- The Cancer Genome Atlas (TCGA)
  - Weinstein *et al.*
  - Accessed via UCI Repo
- 5 classes
  - Cancer type
- 801 samples
- 20531 input features

```
,gene_0,gene_1,gene_2,gene_3,gene_4,gene_
sample_0,0.0,2.01720929003,3.26552691165,
sample_1,0.0,0.592732094867,1.58842082049
sample_2,0.0,3.5117589779,4.32719871937,6
sample_3,0.0,3.66361787431,4.50764877794,
sample_4,0.0,2.65574107476,2.82154695883,
sample_5,0.0,3.46785331372,3.58191760772,
sample_6,0.0,1.224966365,1.69117679681,6.
sample_7,0.0,2.85485342652,1.75047787844,
sample_8,0.0,3.99212487426,2.77273024777,
sample_9,0.0,3.64249364243,4.42355800269,
sample_10,0.0,3.49207108711,3.553372792,7
sample_11,0.0,2.94118144936,2.66327629754
sample_12,0.0,3.9703475182,2.36429227014,
sample_13,0.0,1.5510483733,3.52984592804,
sample_14,0.0,1.9648421858,2.18301003676,
sample_15,0.0,2.90137860229,3.68536833781
```

```
,Class
sample_0,PRAD
sample_1,LUAD
sample_2,PRAD
sample_3,PRAD
sample_4,BRCA
sample_5,PRAD
sample_6,KIRC
sample_7,PRAD
sample_8,BRCA
sample_9,PRAD
sample_10,BRCA
sample_11,KIRC
sample_12,PRAD
sample_13,BRCA
sample_14,BRCA
sample_15,BRCA
sample_16,LUAD
sample_17,KIRC
sample_18,KIRC
sample_19,PRAD
sample_20,BRCA
sample_21,KIRC
sample_22,LUAD
```

Issues

```
UserWarning: No support vectors learned
  warnings.warn("No support vectors learned")
RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
RuntimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)
Accuracy: 0.1577
```

Polynomial

- Lagrange multipliers = 0
- No support vectors found
- Solution: Hyperparameters
  - More erratic w/ gradient

RBF

- Memory issues
- Solutions
  - Nyström approximation (rows)
  - Random Proj., RFF, PCA, (cols)

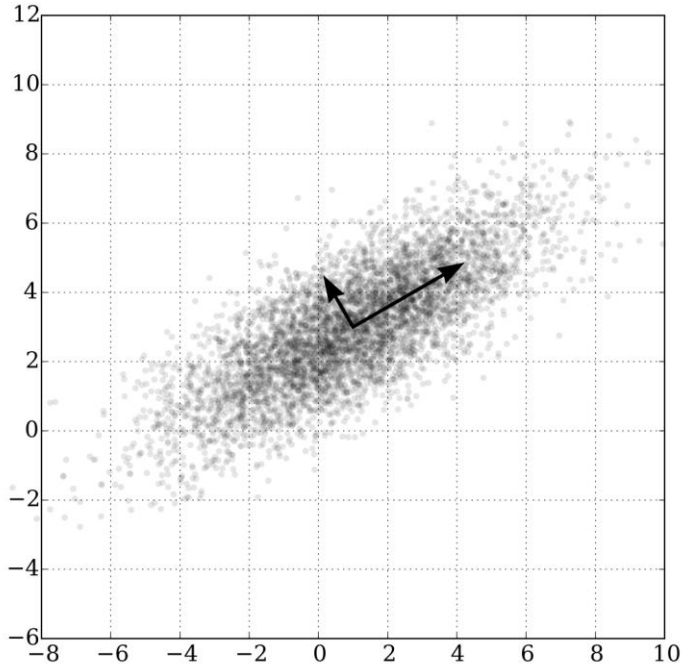
Nyström Approximation (Rows)

- Method for approximating kernel with small subset of data
  - Reduces memory requirements
  - Has regularizing effect
  - Sensitive to subset selection
- Multiple candidates for sampling
  - Uniform sampling
    - Supported by Kumar *et al.*
  - K-means sampling
- Can use Moore-Penrose ‘pseudoinverse’ for speed and to avoid singular K_nm

1. *X_sub = <m rows uniformly sampled of X>*
2. *K_mm = K(X_sub, X_sub) + Id. * 1e-6*
3. *K_nm = K(X, X_sub)*
4. *K_approx = K_nm @ K_nm^-1 @ K_nm^T*
5. *Use K_approx for sub-gradient descent*

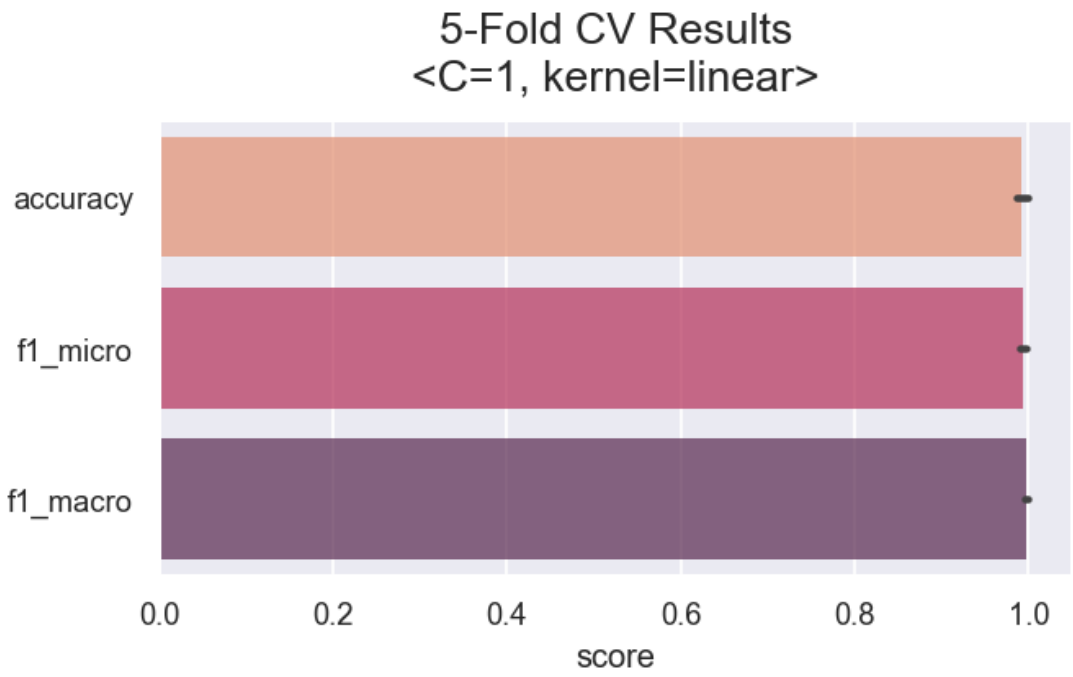
Dimensionality Reduction (Cols)

- Types
  - Random Projection
    - Gaussian & Sparse
    - Fairly fast and simple
  - Random Fourier Features (RFF)
    - Situation dependent
    - Used for dim. reduction here, but likely more suitable for kernel approximation!
  - Principal Component Analysis (PCA)
    - Great for moderately dimensionality
    - Slow for thousands of expression levels
- Most helpful for RBF kernel

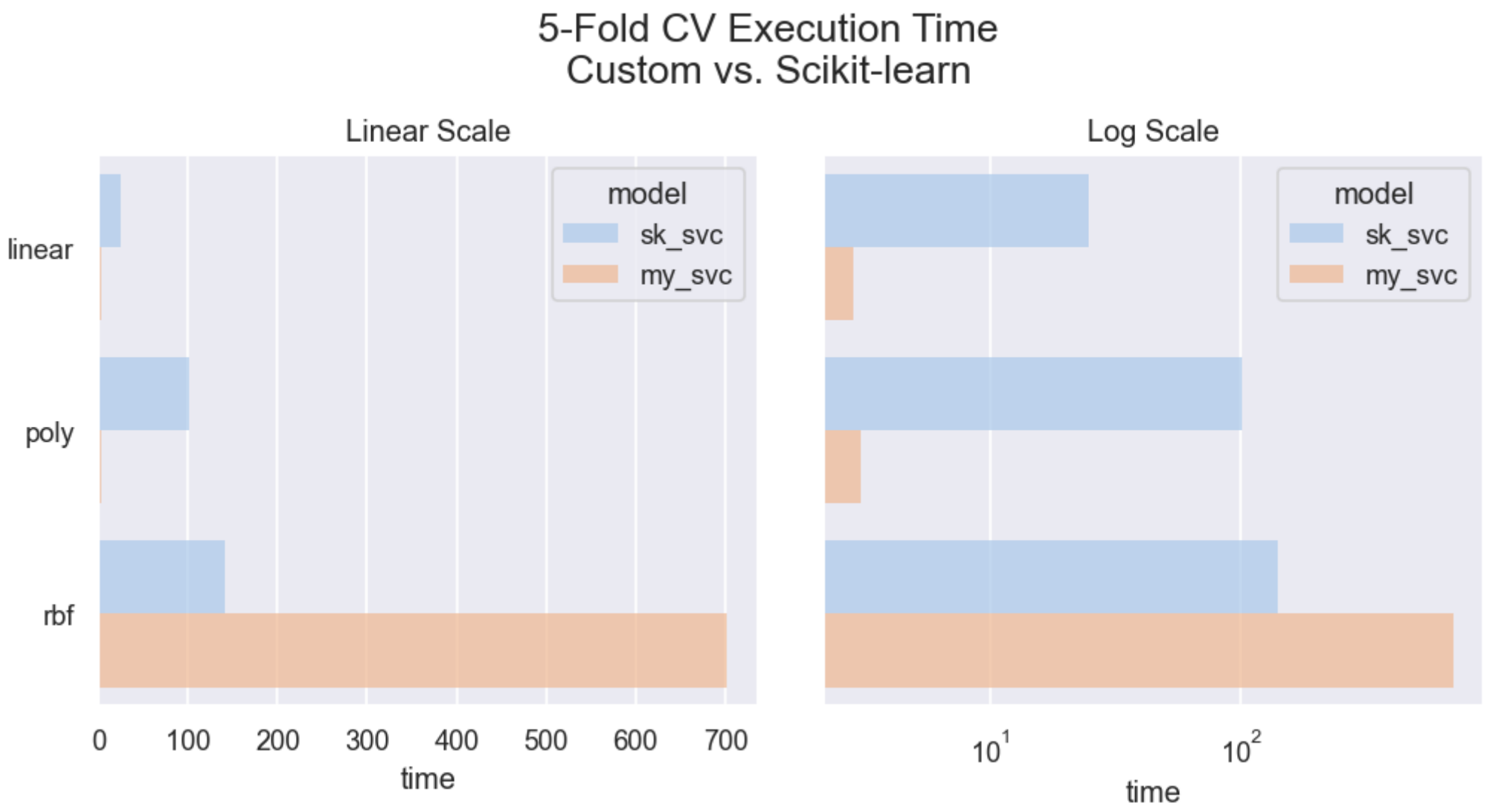


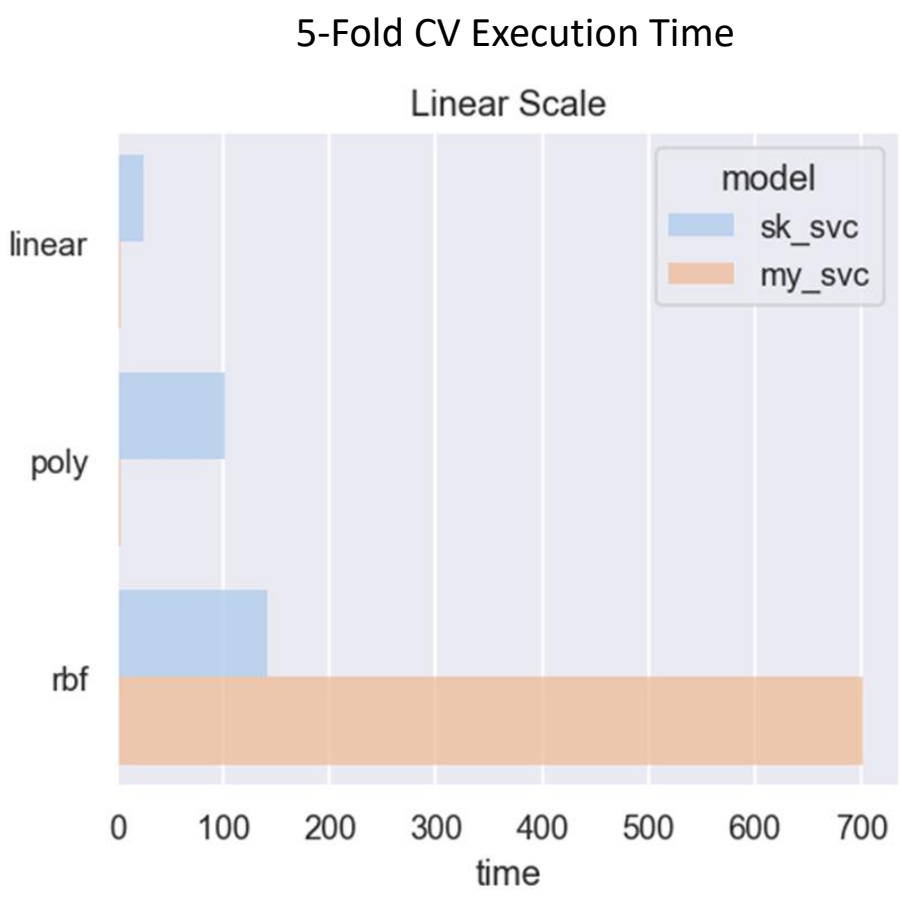
PCA example via Wikipedia

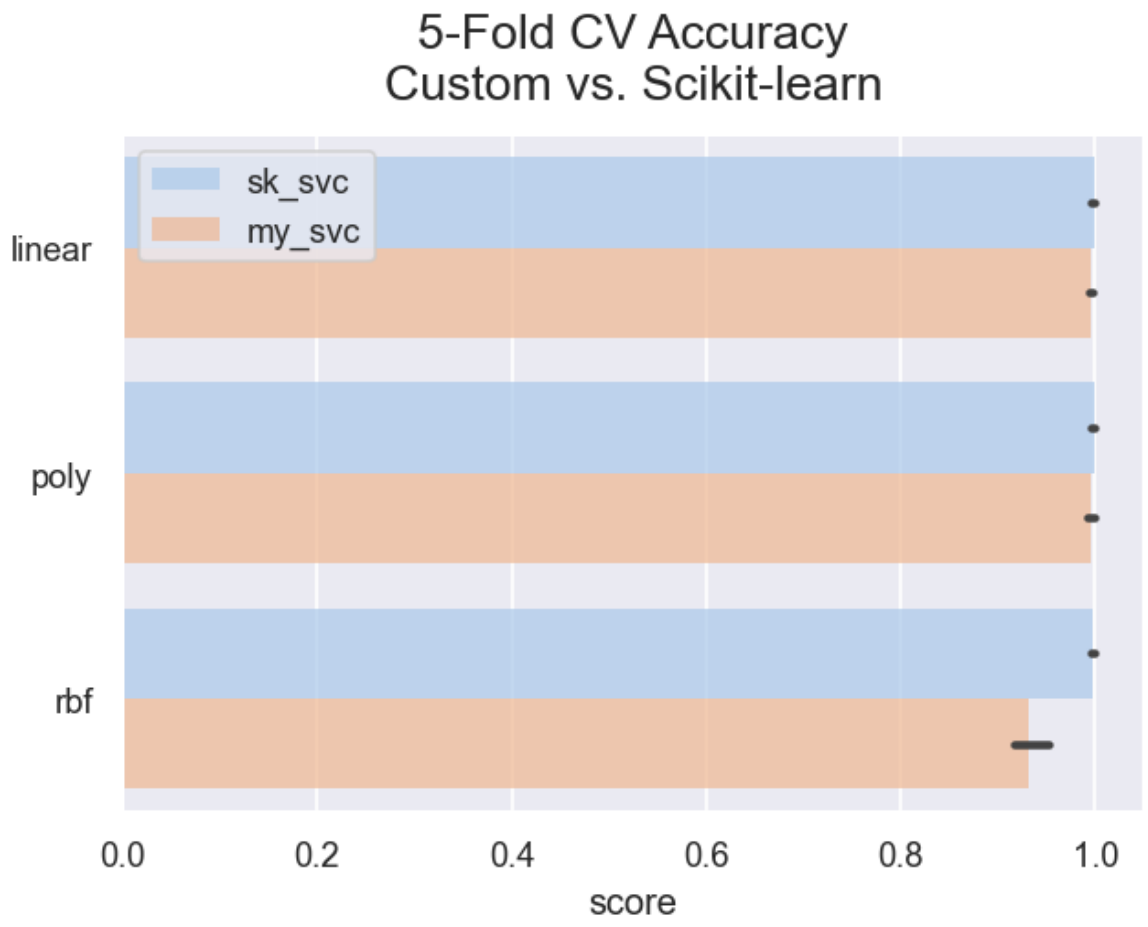
Results



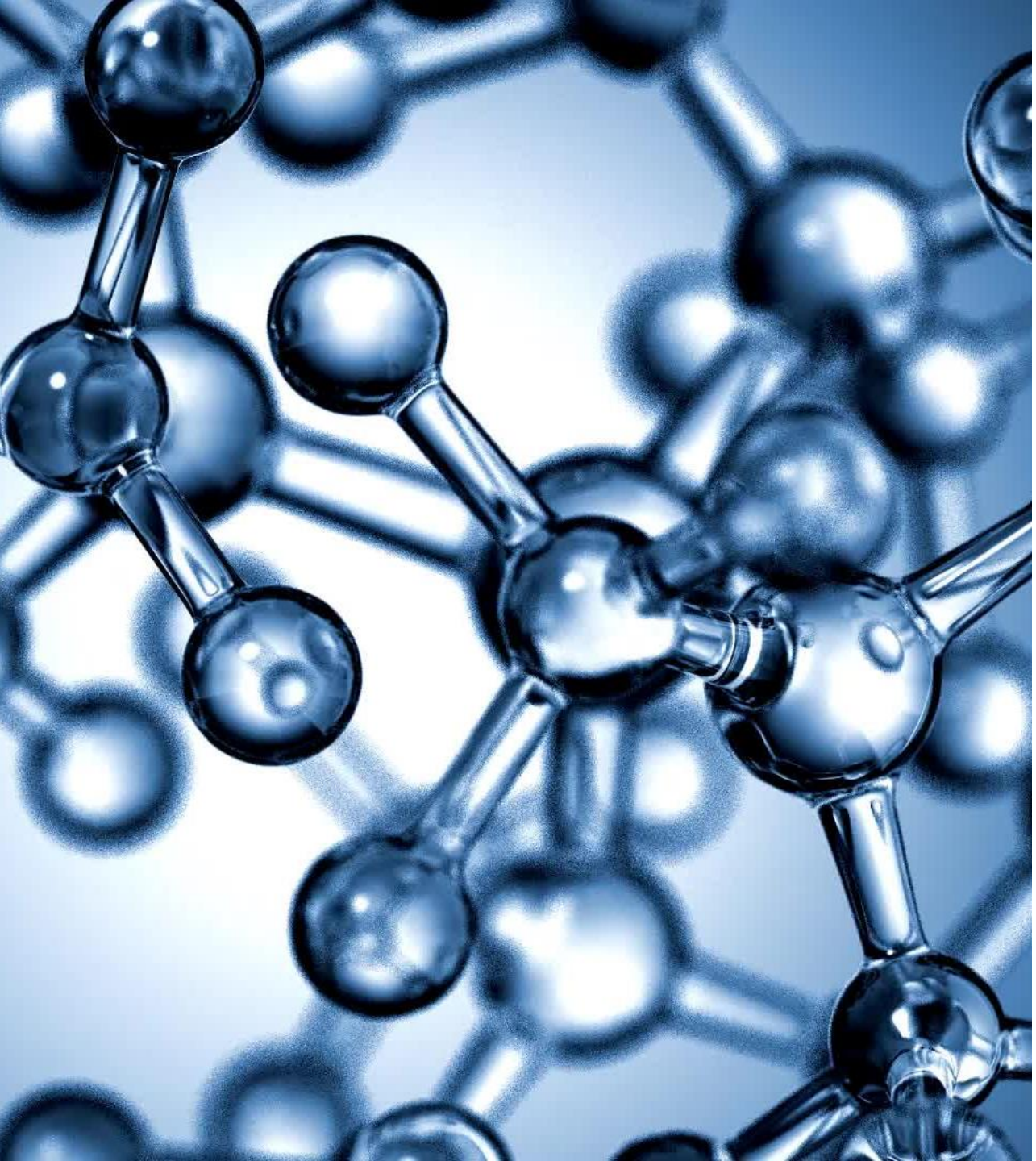
	fold	metric	score
0	0	accuracy	0.987578
1	1	accuracy	1.000000
2	2	accuracy	0.981250
3	3	accuracy	0.993750
4	4	accuracy	1.000000
5	0	f1_micro	0.987578
6	1	f1_micro	0.993750
7	2	f1_micro	0.993750
8	3	f1_micro	0.993750
9	4	f1_micro	1.000000
10	0	f1_macro	0.994600
11	1	f1_macro	1.000000
12	2	f1_macro	1.000000
13	3	f1_macro	1.000000
14	4	f1_macro	0.994811


5-Fold CV Execution Time
Custom vs. Scikit-learn
Linear Scale
Log Scale
model
sk_svc
my_svc
linear
poly
rbf
0
100
200
300
400
500
600
700
time
10^1
10^2
time





Fast… when it converges well (if at all)

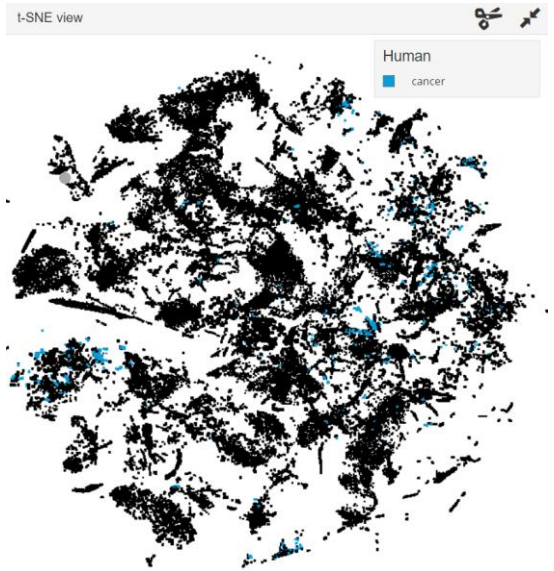


GEO RNA-Seq

GEO data: Barrett et al., 2012

GEO and ARCHS4

- Gene expression omnibus (GEO)
  - Barrett *et al.*
  - Data repository with transcriptome data from many studies
- ARCHS4
  - Lachmann *et al.*
  - Supports access to (somewhat) uniformly processed data and metadata from GEO
  - HDF5 binary data format



Final Dataset

- Gene expression levels
- Collected with R and ARCHS4
- Extracted samples/labels with Python

- 35,240 columns
- 15,554 rows
- 8 classes

35,240
x 15,554

548,122,960

	sample	label	A1BG	A1CF	A2M	A2ML1	A2MP1	A3GALT2	A4GALT	A4GNT	...	bP-21201H5.1	bP-21264C1.1	bP-2168N6.1	bP-2168N6.3
0	GSM1228197	colorectal	30	342	3371	3	5	0	226	0	...	3	2	0	0
1	GSM1177221	prostate	179	9	19413	215	37	3	599	0	...	38	3	0	0
3	GSM1228191	colorectal	100	759	13801	6	7	0	290	2	...	11	3	0	0
5	GSM1177220	prostate	155	12	20213	19	42	14	635	1	...	17	15	0	0
6	GSM1154037	prostate	359	14	4	84	29	3	39	1	...	8	581	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17642	GSM5575423	colorectal	49	357	24121	24	0	0	90	7	...	1	56	0	0
17643	GSM5575426	colorectal	48	2886	20164	11	2	0	125	4	...	1	458	0	0
17644	GSM5575431	colorectal	135	3299	12002	23	0	0	12	0	...	0	1654	0	0
17645	GSM5575435	colorectal	24	1900	18110	11	1	1	131	1	...	0	194	0	0
17646	GSM5575439	colorectal	28	1167	11614	18	0	2	101	0	...	2	98	0	0

15554 rows × 35240 columns

```python
for cancer_type in df.label.unique():
    print(cancer_type)
```

✓ 0.0s

```
colorectal
prostate
breast
lung
gastric
pancreatic
ovarian
kidney
```

Other Additions

- Added shrinking
  - Also discussed by Hsieh *et al.*
  - If a vector's Lagrangian multiplier drops to 0, remove from consideration
  - Speeds up and stabilizes subgradient method
- Experimentation with CUR
  - Described in *Mining of Massive Datasets* ch. 11, Leskovec *et al.*
  - Matrix decomposition method that can be used for dimensionality reduction
  - Not particularly effective

SVD vs. CUR

sparse and small

SVD: $A = U \Sigma V^T$

Huge but sparse Big and dense

dense but small

CUR: $A = C U R$

Huge but sparse Big but sparse

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Additional Experimentation

- Tried Gaussian and sparse projection
  - Sparse performed better
- Mainly stuck to linear kernel
  - Polynomial and RBF has trouble converging
- MinMax Scaling vs. Standard Scaling
- Hyperparameter selection
  - Some small grid searching
  - Mostly manual testing

Results

Custom

```
%%time
svm = MultiGDSVM(
    kernel="linear", C=0.1, nystrom_rows=500, beta1=0.95, beta2=0.999, batch_size=256,
)

t0 = perf_counter()
svm.fit(X_train, y_train)
print(f"Time: {perf_counter() - t0:.4f} s")

y_pred = svm.predict(X_test)
print(f"Test Accuracy: {(y_pred == y_test).sum() / y_pred.shape[0]:.4f}")
```

✓ 1m 42.9s

```
converged (epoch=1108)
converged (epoch=855)
converged (epoch=772)
converged (epoch=722)
converged (epoch=805)
converged (epoch=892)
converged (epoch=1020)
converged (epoch=864)
Time: 102.8323 s
Test Accuracy: 0.9589
CPU times: total: 3min 8s
Wall time: 1min 42s
```

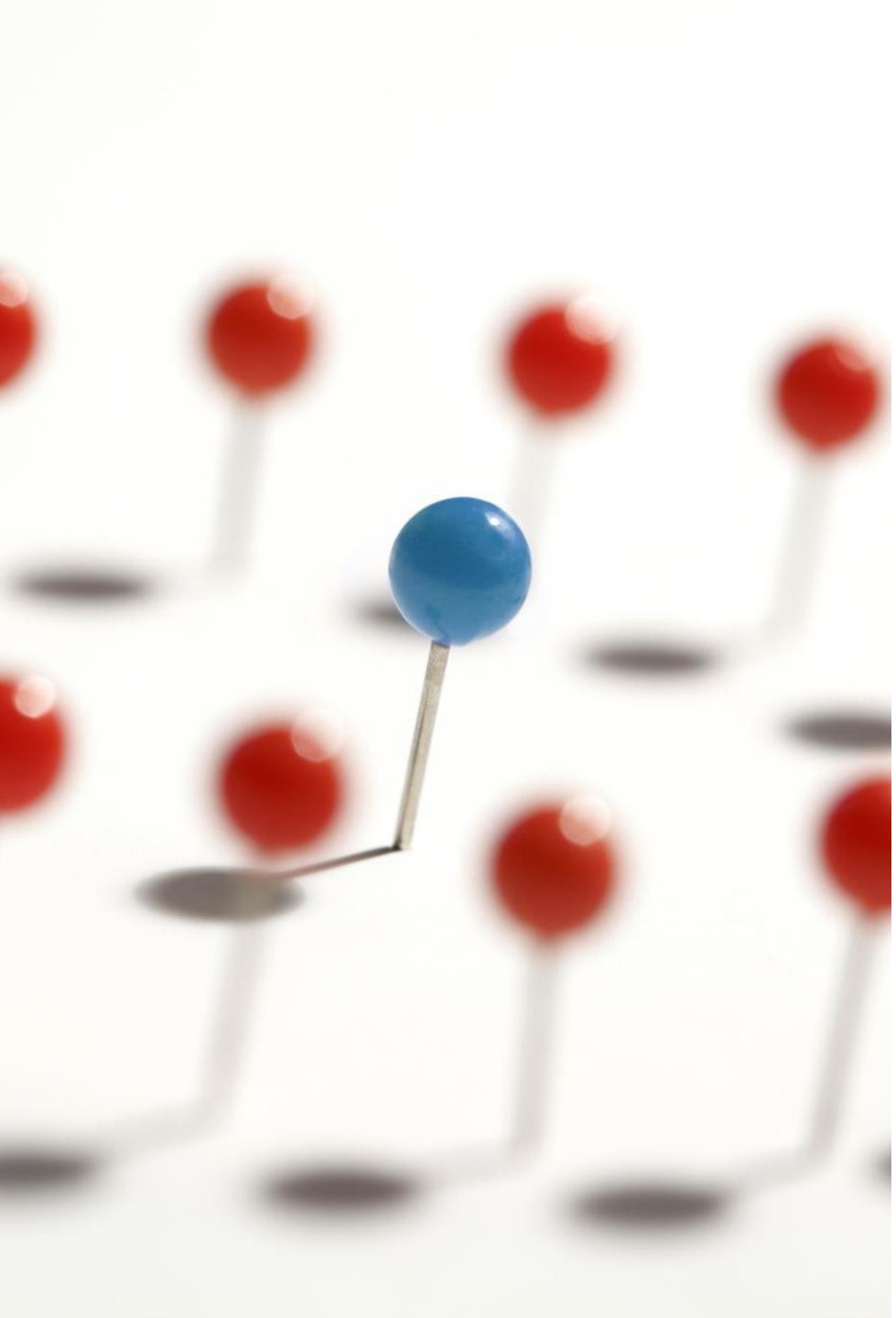
```
custom
Val Accuracy: 0.9653
sklearn
Val Accuracy: 0.9743
```

Scikit-learn

```
%%time
svc = LinearSVC(max_iter=10_000, C=1.0)

t0 = perf_counter()
svc.fit(X_train, y_train)
print(f"Time: {perf_counter() - t0:.4f} s")

y_pred = svc.predict(X_test)
print(f"Test Accuracy: {(y_pred == y_test).sum() / y_pred.shape[0]:.4f}")
```

✓ 3m 41.4s

```
Time: 221.3771 s
Test Accuracy: 0.9794
CPU times: total: 27.1 s
Wall time: 3min 41s
```



Conclusion

Subgradient Method Overview

- Pros
  - Can be very fast when converges
  - Theoretically scalable… for rows
    - Parallelization + Nystrom approx.
  - Could support online learning
- Cons
  - Can be slow when it doesn’t converge quickly
  - Can be ineffective if it doesn’t converge
    - Needs to be suitably tailored to fit data
  - C isn’t as powerful (simply used to clip Lagrange multipliers to [0, C])
  - At least for this implementation, additional dimensionality reduction preferred
    - Improvements to kernel related processing for scalability could help here

Fundamental Challenges

- Stopping criteria
  - Difficult for small batch sizes
  - Must check output or loss changes
    - In this case, checking Lagrange multipliers
- Hyperparameter selection
  - SVM highly configurable
  - ADAM highly configurable
- Dimensionality reduction
  - Memory requirements for RBF in particular
  - Curse of dimensionality
- Difficulties with certain topics
  - RFF/CUR and how to apply them
  - How to manage nonlinear kernels for larger and more complicated data

Areas of Possible Improvement

Parallelization

- Either in memory or otherwise
- One of the big potential strong suits of subgradient method
  - See Kennedy *et al.*
- Ensemble models
  - Example: Cascade SVMs, Graf *et al.*

Other dimensionality reduction methods

- LDA, Autoencoders, variants of t-SNE, other ML related approaches, etc.
- Other sampling approaches for Nystrom approximation

Feature selection

- Many options for exploration

Various optimizations

- Update MultiGDSVM to only calculate Nystrom approximation of kernel once
- Perform more hyperparameter tuning
- Explore techniques for added stability (for poly and RBF in particular)

Transductive SVM

- Potentially valuable for this type of data
  - Relatively few samples
- Bit of a different direction from this project
  - **Probably more appropriate for noisy RNA-seq data!**

Scikit-learn Equivalent? *(Linear only)*

sklearn.linear_model.SGDClassifier

class sklearn.linear_model.**SGDClassifier**(*loss='hinge', *, penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1, n_jobs=None, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False, average=False*) [source]

Linear classifiers (SVM, logistic regression, etc.) with SGD training.

This estimator implements regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). SGD allows minibatch (online/out-of-core) learning via the `partial_fit` method. For best results using the default learning rate schedule, the data should have zero mean and unit variance.

This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter; by default, it fits a linear support vector machine (SVM).

The regularizer is a penalty added to the loss function that shrinks model parameters towards the zero vector using either the squared euclidean norm L2 or the absolute norm L1 or a combination of both (Elastic Net). If the parameter update crosses the 0.0 value because of the regularizer, the update is truncated to 0.0 to allow for learning sparse models and achieve online feature selection.

Topics Explored

- **RNA-seq Data**
  - Transcriptomics
  - HDF5 data
- **SVM**
  - Primal vs. dual
  - Multiclass
  - Kernels
    - Linear
    - Polynomial
    - RBF
- **Regularization**
  - C parameter
  - L1 and L2
- **Subgradient Method**
  - ADAM optimization
  - Stopping criteria
  - Shrinking
- **Nyström approximation**
- **Dimensionality Reduction**
  - Random Projection
    - Gaussian
    - Sparse
  - RFF
  - PCA
  - CUR

Questions?

References

Implementation

- C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, in Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 4 “A Dual Coordinate Descent Method for Large-Scale Linear SVM,” 08–415. doi: 10.1145/1390156.1390208.
- S. Kumar, M. Mohri, and A. Talwalkar, “Sampling Techniques for the Nystrom Method,” in Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, Apr. 2009, vol. 5, pp. 304–311. [Online]. Available: https://proceedings.mlr.press/v5/kumar09a.html
- J. Leskovec, A. Rajaraman, and J. D. Ullman, Mining of Massive Datasets, 2nd ed. USA: Cambridge University Press, 2014.

Data

- W. K. Härdle, Smoothing Techniques: With Implementation in S. Springer, 1991.
- K. B. Gorman, T. D. Williams, and W. R. Fraser, “Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis),” PLoS ONE, vol. 9(3), no. e90081, p. 13, 2014, [Online]. Available: https://doi.org/10.1371/journal.pone.0090081
- J. N. Weinstein et al., “The Cancer Genome Atlas Pan-Cancer analysis project,” Nat. Genet., vol. 45, no. 10, pp. 1113–1120, Oct. 2013.
- T. Barrett et al., “NCBI GEO: archive for functional genomics data sets—update,” Nucleic Acids Research, vol. 41, no. D1, pp. D991–D995, Nov. 2012, doi: 10.1093/nar/gks1193.
- A. Lachmann et al., “Massive mining of publicly available RNA-seq data from human and mouse,” Nature Communications, vol. 9, no. 1, p. 1366, Apr. 2018, doi: 10.1038/s41467-018-03751-6.

Potential Improvements

- R. K. L. Kennedy, T. M. Khoshgoftaar, F. Villanustre, and T. Humphrey, “A parallel and distributed stochastic gradient descent implementation using commodity clusters,” Journal of Big Data, vol. 6, no. 1, p. 16, Feb. 2019.
- T. Joachims, “Transductive Inference for Text Classification Using Support Vector Machines,” ICML, Aug. 2001
- H. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, “Parallel Support Vector Machines: The Cascade SVM,” in Advances in Neural Information Processing Systems, 2004, vol. 17. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2004/file/d756d3d2b9dac72449a6a6926534558a-Paper.pdf